BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

LOW-POWER ELECTRONIC DICE

NÍZKOPŘÍKONOVÁ ELEKTRONICKÁ HRACÍ KOSTKA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JIŘÍ SEDLÁK**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. ZDENĚK VAŠÍČEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2025

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

164571

Institut:	Department of Computer Systems (DCSY)
Student:	**Sedlák Jiří**
Programme:	Information Technology
Title:	**Low-power electronic dice**
Category:	Embedded Systems
Academic year:	2024/25

Assignment:

1. Learn the principles of determining the orientation of an object in space using MEMS sensors and the principles of wireless communication using the Bluetooth LE (BLE) standard.
2. Design a prototype of an active electronic dice communicating with an application using the BLE standard. Focus on the power supply solution and size aspects of the proposal.
3. Implement the designed prototype so that it can be used as a technology demonstrator. The implementation should include an application communicating with the cube.
4. Verify the functionality of the implementation. Discuss the parameters achieved and possibilities for further extensions such as support for a multi-hedral cube.

Literature:

- According to the supervisor's advice.

Requirements for the semestral defence:

- Meeting items 1 and 2 of the assignment.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Vašíček Zdeněk, doc. Ing., Ph.D.**
Head of Department:	Sekanina Lukáš, prof. Ing., Ph.D.
Beginning of work:	1.11.2024
Submission deadline:	14.5.2025
Approval date:	31.10.2024

Abstract

This thesis aims to create low-power electronic dice with physical dimensions comparable to the commonly used ones. To achieve this goal, it was necessary to develop an optimized firmware, custom circuit boards, a companion mobile application, and 3D models. The solution uses Bluetooth Low Energy in both peripheral and broadcaster mode for communication between the mobile application and the dice. The physical orientation of the dice is determined using a MEMS accelerometer with low-power features and a hybrid super capacitor as the primary energy source. The implementation provides excellent flexibility in customizing the hardware and software, including support for multi-sided custom dice.

Abstrakt

Cílem této práce je vytvořit nízkopříkonovou elektronickou hrací kostku s podobnými rozměry, jako mají běžně používané hrací kostky. K dosažení tohoto cíle bylo potřeba vyvinout optimalizovaný firmware a navrhnout vlastní desky tištěných spojů, doprovodnou mobilní aplikaci a 3D modely. Výsledné řešení používá technologii Bluetooth Low Energy jak v periferním tak broadcast módu pro komunikaci mezi mobilní aplikací a hrací kostkou.K určení orientace v prostoru je použit MEMS akcelerometer s nízko příkonovými funkcemi a hybridní superkondenzátor jako hlavní zdroj energie. Implementace poskytuje excelentní flexibilitu v přizpůsobování hardwaru a softwaru, včeta podpory pro více-stěnné kostky.

Keywords

super-capacitor, Zephyr, PCB, Flutter, low-power, Bluetooth, 3D print, pogo-pin, dice

Klíčová slova

super-kondenzátor, Zephyr, DPS, Flutter, nízká spotřeba, Bluetooth, 3D tisk, pogo-pin, kostka

Reference

SEDLÁK, Jiří. *Low-power electronic dice*. Brno, 2025. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Zdeněk Vašíček, Ph.D.

Rozšířený abstrakt

Hrací kostky jsou součástí mnoha her a poskytují prostředky pro vývoj nových herních mechanik a zapojení hráčů do hry. Se zvyšující se popularitou online her však musí hráči používat online generátory náhodných čísel, které mohou ubírat z celkového požitku ze hry. Sice již existují komerčně dostupné varianty elektronických hracích kostek, ty ale neposkytují příliš možností uživatelského přizpůsobení a mohou být také relativně drahé.

Tato práce se zabývá návrhem a realizací elektronické hrací kostky, která kombinuje nízkopříkonový provoz, kompaktní rozměry srovnatelné s normálními kostkami a široké možnosti přizpůsobení, a to jak po softwarové, tak i hardwarové stránce. Cílem projektu bylo vytvořit zařízení, které bude plně funkční i přes limitovaný přísun energie a zároveň bude komunikovat bezdrátově s mobilní aplikací. Ve výsledné evaluaci bylo za úkol zjisit, jaké jsou náklady na výrobu takového řešení a porovnat je s běžně dostupnými variantami.

Za tímto účelem byl navržen a implementován optimalizovaný firmware, založený na Zephyru, který minimalizuje energetickou náročnost a přitom poskytuje rozsáhlé možnosti uživatelského přizpůsobení. Pomocí technologie Bluetooth Low Energy (BLE) komunikuje s vyvinutou mobilní aplikací, a to jak v broadcast, tak i v periferním režimu. Broadcast režim se využívá pro přenášení padlých čísel, protože je méně energeticky náročný. Periferní režim se potom používá při konfiguraci kostky.

Jednou z největších otázek bylo řešení napájení. Konvenční řešení s dobíjecími bateriemi typu Li-Ion nebo Li-Po neposkytují dostatečnou energetickou hustotu a velmi pomalu se nabíjejí. Proto byl zvolen hybridní superkondenzátor s kapacitou 20 F jako střední cesta mezi klasickými EDLC superkondenzátory a dobíjecími bateriemi.

Za účelem miniaturizace bylo potřeba navrhnout vlastní desky plošných spojů a 3D modely pro samotnou hrací kostku, nabíjecí stanici a programovací adaptér. Deska pro herní kostku má velikost 14×14 mm a obsahuje především mikrokontrolér nRF52810 a nízkopříkonový MEMS akcelerometr FXLS8974. Největší překážkou při návrhu desek bylo propojení kostky s nabíjecí stanicí nebo programátorem. To bylo vyřešeno pomocí pogo-pinů na straně nabíječky a programátoru a připojovacích plošek na kostce. Výsledné 3D modely těl byly navrženy se zaměřením na 3D tisk, nepotřebují tak žádné podpory a v případě nabíjecí stanice ani speciální nastavení.

V neposlední řadě bylo zapotřebí navrhnout a implementovat mobilní aplikaci. Ta je napsaná ve frameworku Flutter a slouží pro zobrazování čísel, které na kostce padly, a pro změnu konfigurace kostky. Tato konfigurace zahrnuje také definování vlastních profilů, které uživateli dovolují vytvořit a používat kostky s jakýmkoli počtem stěn i tvarem. To dovoluje přizpůsobení kostky na specifické hry.

Cena výsledného řešení byla i navzdory vyšším cenám, které se pojí k prototypové výrobě, nižší než u již existujících řešení. Testování vyvážení kostky po vylití přiskyřicí ukázalo dobré výsledky s maximálním rozdílem 2.3%. Zařízení také ukazuje slibné výsledky pro výdž podle naměřené spotřeby v různých režimech. To potvrzuje i testování během realistickém scénáře, které napovídá, že výdrž kondenzátoru je až osm hodin, což až šestkrát vyšší než u běžně dostupných řešení. Nabíjecí čas kostky se typicky pohybuje okolo patnácti minut do plného nabití.

Výstupem práce je tedy kompletní řešení elektronické hrací kostky, které je připravené na reálné nasazení. Toto řešení se vyznačuje vysokou mírou flexibility a dovoluje uživatelům nakonfigurovat a používat kostky jakýchkoli tvarů a poskytuje praktický způsob nabíjení pomocí nabíjecí stanice. Součástí výstupu je také mobilní aplikace s jednoduchým a uživatelsky přívětivým rozhraním pro zobrazování čísel, které na kostce padly, a pro nastavování parametrů kostky.

Low-power electronic dice

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of doc. Ing. Zděněk Vašíček, Ph.D. I have listed all the literary sources, publications, and other sources that were used during the preparation of this thesis.

.....
Jiří Sedlák
May 13, 2025

Acknowledgements

I want to thank my supervisor, doc. Ing. Zdeněk Vašíček, Ph.D., for their expert guidance, valuable insight, and help with assembly of PCBs. I also want to thank Ing. Václav Šimek for his help with the dice PCB assembly.

Contents

List of Figures

List of Tables

List of abbreviations

ATT	Attribute Protocol. 10, 11
BLE	Bluetooth Low Energy. 9, 10, 12–15, 27
DFU	Device Firmware Update. 15, 28, 37, 48
DoF	Degree of Freedom. 5
EDLC	Electrical Double Layer Capacitor. 20
FPU	Floating Point Unit. 15
GAP	General Access Profile. 10, 11
GATT	General Attribute Profile. 10–12, 26, 28, 29, 39
GPIO	General Purpose Input/Output. 15
HCI	Host Controller Interface. 9
IMU	Inertial Measurement Unit. 5, 19, 31
LED	Light Emitting Diode. 13, 15, 25, 29, 34–37, 45, 48
MAC	Media Access Control. 38, 40–42
MCU	Microcontroller. 5, 14–16, 19, 22, 24, 33, 48, 53
MEMS	Micro-Electro-Mechanical Systems. 5, 6
OTA	Over the Air. 15, 28, 37, 48
PCB	Printed Circuit Board. 13, 24, 31–34, 36–38, 43, 45, 50, 53
PDU	Protocol Data Unit. 11
PETG	Polyethylene Terephthalate Glycol. 34, 49
SPI	Serial Peripheral Interface. 15, 28, 31, 33
UUID	Universally Unique Identifier. 10

Chapter 1

Introduction

The playing dice are part of many games. However, with the rise of online gaming, players are left with using random number generators, which take away from the physical experience. The available commercial products don't offer much customization, and they are relatively expensive. This project aims to create a highly customizable solution for playing dice and to find out how much it costs to manufacture. The final product allows users to configure any important aspect of it, and at the same time, it keeps the form factor small, comparable to standard playing dice, while providing long-lasting battery life. It can be further adapted to various dice types by changing its shape with outer sleeves.

This thesis is structured into eight chapters, starting from the explanation of the used technologies, continuing to the implementation, and finally a conclusion. The second chapter briefly explains how to determine an object's orientation. The third one introduces the protocol used for communication between the dice and the mobile application. Chapter five provides an overview of existing products and how they solved certain technological challenges, including their upsides and disadvantages. The next chapter explains how the problem is solved, what components will be used, what energy source will power the dice, and finally, a description of the proof-of-concept. Chapter six then explains how exactly all parts of this project were implemented, from the software to the electrical parts, and the physical cases. The second-to-last chapter discusses how well the expected results were achieved, the user test results, user feedback, and the price for each component. Finally, the last chapter recapitulates the achieved goals and suggests possible future extensions.

Chapter 2

Determination of orientation in space

In order to detect what side of the dice is facing up, it is necessary to be able to determine the physical orientation. This is a crucial part of this project, and it is necessary for it to be power-efficient and low-cost as much as possible. There are multiple ways to determine an object's orientation in space, including using the gravitational force of the Earth, the magnetic field of the Earth, or by measuring a rate of rotation. Sensors used for this purpose are called Inertial Measurement Units (IMU), and they differ in what physical properties are observed and used for positional data. To make the orientation detection power-efficient, it would be best to be able to offload it to the chosen IMU. This will limit the choice of sensors, since not every type can detect statically, meaning without constantly keeping track of the measured changes by the microcontroller (MCU). Every IMU has a certain number of degrees of freedom (DoF), which denotes how many independent properties are observed, for example, the number of measured axes and the measured properties [5]. For the purposes of this project, the sensor must be able to measure changes in at least three DoF. Sensors are commonly manufactured using the MEMS Technology, which is more closely described in section 2.1.

Gyroscopes

IMUs that measure angular velocity are called gyroscopes. They are most often used to control and stabilize drones or satellites [7]. Sensors of this type cannot detect the orientation statically and would require constant monitoring of the measured data.

Magnetometers

Another physical property that can be used for orientation determination is the Earth's magnetic field. These sensors are called magnetometers, and they are mostly used in fields such as archeology, magnetic observatories [22] or mineral and oil exploration [35]. They are unsuitable for this project's use case, since any change in the nearby magnetic field would affect the sensor reading. The output data are also dependent on the user's location and angle relative to the Earth's poles.

Accelerometers

Accelerometer IMUs, which are typically used in automotive applications, measure linear acceleration or the time rate of change of velocity, typically as a multiplier of the gravitational constant G, which is equal to 9.81m/s^2. This can be used to determine an object's orientation statically, without keeping track of rotational or speed changes, because accelerometers measure even the gravitational acceleration. [5] For example, if the top surface of an object is facing up, the measured acceleration will have a value of $[0, 0, 1\text{g}]$, the bottom surface will have a value of $[0, 0, -1\text{g}]$, left side might have value of $[1\text{g}, 0, 0]$, and so on (Figure 2.1). This makes them the ideal choice for this project.

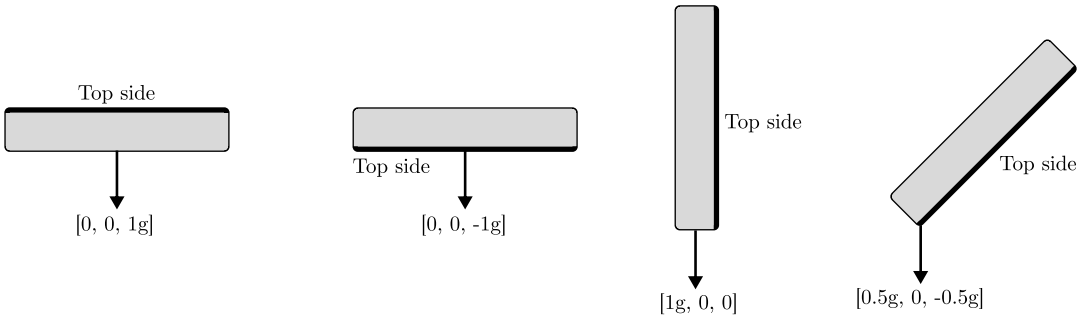


Figure 2.1: Drawing of acceleration values for different rotations.

2.1 Micro-Electro-Mechanical Systems Technology

This technology is better known by the abbreviation MEMS and it is used to create various small-sized devices, ranging from micrometers to millimeters. They combine mechanical components with electronics on a silicone substrate, using integrated circuit batch process manufacturing methods called *microfabrication*. These devices range from simple, with no moving parts, to complex systems with arrays of sensors, electrical, and mechanical components [19] [15] [36].

2.1.1 Manufacturing method

MEMS devices are made using a collection of techniques called microfabrication or micromachining. It most commonly uses silicone as the base substrate because of its excellent mechanical properties. Large single crystals are grown and then sawed into thin wafers with thicknesses ranging from tenths of nanometers to hundreds of micrometers. Then, a pattern is transferred onto the wafers using additive or subtractive processes, for example by deposition of thin films of various materials, or by etching using photoresistive masks.

The mechanical parts inside MEMS include flexible beams, membranes, cantilevers, and mirrors. They are made similarly to the electronic components, by bonding various materials and etching patterns. More complex components, such as accelerometers, micromirrors (Figure 2.2), and micromotors (Figure 2.3), to name a few, are then made by combining them with electronic components [19] [15].

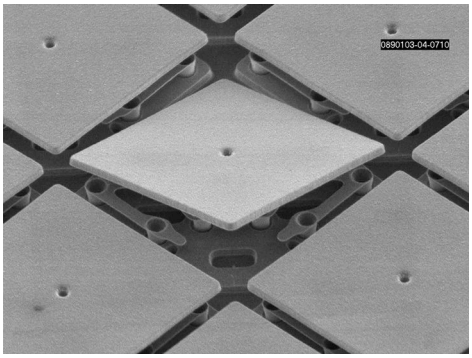


Figure 2.2: Close-up of microscopic square mirrors arranged in a grid pattern. Source: [24].

Figure 2.3: Close-up of micromotor with meshing gears. Source: [38].

2.1.2 MEMS Accelerometers

Accelerometers using this technology can be designed to work by measuring various physical phenomena. The most commonly produced and used are based on piezoelectricity, piezosensitivity, and capacitance transducers [10] [37].

Piezoelectric accelerometers

Accelerometers based on the piezoelectric effect [8], which describes how a mechanical stress applied to an object produces a voltage potential, and similarly, how an electrical charge can result in a mechanical deformation of an object. They consist of a series of cantilever beams fixed on one side and a silicone mass attached at the other. A piezoelectric film is deposited at the fixed end of the beam (Figure 2.4). When an acceleration or pressure is applied to the accelerometer, it will cause the cantilever beam to deform, which will then lead to an electric charge being generated [21] [10] [37].

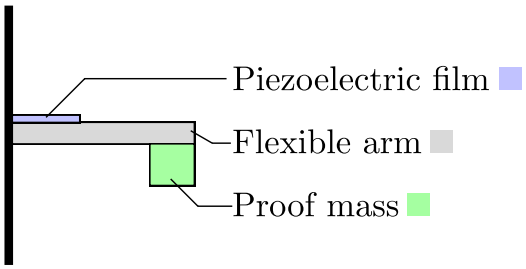


Figure 2.4: Drawing of the general design of piezoelectric accelerometers. Adapted from [37].

Piezoresistive accelerometers

These accelerometers work similarly to piezoelectric accelerometers. They use the piezoresistive effect, which describes how a mechanical stress can change the deformed object's electrical resistance. Their physical construction (Figure 2.5) comprises a series of flexible

beams fixed at one side and connected to a proof mass on the other. Multiple piezoresistors are placed at the location of maximum stress [28] [9] [37].

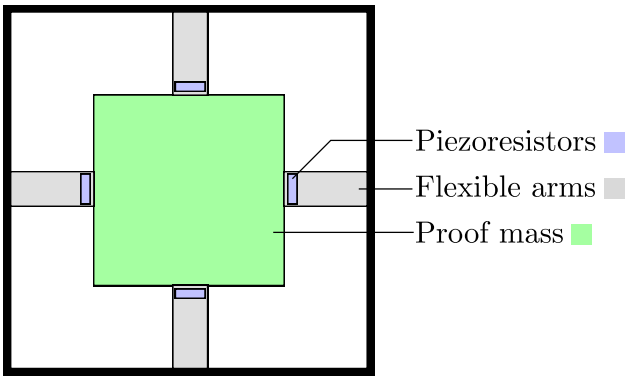


Figure 2.5: Drawing of the general design of piezoresistive accelerometers. Adapted from [9].

Capacitive accelerometers

Capacitive accelerometers measure acceleration by measuring changes in capacitance between two isolated electrodes. The most common structure design is the comb type. This design uses two sets of fingers, where one set is fixed to a metal pad and the other is attached to a suspended proof mass connected to an anchor via a flexible beam (Figure 2.6). When an acceleration is applied to the device, the set of fingers attached to the proof mass moves, changing the measured capacitance [43].

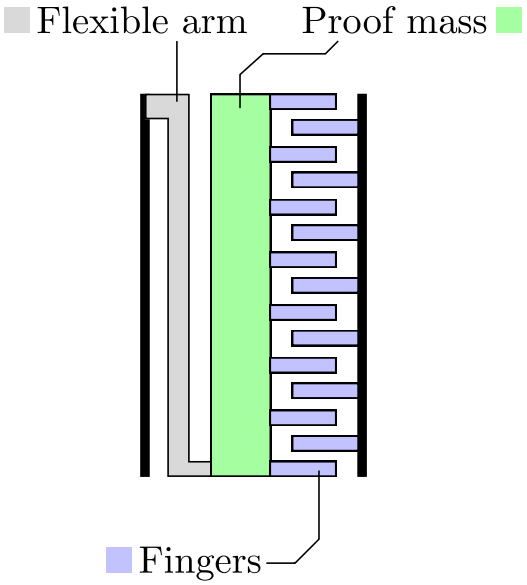


Figure 2.6: Drawing of the general design of capacitive accelerometers. Adapted from [43] and [37].

Chapter 3

Bluetooth Low Energy

In the case of this project, it is necessary to communicate wirelessly. There are multiple options, including a custom transceiver, but the ideal solution would not require an external dongle, and the chosen protocol also must be power-efficiency focused. The Bluetooth Low Energy (BLE) communication protocol is the most suitable protocol as it meets the special demands of this project. It was first introduced in 2010 as a part of the new 4.0 Bluetooth specification. It was designed to be highly power-efficient. It supports one-to-one communication in both connection and connectionless mode, and one-to-many in connectionless mode via broadcasting. Both of these features will be highly beneficial because of the severely limited size of available power sources. It is defined by *Bluetooth Core Specification*, which describes its architecture and key procedures [11] [12].

3.1 Architecture

The BLE stack (Figure 3.1) comprises a host and controller block containing various components. Between these blocks sits the Host Controller Interface (HCI) that acts as a bi-directional logical interface [11] [12]. The main blocks will be briefly discussed in the following chapters.

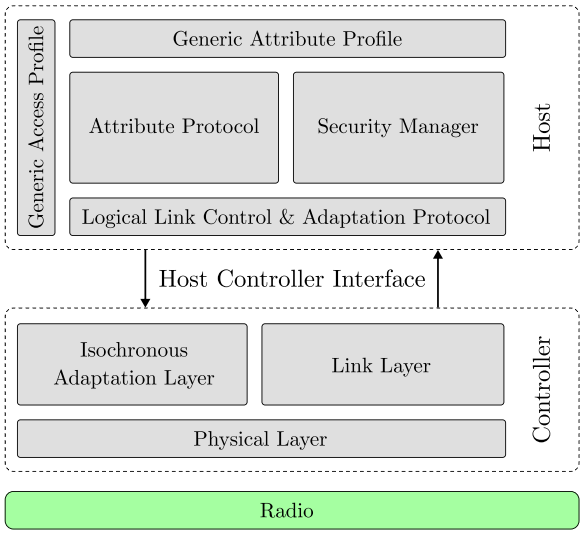


Figure 3.1: BLE architecture stack, adapted from [11].

3.1.1 Controller

The controller represents the device with the radio itself, often something like a Bluetooth module or system on a chip, but it does not have to be physically separate from the host.

Physical layer

This layer describes how data are encoded and decoded for transmission and reception, together with radio parameters, used channels, and their purposes. BLE operates in a frequency band ranging from 2402 MHz to 2480 MHz, divided into 40 channels with a channel width of 2 MHz (Figure 3.2). Channels 37 to 39 are advertising channels, while the rest are for general usage. The core specification defines three modulation variants: the LE 1M PHY, LE 2M PHY, and LE Coded PHY. All BLE devices must implement the LE 1M PHY, while the rest are optional [11] [12].

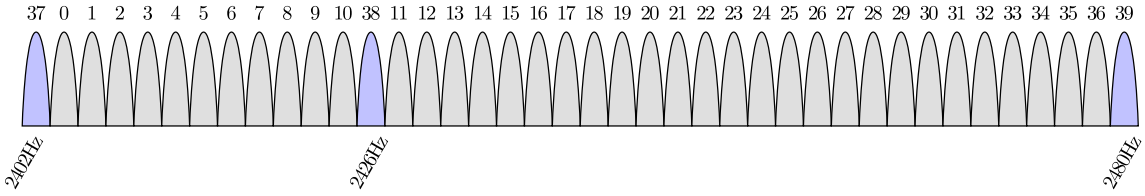


Figure 3.2: BLE channels, with advertising channels highlighted. Adapted from [11] [44].

Link layer

The link layer defines types of packets that can be transmitted and their format, together with states and transitions between them based on the packet type and the link's state. Apart from that, it describes how the channels are used for communication. It supports one-to-one communication in both the connected and connectionless modes and one-to-many communication in the connectionless mode [11] [12].

3.1.2 Host

The host is usually something like an operating system, but the core specification does not mandate specific implementation details [12].

In order to allow communication without requiring each device to have specific drivers, there exists a standardized protocol stack on the host side. This stack includes the Attribute Protocol (ATT), which handles transmission of data in the form of attributes and the General Attribute Profile (GATT), which organizes the attributes into services and characteristics. The stack also includes the General Access Profile (GAP), which defines how devices communicate and discover each other.

Attribute Protocol

ATT defines how data are organized into *attributes*. Attributes are organized on the server side into an array called *attribute table*. The client uses ATT to discover the server's attribute table. Each attribute comprises a handle, which is an index in the attribute table, a universally unique identifier (UUID) that identifies the attribute type, a permission field that defines if the attribute can be read or written to, and the value field, which contains

the attribute's data as a byte array. ATT defines 31 protocol data unit PDU types, some of which are: request, response, notification, and indication. The ATT PDU consists of an opcode, parameters, and a signature. The opcode defines the type of the PDU. Signature is optional. ATT uses transactions, meaning some PDUs expect another specific PDU as a response in a certain time window. For example, the request PDU expects a response PDU. The time window is 30 seconds; if the corresponding response is not obtained during this time, the transaction is marked as timed out. Otherwise, it is marked as completed. [12].

General Access Profile

GAP describes how to perform connectionless communication, discovery, and establish a connection. GAP defines *advertising* and *scanning*, which are methods for transmitting and receiving packets on advertising channels, respectively. [12]

Advertising is done by periodically sending packets on advertising channels. The period between each transmission is defined as a fixed time between 20 milliseconds and 10.24 seconds, with a random delay between zero and ten milliseconds added. The random delay is added to decrease the probability of collisions [44].

Apart from that, GAP also defines four device roles:

1. **Broadcaster**
   This type of device can only transmit data in connectionless communication called broadcasting. It is not connectable, and devices with this role don't have to have a receiver.

2. **Observer**
   Observer devices can only receive data through non-connectable types of communication via scanning.

3. **Peripheral**
   A peripheral is a connectable device to which a *central* can connect. Devices with this role have to possess both a transmitter and a receiver.

4. **Central**
   Devices of this type can connect to *peripheral* devices, and they must possess both a transmitter and receiver.

Generic Attribute Profile

GATT describes *characteristics*, and optionally their *descriptors*, that belong to a *service* (Figure 3.3). All of these types are based on the ATT attributes. A device can have multiple services; the main purpose of a service is to group related characteristics. A characteristic can be included in more than one service. Optionally, one or more descriptors can be attached to a characteristic. Descriptors provide additional information about the characteristic, such as a human-readable name. Each characteristic has a type, value, and set of properties. These properties define how the characteristic may be used, for example, if it can be read or written.

All GATT servers have to implement two special services: the generic access service and the generic attribute service [12]. The generic access service has three mandatory characteristics: device name, appearance, and preferred connection parameters. The generic

attribute service has one optional characteristic that is used to notify the center of any changes in service structure.

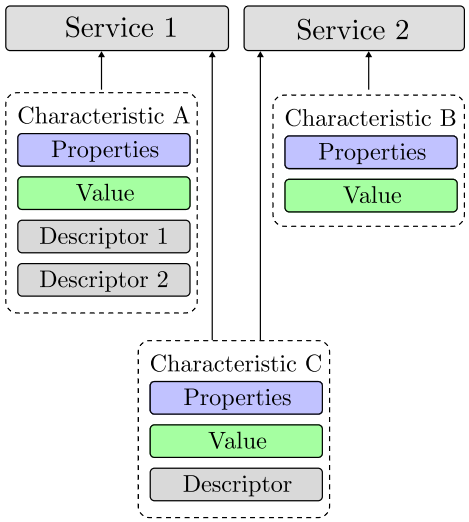


Figure 3.3: Hiearchy of BLE services and characteristics.

3.2 Profile specifications

Profile specifications describe which parts of the core specification are used by certain device types, like heart monitors, blood pressure monitors, hands-free, and many more. This allows devices of the same type to be interchangeable, even if they are made by different manufacturers. Profiles describe how GAP is used for device discovery, how communication takes place on the link layer, and they provide GATT services with characteristics. Profile specifications define the required or optional services for specific applications, together with other additional requirements for both the server and client. It's not strictly necessary to implement one of the available profiles, as devices (such as playing dice) sometimes don't fit into the standard profiles [11] [12].

3.3 Custom Bluetooth Low Energy device

In the case of specialized devices that don't fit into the standard profiles, it is necessary to create custom services and characteristics within them, which will act as a sort of endpoint for communication with other devices, or, for example, a mobile application. This custom implementation can also include defining the device name in the generic access service and adding descriptors to the characteristics with human-readable names. If some parts of the devices can be generalized, such as battery state, it is possible to include profile specifications that provide predefined services and characteristics.

Chapter 4

Existing solutions

There are two, already existing and commercially available, products that implement the same functionality. This chapter will discuss the details of their technical design and potential issues. Since all of them use BLE for communication, this chapter will focus on the other aspects of their design.

4.1 GoDice by Particula

GoDice[1] uses a supercapacitor that is charged within 10 seconds and can be used for up to 2 hours of playtime. It features one Light Emitting Diode (LED), and its light is scattered around the whole shell of the cube. Using a separate charging station, the dice is charged through pogo pins in the number 5. Their application is used as a platform for games that can be played using the GoDice, but they offer integration with Discord[2], Foundry VTT[3], and Role20[4]. GoDice is usually sold in a pack of six, together with a charging case and other accessories for 2660 CZK.

4.2 eDice by Systemic Games

This solution[5] uses a 60mAh battery with wireless charging. Charging time is claimed to be around 90 minutes, and the battery can last up to 10000 throws when all LEDs are turned off. When LEDs are turned on, it can last between 30 and 60 minutes. This dice uses a flex Printed Circuit Board (PCB) that wraps around the whole dice, allowing an LED to be placed under each side. Each type of dice requires a unique charging case. A mobile application can be used to change lighting patterns on the dice. Charging multiple dice at once is solved by providing a case that fits eight dice simultaneously. A single dice of any type with a charger costs around 1300 CZK.

[1] https://particula-tech.com/pages/godice
[2] https://discord.com/
[3] https://foundryvtt.com/
[4] https://roll20.net/
[5] https://gamewithpixels.com/

Chapter 5

Proposed solution

The proposed solution consists of two main parts: the playing dice and a mobile application. It will use the BLE protocol for communication between the dice and the companion application. The mobile application will only serve as a user interface for displaying the number the dice landed on and for accessing the dice configuration. This makes it easier to heavily modify it or develop a new application without changes to the dice. These changes can include different connections to APIs or features tailored for specific games.

The dice will require a microcontroller, an accelerometer, and a source of energy. The microcontroller will be used for the wireless communication and for controlling the accelerometer, which will be used for the determination of the physical orientation of the dice. The chosen power source has to fit within the dice and provide long enough uptime. Specific details of the selection criteria are described in this chapter, together with the selection process and finally the selected components.

To charge the dice, a separate charging dock will be designed with the circuitry required by the specific selected power source. This charging dock will connect to the dice via pogo pins; the same goes for a programming adapter, which will be required to program the dice at least once.

Choosing criteria for microcontroller

Besides having BLE connectivity, the main criterion for choosing the microcontroller was current consumption in the lowest-power mode available on each chip (on most of them, this mode is called deep sleep). This is because the microcontroller (MCU) will be in this state most of the time, for example, when the dice are not used or between each throw. It also had to be checked if each microcontroller could be woken up using an interrupt on at least two pins from this lowest-powered state. If this were not the case (e.g., the MCU could only be awakened by reset), I used data for current consumption from another low-power mode that enabled this.

Current consumption during data transmission over BLE was also considered. However, the weight for this parameter was much lower since the MCU will be transmitting only for a very short period compared to the idle state.

Another important parameter was the size of the chip itself. While most are between 5x5mm and 6x6mm, it was also essential to consider the additional passive components, such as a crystal, bypass capacitors, and pull-up/pull-down resistors.

Because the most straightforward hardware implementation of the dice only needed two pins for the I2C bus or 4 pins for Serial Peripheral Interface (SPI), and two pins for the interrupts from the accelerometer, the GPIO count or other features, such as pulse width modulators or timers, were not considered when choosing the MCU. The same applies to the actual speed of the MCU or the presence of a Floating Point Unit (FPU). However, spare General Purpose Input/Output (GPIO) pins could drive LEDs around the dice or a piezo buzzer for audio feedback. Since the over the air (OTA) device firmware update (DFU) functionality isn't critical for this project, and because all of the considered BLE-capable microcontrollers had minimal flash size of 192kB and RAM size of 24kB, this parameter was also excluded from the comparison.

5.1 Choosing a microcontroller

Looking at the choosing criteria, a few microcontrollers from the three most popular major manufacturers of microcontrollers that have BLE functionality were looked into:

Espressif

The selection included the ESP32-C3 [17] and ESP32-C6 [18] microcontrollers, primarily due to their widespread popularity, the extensive availability of various developer kits and modules, and pre-existing familiarity with these models. They both come in suitable QFN-32 packages with a size of 5x5mm. Nevertheless, they demonstrated worse power efficiency compared to alternative options, particularly in data transmission scenarios.

STMicroelectronics

Two series of chips were looked into, the STM32WB and STM32WBA, more specifically, the STM32WB10CC [39], the bottom range of the WB series, and the STM32WBA52 [40]. Both offer excellent current consumption while in deep-sleep mode, but the WB10CC has above-average current consumption while transmitting data, and it also has a large package of size 7x7mm (QFN-48). The WBA52 features good current consumption while in deep sleep and transmitting, but it was difficult to find development kits that would leave enough budget for the rest of this project.

Nordic Semiconductor

Multiple chips from this company were selected because they offer a broad range of Bluetooth LE solutions. Since their newest series, the nRF54, is unavailable, I compared various MCUs from the nRF52 series. I selected two from the lower range, two from the middle range, and one from the higher end of the series. I also included the nRF5340 from the nRF53 series.

Most of these chips differ in the number of peripheral modules and protocols they support. This made the nRF52832 [31], nRF52833 [32], and nRF52840 [33] fall into the same category, offering more features that weren't needed while having slightly higher current consumption.

The nRF52805 [29] has the lowest power consumption, but it is only available in the WLCSP package, which is unsuitable for assembly using available hobby equipment. Furthermore, development boards for this MCU are very scarce.

The only MCU from the nRF53 series — the nRF5340 [34] had exceptionally low current consumption while transmitting data. However, its large package size of 7x7mm and current consumption in a deep sleep of 1.3uA made it unsuitable for this use case.

The last chip is the nRF52810 [30], which has the same current consumption in deep-sleep mode and while transmitting data as the nRF52805, but it is available in a QFN-32 package with a size of 5x5mm.

5.1.1 Chosen microcontroller

The **nRF52810** was chosen for this use case because of its low power consumption and plenty of developer kits available. Its small package size will fit onto the PCB, and the package type makes it suitable for assembly using hobby equipment.

The chosen MCU has an ARM Cortex-M4 32-bit processor running at 64 MHz. It features 192kB of flash and 24kB of RAM. It supports Bluetooth 5 with a maximum speed of 2 Mbps with output power ranging from -20 dBm to +4 dBm. It also has an integrated DC/DC converter that can be used instead of LDO to decrease power consumption further.

MCU	**Current consumption in deep-sleep mode**	**Current consumption while transmitting data**	**Size and package**
ESP32-C3	5uA	278mA – 335mA	5x5mm (QFN-32)
ESP32-C6	7uA	130mA – 315mA	5x5mm (QFN-32)
STM32WB10CC	0.018uA	8.6mA	7x7mm (QFN-48)
STM32WBA52	0.16uA	5.2mA	5x5mm (QFPN-32)
nRF52805	0.3uA	4.6mA	2.5x2.5mm (WLCSP)
nRF52810	0.3uA	4.6mA	5x5mm (QFN-32)
nRF52832	0.3uA	5.3mA	6x6mm (QFN-48)
nRF52833	0.6uA	4.9mA	5x5mm (QFN-40)
nRF52840	0.4uA	4.8mA	6x6mm (QFN-48)
nRF5340	1.3uA	3.2mA	7x7mm (aQFN-94)

Table 5.1: Selected properties of microcontrollers used in comparison.

Choosing criteria for accelerometer

The main criterion for choosing the IMU was the lowest possible current consumption when not in use. This is so the dice can save power when we don't need to determine its orientation.

The second parameter was the ability to generate interrupts for the microcontroller. Specifically, any motion or shake detection for determining when the dice is thrown. Other interrupt sources, such as double-tap interrupt, were also considered for the possibility of signaling power off and on to the microcontroller.

The third criterion was the package size, which needed to be as small as possible and with the least required external components.

The final requirement was the general availability of the accelerometer chip and development/evaluation boards that could be used for prototyping.

5.2 Choosing an accelerometer

One accelerometer from each significant manufacturer was chosen. When selecting from the manufacturer's range, the above criteria had to be met. If multiple accelerometers fit the parameters, the one with the lowest power consumption while maintaining an acceptable price was chosen.

Bosch Sensortec

First, BMA400 [13] from Bosh Sensortec offers very low power consumption overall. It has the best ODR for current consumption compared to other selected accelerometers: only 0.8uA at 25Hz in the lowest-power setting and 14.5uA with an output data rate of 800Hz. It comes in a small LGA-12 package and requires only two ceramic capacitors. It also features an auto low-power feature, allowing it to switch between states based on detected motion. This unit does not support ODR values lower than 12.5Hz.

NXP Semiconductor

The second on the list is FXL8967AF [20] from NXP Semiconductor. This is the only accelerometer in the list that comes in a DFN-style package. This means that the final design could be assembled at the university. It offers good current consumption with low output data rates. Only two ceramic capacitors are required for final implementation.

STMicroelectronics

From this manufacturer, the LIS2DW12 [26] was chosen. It offers excellent power consumption in low power mode and lower noise density than the other alternatives. Still, it has very high current consumption at higher output data rates (low power mode has a maximum ODR of 100Hz). It also requires an aluminum capacitor for power supply decoupling, drastically increasing the required space.

Analog Devices Inc.

ADXL367 [3] from this manufacturer offers excellent power consumption of 0.18uA in a motion-activated wake-up mode that turns the electronics off between individual measure-

ments. However, it has a very noisy output in its low-power mode with a typical performance of $340ug/\sqrt{Hz}$. This enables the accelerometer to monitor motion, and if it exceeds a certain threshold, it can switch to a different state or generate an interrupt and vice versa. It comes in a small LGA-12 package. The main downside of this unit is the scarce availability of development modules and the high price of evaluation boards, which wouldn't leave enough budget for the rest of this project. Even the IC itself is costlier compared to other options.

TDK InvenSense

While ICM-42370-P [23] has the lowest current consumption among other accelerometers from this manufacturer, compared to other solutions available, it is very high, making it unusable in this project. Specifically, it has a current consumption of 3.5uA even in sleep mode, which doesn't provide any accelerometer readings.

Memsic Inc.

The capacitive accelerometer series from this brand offers MC3635 [27]. It provides a very low-power sniff mode. Sniff mode operates at 1Hz with a current consumption of 0.3uA or 6Hz with a current consumption of 0.4uA. This mode is used for activity detection, leading to a switch to a different, higher-powered state. Current consumption in these states ranges from 0.9uA at 25Hz to 36uA at 1.3kHz in ultra-low-power mode. It lacks support for double-tap detection, which would make it difficult to enter the complete shutdown state of the dice.

Kionix Inc.

The accelerometer with the lowest current consumption from this manufacturer is the KX132-1211 [25]. This unit is designed to operate at 2.5V but can also be used with a 3.3V power source. The current consumption is relatively high, 2.7uA at 25Hz, compared to the other options, making it unsuitable.

5.2.1 Chosen accelerometer

Multiple accelerometers fit the required parameters, and because of this, three units were selected for further testing.

The first is **BMA400** from Bosh Sensortec because of its low power consumption at 25Hz ODR, which could lead to higher responsivity of the dice, auto low power functionality, and many development modules available. It features 12-bit resolution, ODR up to 800Hz, orientation, and double-tap detection. It comes in a small LGA-12 package with a size of 2x2mm and requires only two ceramic capacitors. The main downside to this choice is the package, which can't be reliably soldered using available hobby equipment.

The second one is **FXLS8967AF** from NXP Semiconductor, which comes in the DNF-10 package with a size of 2x2mm and could be used. The advantage over BMA400 is that the package can be used in assembly at the university. However, the selected ODR has to be 6.25Hz or below to achieve current consumption comparable to the Bosh unit, which could affect the response time.

And the last one is **MC3635** from Memsic Inc. Compared to the other two, it has very low power in sniff mode of 0.4uA at 6Hz ODR, but unlike them, it can't automatically

switch back to the sniff mode after an activity has ended, which will require the MCU to handle the no-motion detection.

Table 5.2: Selected properties of accelerometers used in comparison.

Accelerometer	**Typical current consumption**	**Typical noise density**	**Size and package**
BMA400	0.8uA – 14.5uA	$240ug/\sqrt{Hz}$	2x2mm (LGA-12)
FXLS8967AF	0.65uA – 150uA	$230ug/\sqrt{Hz}$	2x2mm (DNF-10)
LIS2DW12	0.38uA – 90uA	$90ug/\sqrt{Hz}$	2x2mm (LGA-12)
ADXL367	0.18uA – 1.3uA	$370ug/\sqrt{Hz}$	2x2mm (LGA-12)
ICM-42370-P	200uA	$100ug/\sqrt{Hz}$	2.5x3mm (LGA-14)
MC3635	0.3uA – 36uA	$230ug/\sqrt{Hz}$	1.6x1.6mm (LGA-10)
KX132-1211	0.67uA – 148uA	$130ug/\sqrt{Hz}$	2x2mm (LGA-12)

5.3 Choosing a power source

Choosing a power source is one of the main problems of the hardware design decisions. It requires the source to be small enough to fit within the dice and to have a big enough energy storage for meaningful real-world use. This section focuses on the comparison of available options and presents the selected one.

5.3.1 Possible solutions

To be able to compare the various options, at least an approximate power draw has to be calculated. Based on the chosen microcontroller and accelerometer, we can calculate the minimal total system current draw I_{system} to be approximately 2uA when the dice is in low power mode and 7.5mA while transmitting data. These values are calculated from the provided current draws of MCU and IMU with a margin to account for other required components.

Only rechargeable power sources were considered to make the dice easy to use. The main reason behind this decision is that changing batteries in small devices could be difficult for some users and potentially lead to damaging the dice. Non-rechargeable sources would also need the dice to have a more robust shell that would have to be openable, significantly decreasing the internal space for electronics.

Embedded energy harvesting source

One alternative source that was considered was generating energy when the dice was thrown or shaken using an inductor with a magnet inside the dice, paired with a capacitor to store excess energy. This solution didn't require any terminals leading out of the dice, but it would make it highly unbalanced. This power source would also require many components, making it unsuitable for this use case.

Supercapacitor

One of the possible power sources is a supercapacitor. The considered categories are electrical double layer capacitors (EDLCs) and a hybrid supercapacitors. They both provide higher capacitances than classic capacitors, but the hybrid supercapacitors typically use doping of the electrodes, which provides higher possible potential and lower self-discharge rates. Doping is a process where metal oxides or polymers are combined with other carbon-based materials [42].

Using a supercapacitor would enable the dice to be quickly charged. With this solution, the dice could remain charged for a long time due to low current consumption in an idle state. On the other hand, it would also quickly discharge during actual use.

The safe voltage can range between 1.7V and 3.6V for the chosen microcontroller. For the selected accelerometers, the minimum voltage ranges between 1.2 and 1.7V, and the maximum is typically 3.6V. This means that the capacitor can be charged to a higher voltage of 3.6V instead of the typical 3.3V, which provides more uptime for the dice.

This solution can be realized with a low number of external components. The central part requires a voltage regulator to generate the required voltage to charge the capacitor. As described above, this voltage would ideally be 3.6V. This part can be placed in a separate charging dock. The second part is a current-limiting resistor, which could also be placed in the charging dock, with a value depending on the maximum current rating of the supercapacitor.

The main disadvantage of this approach is the low transmitting state time it can support compared to rechargeable batteries, but on the other hand, it is charged much more quickly.

Rechargeable batteries

Rechargeable batteries, such as Li-Po or Li-Ion types, are a popular choice and are used across multiple industries, including automotive and phone. The main downside to this approach is that it requires more external components than the supercapacitor solution, with minimal implementation requiring a battery charger IC paired with a voltage regulator and battery protection circuit. This can be partially solved by moving the charging circuitry outside the dice to a separate charging dock, but the regulation and battery protection sections would still need to be placed inside the dice. Due to the constrained size of this system, the selection of batteries is minimal. But batteries that are designed to be used in true wireless earbuds come in small enough packages that would fit inside the dice.

For example, the Li-Ion button battery **LIR1040** has a capacity of 35mAh, a size of 10x4mm, and a nominal voltage of 3.6V. Two could be stacked comfortably, providing up to 70mAh of capacity. The recommended charge and discharge rate is 0.2C, which equates to 7mA with a maximum rating of 1C or 35mA. This limitation could lead to long charge times of around one hour.

Alternatively, a 401010/400909 type Li-Po battery could be used. This battery type has a capacity of around 30mAh with a nominal voltage of 3.7V. The size of this particular type is 4x10x10mm, so two could be stacked, providing 60mAh of capacity combined. The main benefit of using such a battery is the charging speed. Typically, Li-Po batteries can be safely charged at a 0.5C to 2C rate. They also have a lower self-discharge rate than lithium-ion-based batteries.

The additional components would also draw more power if placed in the dice, and while it wouldn't impact the transmission support time much, the idle time would decrease significantly. Even with a minimalist solution, the idle current draw would be approximately 10uA.

5.3.2 Chosen power source

Both the supercapacitor and battery approach have their advantages. The selection between these two primarily depends on the time they can support the system. Supercapacitors are more short-lived but recharge quickly, and the two-cell battery setups last almost half a day if used extensively, but take longer to charge. Choosing between these two comes down largely to personal preference.

The **AHCR-S04R0SA206Q** [4] hybrid supercapacitor was selected, as it serves as a middle ground between these two solutions. It maintains a large capacity while requiring a low number of external components. Even if the capacitor is not fully charged, it can provide enough uptime for real-world use, but the large capacity means it can take advantage of longer pauses in gameplay.

It was selected because it offers the biggest capacity at this size and the highest current rating, which determines charging time. It is a radial can hybrid lithium supercapacitor with an 8mm diameter and 12mm height, with a capacitance of 20F rated at 4V with an ESR of 680mΩ.

Power source	**Design complexity**	**Charging time**	**Idle state time**	**Transmitting state time**[1]
In dice generator	Very high	–	Depending on generated power	Depending on generated power
Supercapacitor	Low	15 minutes	150 hours	70 minutes
Rechargeable batteries[2]	High	1 – 2 hours	3800 hours	5 hours

Table 5.3: Properties of considered power sources.

5.4 Proof-of-concept

I made a first prototype dice from existing, commercially available modules to develop the firmware and to choose one of the accelerometer candidates.

[1] Considering that the microcontroller is transmitting non-stop

[2] One cell only

During firmware design, several issues occurred with the development board that used the nRF52810 microcontroller. Due to these issues, a different board was used. This board (Adafruit Feather nRF52832[1]) has a wide community support and, more importantly, is well documented, but uses an nRF52832 microcontroller. However, both of these MCUs are from the same family, and the additional features of the nRF52832 aren't used, so the code is portable between these two. I also selected the accelerometer for the final implementation during development using this demo dice, by testing the implementation with all three candidates.

The first one considered was the BMA400, because the higher ODR with low power consumption could lead to higher responsivity. It wasn't selected because the double-tap functionality has proven unreliable and often failed to trigger even with the most sensitive settings. The other reason was the package, which was not reliably solderable using hobby equipment. The prototype with the BMA400 module can be seen in Figures 5.1 and 5.2.

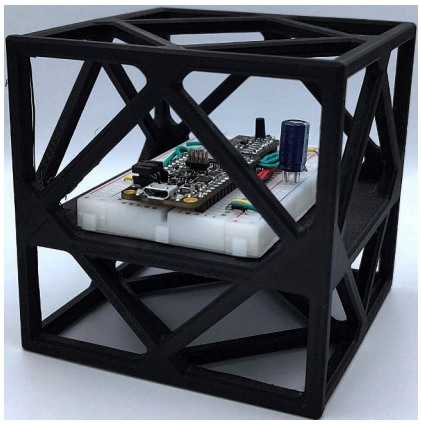


Figure 5.1: First version of prototype dice from side.

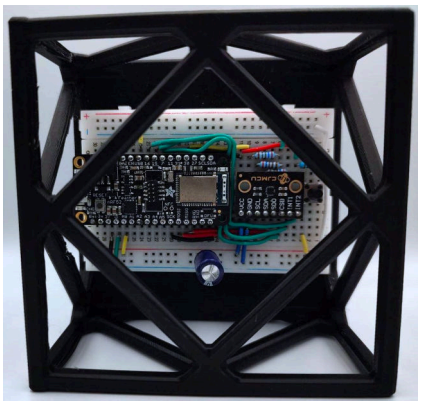


Figure 5.2: First version of prototype dice from top.

The MC3635 offers low power draw in sniff mode, but does not automatically switch back to it when an activity ends. This means that the no-motion detection would have to be done on the microcontroller side, drastically increasing power consumption. This was the main reason for not being selected, together with a package that is hard to hand-solder.

Two major versions of the prototype were made. The first version (Figure 5.1) used a BMA400 module and was used to develop most of the basic firmware capabilities before switching to the second version (Figure 5.3), which uses the NXP development shield with the FXLS8974AF. A complete prototype dice for the MC3635 wasn't made, because its disadvantages became apparent when developing a library for it, and they were severe enough to move to the next candidate, FXLS8974AF.

[1] https://www.adafruit.com/product/3406

Figure 5.3: Second version of prototype dice from accelerometer shield side.

Figure 5.4: Second version of prototype dice from side.

The FXLS8974AF provides power consumption comparable to the Bosh unit if the selected ODR remains below or equal to 6.25Hz, which proved to be sufficiently responsive in the testing. It also comes in a package that is easier to solder by hand, which is why it was chosen for the final implementation. The second-generation prototype, which uses a development shield with this accelerometer, can be seen in Figures 5.3 and 5.4.

Chapter 6

Implementation

This chapter will describe details about all three parts of the implementation, starting from the firmware of the dice, continuing to the schematics and PCB design, and ending with the mobile application design.

6.1 Firmware design

This section will discuss how the firmware for the dice is designed. The primary focus of the firmware was on low-power consumption, which is reflected in how the dice operates. The second criterion was easy extensibility for allowing future expansions. To make the source code easy to work with, several custom Kconfig options have been added to alter some dice behaviour. The controlling library for the accelerometer is implemented standalone and works for the entire FXLS89 family, which makes it usable for other projects.

The firmware was developed using the nRF Connect extension for VSCode, which provides a GUI for projects built on the nRF Connect SDK. This development kit integrates support for Zephyr RTOS with numerous protocol stacks and hardware drivers.

6.1.1 Operational modes of the microcontroller

The state diagram that describes the microcontroller operation is depicted in Figure 6.1. The MCU starts in an initializing state, where it configures the accelerometer. It then transitions to pairing or active mode, depending on the charging dock connection state. It also binds all the functions that will be executed on interrupts and initializes flash storage.

If the dice is placed in the dock, it will transition to a visible state in pairing mode regardless of the activity state. The same applies to transitioning from the pairing mode to the active mode when removed from the charging dock, where the microcontroller will transition to the active state.

Active mode

When in active mode, which starts with the active state, the MCU is placed into a sleep, the interrupt from the accelerometer is enabled, and the Bluetooth module is turned off. The microcontroller reacts to both the falling edge of said signal, which indicates that motion has stopped, and the rising edge, which shows that motion has started.

When a motion starts, the microcontroller will turn on the Bluetooth module and transmit a state message that indicates a `ROLLING` status. The microcontroller can also

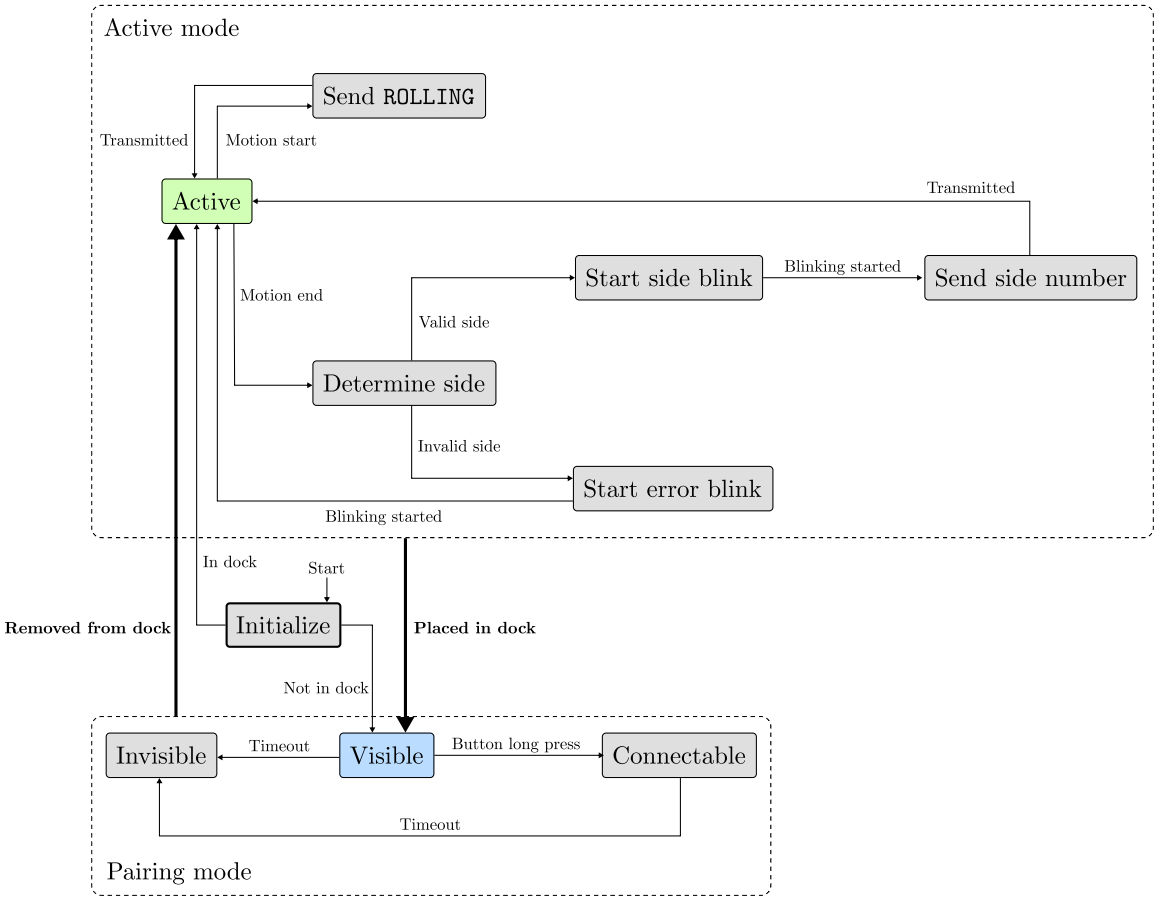


Figure 6.1: State diagram for microcontroller.

send the capacitor's state of charge as the message content, together with the status update. This behaviour is configurable via the mentioned Kconfig options. Sending the capacitor state requires the microcontroller to read from the appropriate ADC channel. After the transmission has completed, the Bluetooth module is turned off.

When a motion ends, the microcontroller will get the acceleration values from the accelerometer and calculate which side is facing up. After that, the Bluetooth module is turned on to transmit that number via non-connectable advertisement, after which it is turned back off. After transmitting, the on-dice LED is flashed multiple times, depending on which side the dice landed on. The transmission of the side number is always finished, even if another motion-stopped interrupt occurs, but the LED blinking will be restarted with the new number.

Pairing mode

The pairing mode starts in a visible state, where the dice appears visible to other Bluetooth devices by sending a non-connectable advertisement. If the button, which is located on the charging dock, is not pressed for a specific time, it will transition into an invisible state. The dice will only leave this state when removed from the charging dock.

If the button on the charging dock is held down for a predefined time, the dice will transition into the connectable state, in which it starts sending a connectable advertisement. If no device connects for a specific time, the dice will transition into the invisible state.

The firmware does not explicitly differentiate between connectable and connected states. The only way they differ is that the connected state has its separate timeout duration, and this time resets on every read or write of any GATT characteristic.

6.1.2 Microcontroller implementation details

The most critical sections of the implementation details of the firmware are closely described in this section.

Important interrupt callbacks

Functions executed on interrupts are defined in `firmware/src/main.c`. All of them are executed on worker threads by submitting a new work item into the work queue after the debounce time has been checked on reception of the interrupt:

- `acc_int1_callback()`
  This function is executed when a motion starts or stops; detection of these events is offloaded to the accelerometer and is represented as a rising edge for motion start and a falling edge for motion end. If the motion has started, other functions are called to notify the mobile application. If the motion has ended, the dice will determine which side it landed on and send that number to the application.

- `dock_btn_callback()`
  This function is executed when the state of the `DOCK_BUTTON` pin changes. If the signal rises, the timer for connecting will start. Otherwise, it will stop the timer, preventing it from finishing. If the timer successfully finishes, the connectable advertisement is turned on.

- `dock_con_callback()`
  This function is executed when a state on the `DOCK_CONN` pin changes. This function will disable the wake-up interrupt on the accelerometer, preventing it from unnecessarily waking up when the dice is docked. It will also turn on the Bluetooth module and make the dice visible to other Bluetooth devices. If the signal is falling, it means that the dice has been picked up from the dock, and the wake-up interrupt is re-enabled. The dice is made invisible, and the Bluetooth module is turned off.

Dice definition data

Necessary dice definition (may also be called dice profile) structures are defined in `firmware/src/lib/side_defs.h/` together with predefined definitions. The definitions are represented as `struct dice_definition`, which consists of a header and an array of sides. The header is of type `struct dice_definition_header` and the array of type `struct side_definition`. The header holds information about the name of the given profile, the number of sides that definition has, its unique ID, and the range that is used when determining the side facing up. The side definition structure then contains information about the number that the side has, the blink mode, and a three-dimensional vector containing the acceleration values for that side.

Non-volatile long-term storage

The dice stores user presets in flash memory using *NVS*[1]. Helper functions for storing and retrieving this information are defined in `firmware/src/lib/storage/`. This includes: global options to toggle blinking on land and error, selected dice profile, and all data in dice profiles. The read functions will write a predefined default value back into the flash memory on an unsuccessful write and return this value. IDs for individual values are either predefined or, in the case of the dice definition data, determined by the ID of that definition and an offset value. The IDs of dice definitions are also stored in a separate array of 8-bit unsigned integers saved in the flash memory.

Determining side

The function that determines what side the dice landed on is defined in `firmware/src/lib/side`. This function iterates through all side definitions of the given dice definition. It compares the given accelerometer values to the vector of that side definition with a margin defined in the dice definition header as `range`. Suppose the side definition vector matches the acceleration values. In that case, the function then compares if the sum of the acceleration values is within $\pm 1g$ with a margin defined as double the `range` value from the dice definition header. If all conditions are met, the function returns the side definition that matched the criteria. If no match is found, the function returns an empty side definition, where all fields are set to zero.

Bluetooth functions

Functions and structures for working with the BLE are defined in `firmware/src/lib/bt`. One of the most critical functions is `dice_bt_broadcast`. This function sends data in a non-connected broadcast fashion via non-connectable advertising. For example, sending the side that has landed on the dice. Sending data this way can reduce the time the microcontroller can't be in a sleep state and the current consumption because the dice doesn't have to connect with the mobile application. The significant disadvantage of this approach is that it is unreliable, and the dice has no way of knowing if the data made it to the host. The message is broadcast multiple times with a unique identifier to mitigate this. Doing it this way increases the chances of the message getting to the mobile phone. Adding the identifier allows the mobile application to ignore messages it has already received. This function sets the manufacturer-specific data to two octets for the manufacturer identifier, one octet for the unique identifier, one octet for the message type (available types are listed in table 6.1), and two octets for message contents.

Other functions may be used to change the Bluetooth status of the dice, for example, setting it to be visible, invisible, or connectable. And it includes an initializer function for the `struct bt_dice_dev` that holds information about specific work queues, Bluetooth state, and timers.

[1] https://docs.zephyrproject.org/latest/services/storage/nvs/nvs.html

Message code	Message type	Description
0x01	`bt_message_dice_number`	Number that the dice landed on
0x02	`bt_message_sides_number`	Number of sides that the dice has
0x03	`bt_message_cap_state`	Current state of charge of the capacitor
0x04	`bt_message_rolling`	Dice started moving

Table 6.1: Bluetooth message types.

When the dice is connected to the mobile application, it exposes several data points through a custom GATT service with specific characteristics (Table 6.2).

When a mobile phone connects to the dice, the dice starts periodic functions that get the state of the capacitor charge and accelerometer values. The accelerometer values are obtained through the manual mode, see subsection 6.1.3. This has been done to avoid running long tasks such as SPI communication in the GATT read callbacks. All characteristics have descriptors attached with a human-readable short name denoting their purpose.

The update characteristic provides a unified way to add new dice or side definitions, or to delete or update existing dice or side definitions. All of those actions are done by sending an update header, which is defined as `struct bt_update_header_t`. This header contains the requested action, which is defined by `enum bt_update_action_t`, dice definition ID, which can be set to zero, in case of adding a new dice definition, and a side definition index, which has to be set to valid value only if updating or deleting an existing side definition. This header should be followed by data, which are dependent on the selected action. For updating or adding a dice definition, this data must contain dice definition header. Similarly, for updating or adding a side definition, the data have to contain a side definition. This characteristic also provides the last new dice definition ID, which is updated after an add dice definition update action has been successfully issued and run.

For issuing various commands, such as a restart or a factory reset, there is a command characteristic available. Available commands are defined by `enum bt_command_t`. All commands are executed immediately after they have been received. The restart command will cause the dice to be cold rebooted.

The capacitor state is reported through a custom characteristic instead of using the provided battery service. The reason for this is the limited space on the device, which meant that the OTA DFU could not be implemented. This did not allow for any potential fixes that would have to be made in the firmware. Providing the capacitor state this way is simpler on the firmware side, provides more flexibility, and shifts more work onto the companion mobile application. It is provided as an unsigned 8-bit integer with the value 255 representing a voltage of 3.6V.

Name	Permissions	Description
Side blink	Read and write	Get or set the current global side blink setting.
Error blink	Read and write	Get or set the current global error blink setting.
Current dice definition	Read	Provides full dice definition (header and sides).
Supported dice definition IDs	Read	Provides list of IDs of all available dice definitions.
Current dice definition ID	Read and write	Allows for getting the currently selected dice definition or changing it by writing a new ID.
Selected dice definition header	Read and write	Writing to this characteristic selects a definition. The selected definition's header is then available for reading from this characteristic.
Dice and side definition updates	Read and write	Unified endpoint for all updates. When writing, the data has to be prepended with an update header, defined as `struct bt_update_header_t`. Last new dice definition ID is returned when reading. (update on add dice definition update action).
Acceleration values	Read	Provides last measured acceleration values. These values are updated periodically in the background with default period set to 100 milliseconds.
Commands	Write	Provides a set of commands defined by `enum bt_command_t`. Value of specific command can be written to this characteristic, upon which the command will be executed immediately.
Capacitor state	Read	Provides last measured capacitor state as 8-bit unsigned integer. This value is updated periodically in the background with default period set to 100 milliseconds.

Table 6.2: Available GATT characteristics.

Physical behaviour

Functions and structures representing the physical dice and dock are defined in `firmware/src/lib/dock` and `firmware/src/lib/dice`. These functions are used to control LEDs and hardware associated with the dice and dock. The dice has predefined functions for controlling LEDs in special modes, such as error blinking or number blinking, and a function for getting the current state of charge of the capacitor. The dock has functions for assessing the state of the connection button and connection pin.

6.1.3 Accelerometer operational modes

A simplified state diagram describing the accelerometer operation is depicted in Figure 6.2. It starts in the standby state, in which the microcontroller can configure any required parameters. Then, the accelerometer behavior can be toggled into two different modes. The first is an interrupt mode, where the accelerometer is set to route its wake state as an interrupt to pin `INT1`. It also has a configured *within-threshold* interrupt, which remains unrouted and is only used internally. When the accelerometer detects that the measured acceleration is outside of the defined range, it will wake the unit up, transitioning to the wake state and marking the start of a motion. After a timeout, that is defined as a number of ODR cycles, is reached, the accelerometer will return to the sleep state, marking the end of a motion. This timeout is reset whenever the acceleration values are outside the within-treshold values.

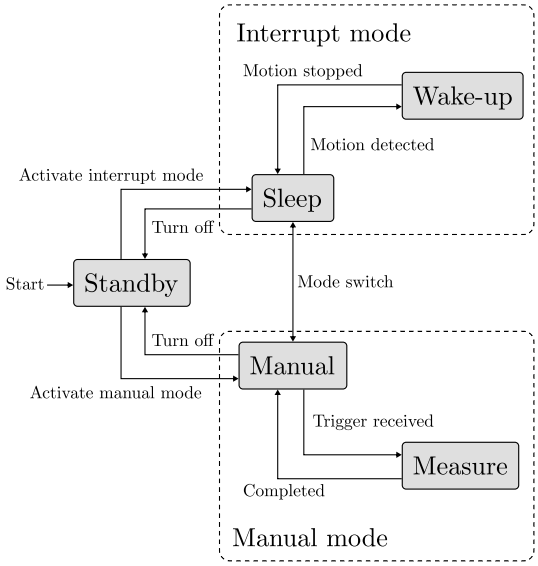


Figure 6.2: State diagram for accelerometer.

The accelerometer provides an independent power setting for the sleep and wake states. The sleep state is configured to use an ODR of 3.125Hz and a low power setting. While this makes the output noisy, it is accurate enough to detect a significant motion reliably. The wake state uses a higher ODR of 6.25Hz to make the transition back to sleep as responsive as possible, since the timeout is measured in ODR cycles. A value of 3.125Hz or lower caused the motion end detection to have a slow response time, delaying the rest of the detection and transmission process.

The second mode is manual, which starts with the manual state. The function of `INT2` is set to `EXT_TRIG`, and the within-treshold interrupt is turned off. In this mode, the accelerometer reacts to a rising edge on the `INT2` pin, and when received, it will transition into the measuring state, where it takes temperature and acceleration data samples. After the measuring is completed, the new values are saved to registers, and the accelerometer transitions back into the manual state.

6.1.4 Accelerometer implementation details

I wrote a separate Zephyr module for communication with the FXLS8974AF. This makes it easy to use in other projects or expand with, for example, I2C communication. This module contains functions for configuring specific parameters, binding a microcontroller pin to a particular interrupt, or manually triggering a read of acceleration values. The most important functions are:

- `fxls89xx_init()`
  This function is not called by the user directly, but it initializes the accelerometer at startup. It also runs a soft reset of the unit and verifies that the `WHO_AM_I` of the used IMU corresponds to one of the valid values from the FXLS89 family of accelerometers.

- `fxls89xx_read()`
  This function returns the value from the given register by first sending a read access flag with the registered address while simultaneously shifting bits from the SPI register to a buffer. After that, the second byte in the buffer is returned, as the first was acquired while transmitting data to the accelerometer.

- `fxls89xx_write()`
  This function is used to change the value of the given register. This is done by sending the register address together with the access mode, in this case `fxls89xx_access_mode_write`, and then sending the new register value.

- `fxls89xx_trigger_read()`
  This function only takes effect if the function of the interrupt pin `INT2` is set to `EXT_TRIG`. It will firstly set the pin to logic low, if necessary, and then to logic high. This causes the accelerometer to take samples of acceleration and temperature values and save them to registers. This function is used to take measurements in the *manual* mode.

Other functions are wrappers using the `fxls89xx_read()` and `fxls89xx_write()` to access and write to the required registers correctly, by providing predefined values and handling the position of individual attributes within these registers.

6.2 Hardware design

This section reviews the hardware designs of all three components: dice, charging dock, and programming adapter. It discusses the proposed electrical schematics and 3D printed assemblies.

Additional external components required by the microcontroller and the accelerometer were chosen, where possible, according to their Datasheets and PCB design reference manuals. Otherwise, components that matched the provided values were selected.

SPI bus is used to communicate between the microcontroller and accelerometer because it requires less power than the I2C bus. I2C needs at least two pull-up resistors that would constantly draw power.

6.2.1 Playing dice

The hardware design of the dice posed significant challenges due to physical size constraints. A way had to be found for the board to be connected to the charging dock and programming adapter, while keeping the size small.

Microcontroller section

The data sheet for the nRF52810 provided all the necessary resources, including a bill of materials and recommended schematic design. Based on the given description, a configuration with the internal DC/DC regulator was used to reduce current consumption as much as possible. Figure 6.3 shows the schematics for the microcontroller section.

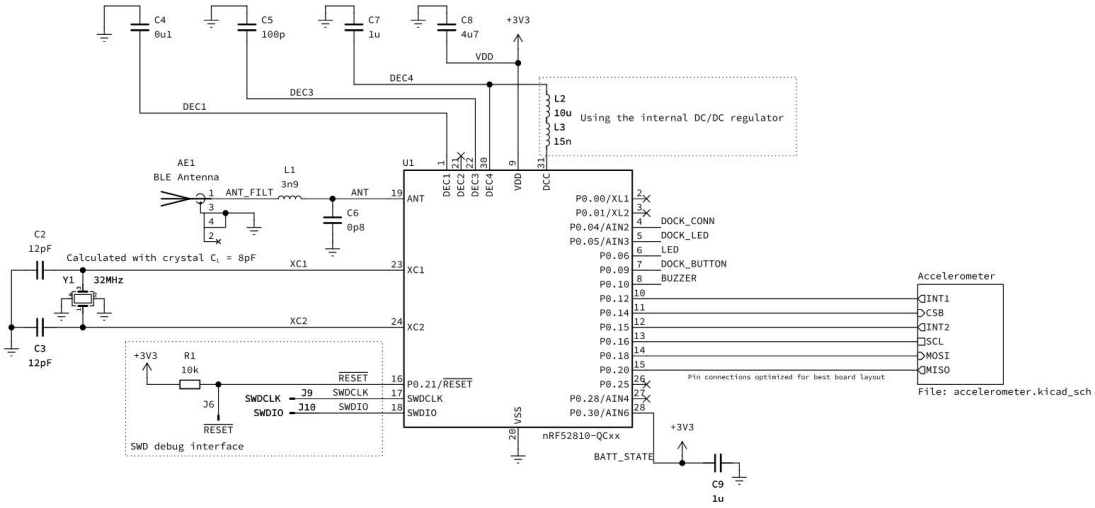


Figure 6.3: Circuitry design for the microcontroller.

Decoupling capacitors were placed as close to the microcontroller IC as physically possible to reduce parasitic inductance, as described in [14]. This includes capacitors for the VDD supply and the DEC pins.

The SWD debug interface pins (SWDIO, SWDCLK, and RESET), together with GND and 3V3, were routed to the pads on the bottom side of the circuit board, see Figure 6.4 for copper layout and Figure 6.5 for full rendering of the board including silkscreen that denotes function of each pad. The same was done for pins designated for the dock, with the idea of soldering small wires between these pads and the actual outer pads on the dice.

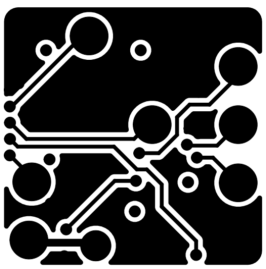


Figure 6.4: Back copper layer (mirrored horizontally).

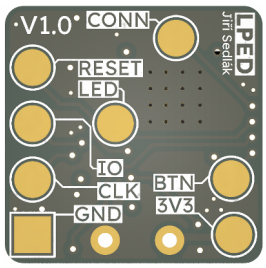


Figure 6.5: Rendered PCB dice design from back view.

Chip antenna **XDCR2450AT42E010BE** [2] was chosen because it requires only small copper clearance on the same layer and copper directly underneath the antenna at a certain distance. It comes in a small enough package to fit the limited size of the circuit board. The components for the impedance matching network were set according to the reference circuitry of the nRF52810, and the feed trace was kept straight and as short as possible. A keep-out area was added under the filter network on the inner layers, as recommended in the datasheet.

Accelerometer module

The selected accelerometer required only two decoupling capacitors for VDD and VDDIO. An additional pull-down resistor was placed on the MISO pin. This was done because the MISO pin is left in an undefined state when the microcontroller deselects the accelerometer by pulling the associated CS pin high. Other pins always have a defined state, either by the MCU side in the case of the SPI communication pins or the accelerometer itself in the case of the INT1 and INT2 pins. However, this resistor can remain unpopulated as the pull-down resistor can be toggled inside the microcontroller. Schematics for the accelerometer module can be seen in Figure 6.6.

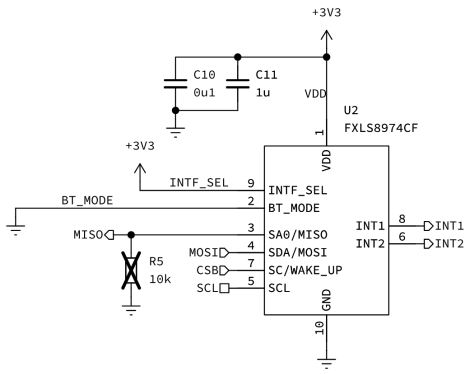


Figure 6.6: Circuitry design for the accelerometer.

The accelerometer is placed in a location with minimal board stress and away from PCB edges, per PCB design guidelines from [1]. Decoupling capacitors were placed as close as possible to the accelerometer, as described in [14].

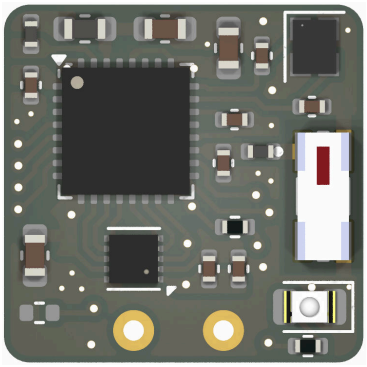


Figure 6.7: Rendered PCB dice design from top view.

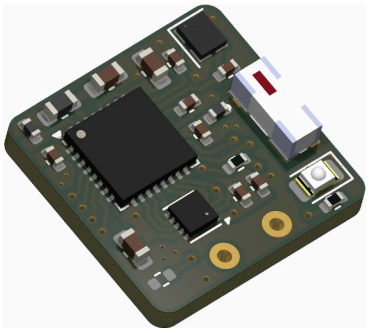


Figure 6.8: Rendered PCB dice design from side view.

The resulting circuit board (Figure 6.7 and Figure 6.8) has dimensions of 14mm by 14mm with four layers, with stackup described in Table 6.3. A four-layer design was chosen to ensure proper RF performance, signal integrity, and power distribution.

PCB Layer	**Description**
L1 (Top layer)	Components, signal routing and ground plane
L2 (First inner layer)	Continuous ground plane
L3 (Second inner layer)	3.3V power routing and ground plane
L4 (Bottom layer)	Connection pads, signal routing, and ground plane

Table 6.3: Dice PCB layer stack up.

It doesn't feature any mounting holes because there is not enough room to place it. Instead, it will be held by friction fit. The two plated holes on the bottom center of the board are mounting pads for the selected supercapacitor that will be mounted on the bottom of the board.

The selected LED for user interaction is **APTD2012LSURCK** [6]. The reason behind the selection is its low forward voltage of 1.75V and its high luminosity of 150mcd at 2mA compared to other LEDs with similar size and forward voltage.

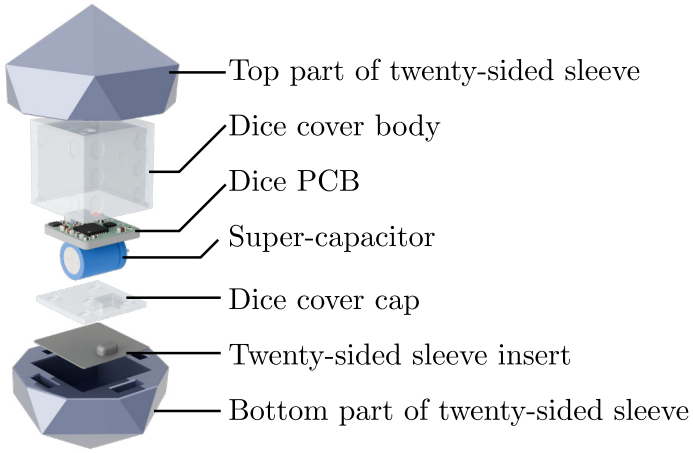


Figure 6.9: Exploded view of dice assembly, including twenty-sided sleeve.

The dice assembly (Figure 6.9) comprises a cover body and cap. The outer shell has a one mm-thick outer wall with recesses for pogo-pin contact pads. While this isn't necessary, it makes the dice look more consistent, as all sides have the same aesthetic. The top cap additionally features a cavity for aligning the dice in the charging dock, and instead of recesses, it has holes through to route the connection wires. Solid core wires with a diameter of 0.35mm were used for 3.3V and ground pads, while the other pads were connected using 0.1mm wires. The cap and the main outer shell were printed from transparent natural Polyethylene Terephthalate Glycol (PETG) filament, which lets the LED shine through, and with a 0.25 mm nozzle because of these objects' thin walls.

Physical support for multi-sided dice is achieved by sleeves. The main six-sided dice module fits in these sleeves, which change the appearance and number of sides of the dice.

The sleeve featured in figure 6.9 has twenty sides, and it is made out of top and bottom body parts and an insert with an aligning key.

6.2.2 Charging dock

The charging dock serves multiple purposes: charging the dice with the required voltage, telling the dice that it is connected to the charging dock, making the dice visible and bondable by pressing a button, and providing user feedback with an indicator LED. Figure 6.10 shows a block schematic that provides an overview of the charging dock design.

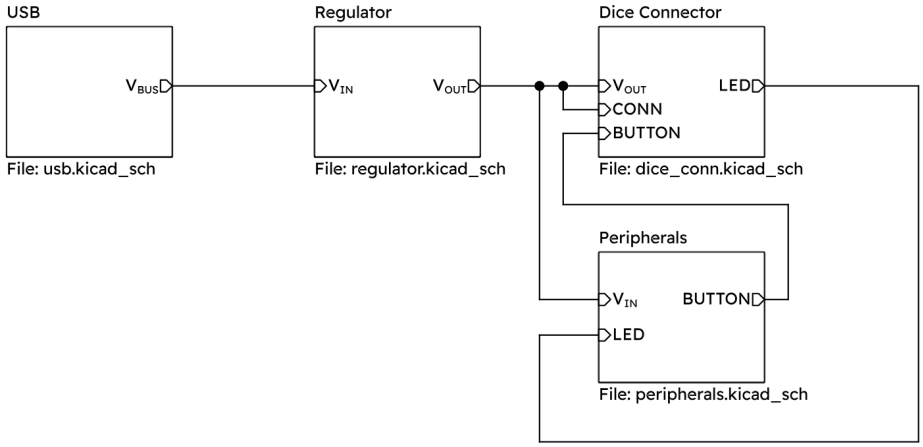


Figure 6.10: Block diagram for charging dock.

To provide power for the dock, a USB-C receptacle was placed on one edge of the PCB, together with ESD protection. One 5.1k resistor was placed on each CC line, providing up to 500mA at 5V, as described in [16]. Schematics describing this section is in Figure 6.11.

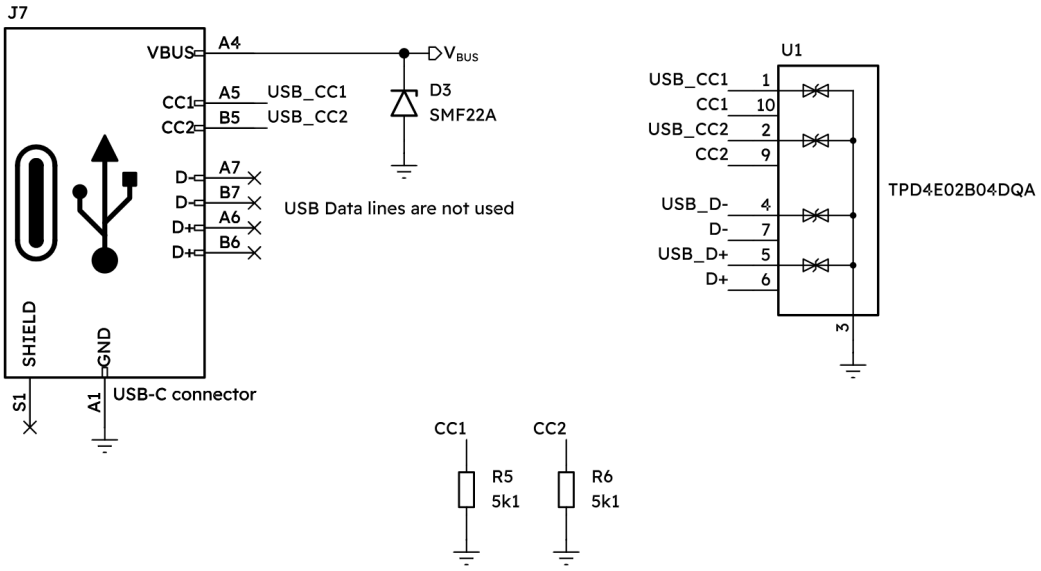


Figure 6.11: Circuitry design for USB-C port on docking station.

The voltage regulator chosen for this application is **TPS560200** [41]. It was selected because it was the cheapest voltage regulator that could step down 5V to 3.6V with the required maximum current of 400mA (this regulator can provide up to 500mA). The circuitry design (Figure 6.12) follows the typical application schematic from the datasheet with feedback resistor values calculated using the provided formula.

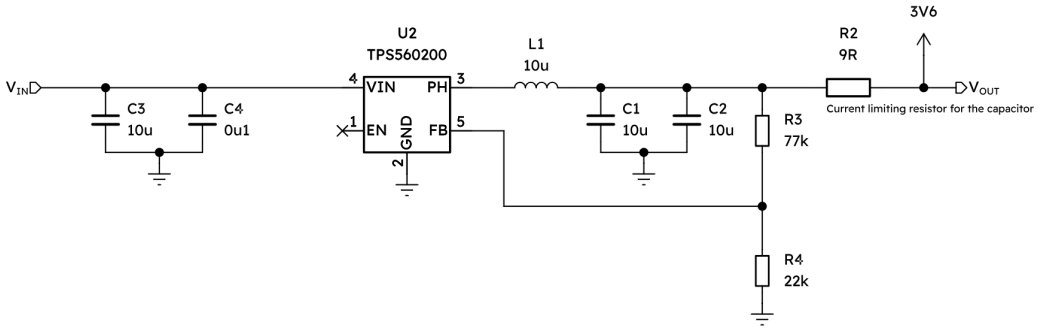


Figure 6.12: Circuitry design for voltage regulator.

The decoupling capacitors and switching inductor were placed near the IC to minimize the loop area, according to layout guidelines in the datasheet.

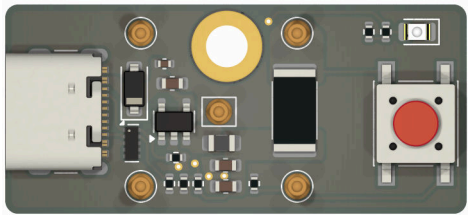


Figure 6.13: Rendered PCB dock design from top view.

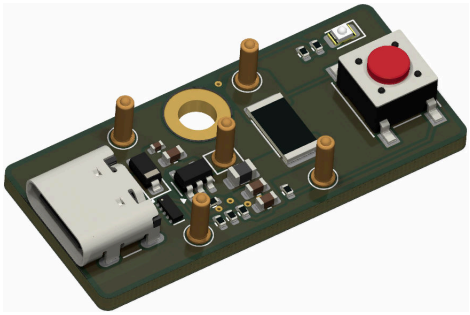


Figure 6.14: Rendered PCB dock design from side view.

The resulting circuit board design (Figure 6.13 and Figure 6.14) has two layers, both of them being used for signal and power source routing with ground planes.

The PCB features one mounting hole for a M3 screw to secure it to the dock shell and five mounting holes for pogo pins. The charging dock also has a dedicated LED connected to the dice through a pogo pin. The LED itself is the same as the one used on the dice.

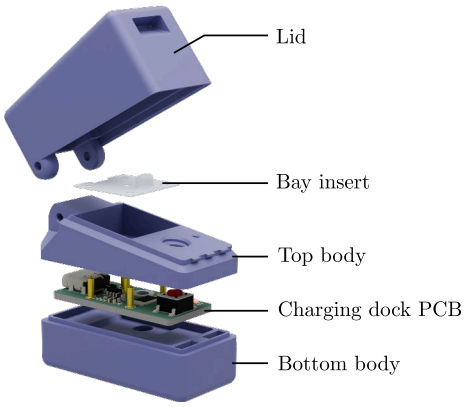


Figure 6.15: Exploded view of charging dock assembly.

The charging dock assembly (Figure 6.15) consists of the bottom and top body parts, the lid, and a bay insert. The bay insert is required to provide a flat surface and an aligning protrusion, since the top body part is with the bay facing towards the build plate and without supports. The top body part and the lid both have a pair of magnets pressed fitted at the ends to keep the charger closed and the dice pressed down on the pins.

6.2.3 Programming fixture

The programming adapter is necessary to program the dice board at least once before the firmware with OTA DFU capability is loaded in. The programming fixture is a simple breakout board connecting the needed dice pins (SWDCLK, SWDIO, RESET, GND, and VDD) via pogo pins to the J-Link compatible header. It also includes a connection to the DOCK_LED pin and one LED for debugging purposes. Power to the dice board can be delivered using J-Link or through the capacitor pads via an external power supply.

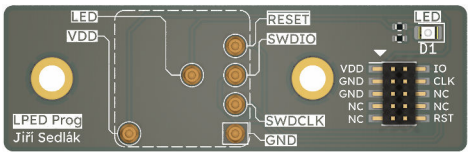


Figure 6.16: Rendered PCB programming fixture design from top view.

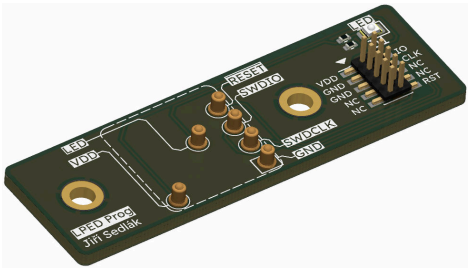


Figure 6.17: Rendered PCB programming fixture design from side view.

The adapter board (Figure 6.16 and Figure 6.17) features six mounting holes for pogo pins, a 1.27mm pitch 2x5 header, and two plated mounting holes for M3 screws. It has a 2-layer stack-up, with the top layer used for signal routing with ground fill and the bottom one being an uninterrupted ground plane. The debug LED is the same as the one used in the dock and dice designs.

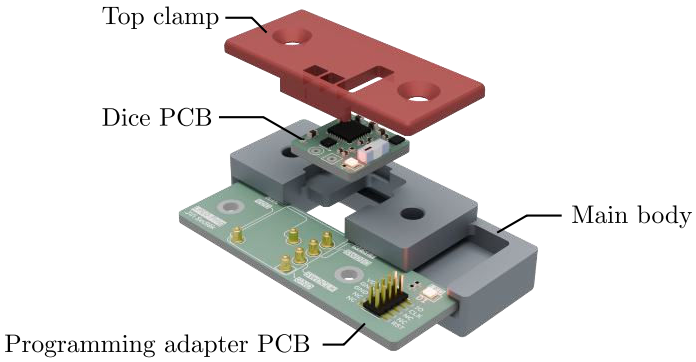


Figure 6.18: Exploded view of programming adapter.

The complete assembly (Figure 6.18) consists of the main body, where the PCB slots in from the side, and a top hard-mounted clamp, which presses the dice PCB down on the pogo pins. The top clamp was printed from thermo-reactive PLA filament that changes color from red to white when heated. This makes it easier to see potential issues, such as shorts or bad connections.

6.3 Mobile application

This section describes the design of the mobile application. How it is structured, what additional libraries were used, and what functions have been implemented.

The mobile application is written in Flutter and tested on Android. It uses Flutter Bluetooth Plus[2] for Bluetooth communication and connection management, Flutter Toast[3] for informational and error messages, and Flutter Spin Kit[4] for spinning loaders.

It aims to provide an easy-to-use graphical interface for dice customization and number display. It is composed of two main sections: **Play** and **Device management**.

6.3.1 Services

Functions for operating the persistent storage and Bluetooth are separated into two services: **Storage**, available in `lib/services/storage`, and **Bluetooth**, available in `lib/services/bluetooth`.

Storage

Storage service is used to store and retrieve dice information. Dice data are represented in device models. These models hold information about the last known capacitor state, the last message received during playing, the name of the device, the dice's Medium Access Control (MAC) address, and a history of landed numbers. The MAC address is later used for dice identification during the configuration and interpretation of received broadcast messages. Models are stored as `json` files for persistent storage. The history provides information about how many times certain sides landed on the dice.

[2] https://pub.dev/packages/flutter_blue_plus
[3] https://pub.dev/packages/fluttertoast
[4] https://pub.dev/packages/flutter_spinkit

Bluetooth

While communication and connection are handled by the Flutter Bluetooth Plus library, this service provides necessary constants, transform functions, and wrapper functions for easier reading and writing of specific characteristics.

The constants include positions of the GATT service and individual characteristics. Although the `RANGE` inside the dice definition can be any 16-bit number, three predefined values are provided for better user interaction. These are provided as a list of `SensitivityListModel`, which defines a name, which is a human-readable short description, and a value. Blinking modes are available in a similar fashion, as a list of four options represented as `LedModeListModel` with the same fields as the sensitivity model.

To easily translate status messages from the dice, several transform functions are available. These functions typically transform a list of integers into usable values, such as a capacitor state, message type, or message ID.

This service also provides functions for reading and writing specific GATT characteristics. The read functions return either a `null` if the reading failed or a value with the specific required type. Write functions return either a `true`, if the given value was written successfully, or `false` if the writing failed. Base generic read function returns a `GattResultRead`, which has a bool indicating if the read was successful or not, and the data itself as a list of integers that will be empty if an error occurred. The generic write function returns a `GattResultWrite`, which only contains the indication boolean value.

6.3.2 Sections

The application contains two main sections. The play section, which is represented by a single page called play, and the device management section, which is separated into two pages: devices and device settings.

Pages are then separated into sections and flows. Sections can be Widgets, such as custom cards or control components, used anywhere on the page. Flows are custom dialogs tailored to specific user interactions. Base classes for flows can be found in `app/lib/global` and they include base widgets for generic dialogs, simple confirmation dialogs, dialog action buttons, and multi-step dialogs. Flows always contain at least two files, one of which is the dialog itself, and the other one contains helper functions, including a function to display the dialog.

Play section

The functionality of the play section is defined in `app/lib/screens/play`. It displays the current state of all added dice (Figure 6.19). This state is derived from broadcast messages received from each dice, which are interpreted using the provided functions in the Bluetooth service. This status includes the current side facing up on each dice and its capacitor status. The capacitor status is only visible when its state of charge is below 15% in which case a red border will be displayed around the dice card.

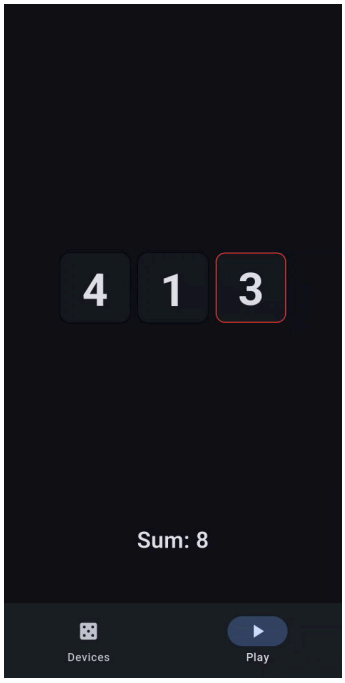


Figure 6.19: Play screen.

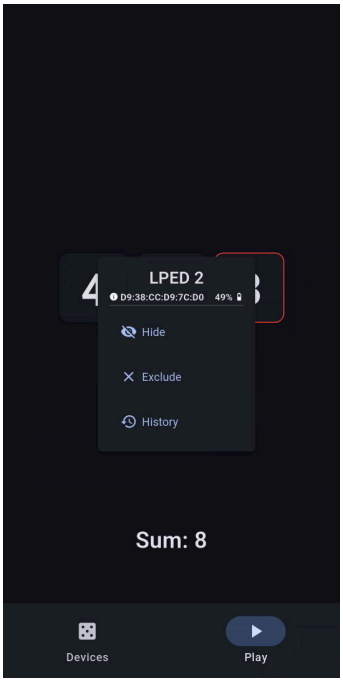


Figure 6.20: Play screen with dice dialog opened.

Apart from the dice status, this page also shows the current sum. The sum is calculated from the side numbers of all dice that are included and visible. Changing the dice inclusion or visibility can be done via a long-press on the dice card, which will bring up a context menu (Figure 6.20) with the dice name, MAC address, capacitor state percentage, and three buttons. If a dice is set as excluded, it will still be visible on the screen, but it will be greyed out and it will not be counted in the sum. When a dice is set to be invisible, it will not be shown on the screen at all, and it will not be counted in the sum. To make it visible again, the user has to tap on the devices tab and go back to the play tab, which will show all available devices. The history button will show a list of numbers and how many times they have been recorded by the application.

Device management section

The device management section allows user to search and add new dice and to change their settings. It is split into two pages: devices and device settings. These can be found in `app/screens/devices` and `app/screens/settings`.

The devices page allows users to search for new dice by clicking the action button that is located at the bottom center of the screen. This will show a dialog with a list of all discovered devices. Discovering devices is done by scanning for connectable advertising from devices with the name „LPED“. A found device can be added by clicking on its plus icon located on the right side.

When a device is added, it will be shown on the devices page as a device card, which displays how long ago the last status message was received, the last known capacitor state, its name, and MAC address, along with two buttons. The left button can be used to delete the device, and the right one will open the device's settings.

The device settings page will first try to find the correct device by scanning for connectable advertisements from devices with a specific MAC address that matches the requested dice MAC address. When the dice is found, the application will try to connect and load all data using the provided functions in the Bluetooth service. After all the data are read from the dice, the setting screen will be shown. It comprises a general and profile subpages. The general structure of these subpages is a title, which can be editable, general settings related to the subpage, a list of available actions, and a list of structures.

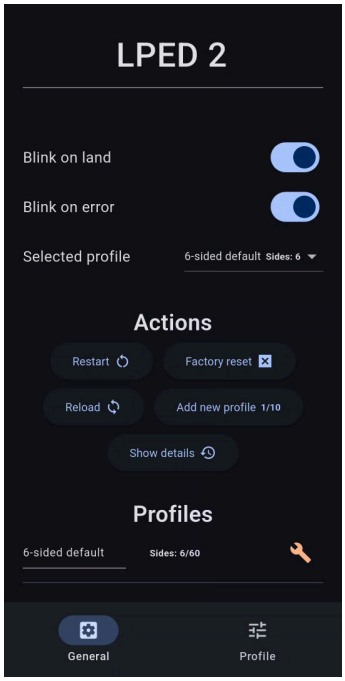


Figure 6.21: General dice settings page.

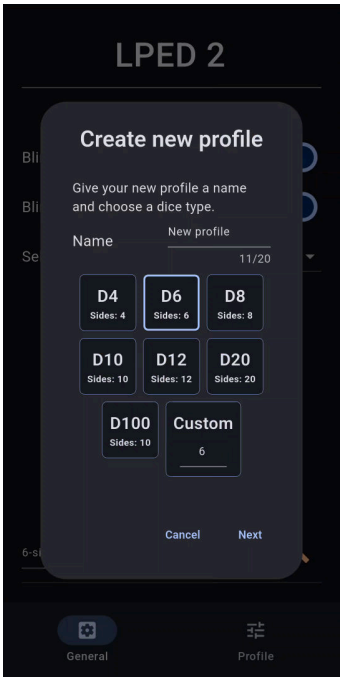


Figure 6.22: Dialog for adding a new dice profile in the type selection step.

The general subpage (Figure 6.21) can be used to change the dice name, blink on land, blink on error, and the currently used dice definition. Actions on this subpage are:

- **Reset**
  This action will initiate a cold reboot of the dice after user confirmation.
- **Reload**
  Reload will read all data from the dice and refresh the shown values.
- **Factory reset**
  This action will erase all user customization and user-defined profiles from the dice memory after user confirmation.
- **Add dice profile**
  This action will open a dialog (Figure 6.22) that will guide the user through all steps of adding a new dice definition. These steps include giving the dice profile a name, selecting the dice type or specifying the custom number of sides, and recording a vector for each side.

- **Show details**
  This action will open a dialog with numbers and their recorded counts, with the option to reset this log, alongside the device name and MAC address at the top of the dialog.

At the bottom of this subpage is a list of saved profiles on the dice. Each profile can be either deleted or edited by clicking the respective buttons. When a user clicks on the edit button, the profile will be selected as active, and the profile subsection will be automatically shown after the profile data is loaded.

The profile subsection (Figure 6.23) is used to change details of a selected dice definition. This includes the name and sensitivity. It also allows for the configuration of individual side numbers, blinking modes, and acceleration values. There is a single action on this subpage, which allows the addition of a new side. This is done by showing a dialog (Figure 6.24) that allows the user to either change the acceleration values manually or to capture them from the accelerometer, and to specify the side number and blink mode. The list of all sides in the given profile is shown below the actions. Each row of this list displays the side number, acceleration vector value for each axis, a drop-down with blink mode, and buttons for deletion and vector value change.

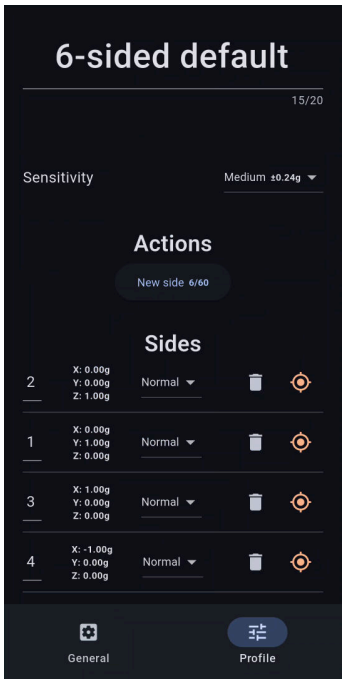


Figure 6.23: Dice profile settings page.

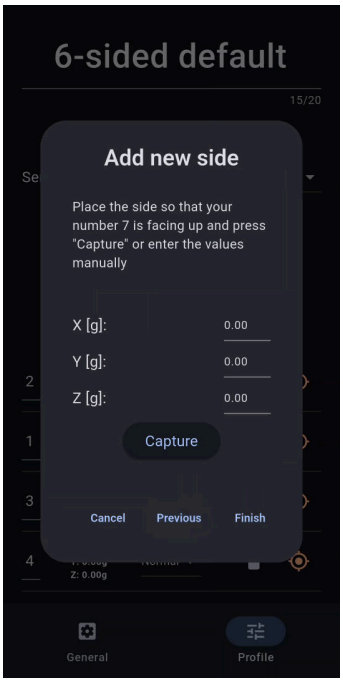


Figure 6.24: Dialog for adding a new side to a dice profile.

Chapter 7

Evaluation

This chapter will discuss how well the final implementation achieved the expected parameters, such as dice balance, battery life, and how much it costs to manufacture. The final version of the six-sided dice with an edge size of 17mm can be seen in Figure 7.1 and 7.2. The second figure also shows the charging dock.

Figure 7.1: Final version of the six-sided dice.

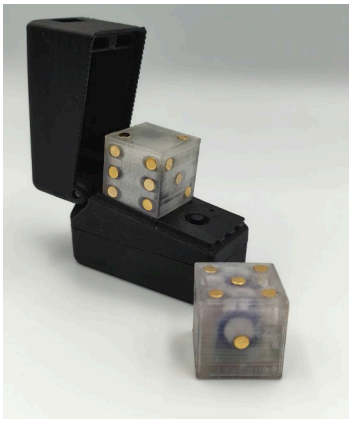


Figure 7.2: Final version of the charging dock with two finished dice.

7.1 Balance testing

The balance of the dice was tested with two different samples, both without any external sleeve. Each of them was thrown on a hard silicone mat at least a thousand times per face or six thousand times total per dice.

The results (Figure 7.3) have shown that the dice have a tendency to omit number two and are more likely to land on number five. This can be explained by the physical construction of the devices, where the PCB is placed on the side with number two, making it heavier.

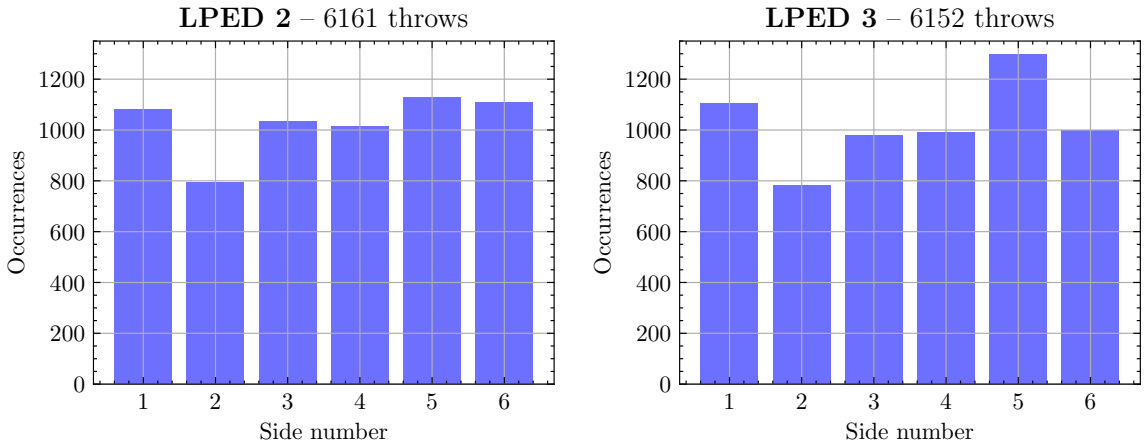


Figure 7.3: Balance test results for two dice.

To mitigate this, both of the dice were filled with epoxy resin to make them more homogenous inside. Then, the same testing method was used to check the dice balance, but due to time constraints, the number of samples per dice was reduced to four thousand. Results for LPED 3 are a bit skewed towards numbers one and three. This is because there was a malfunction during testing that caused the mobile application to register those two numbers multiple times, and without knowing the full extent, the results could not be corrected. The results can be seen in Figure 7.4. They show that by filling the dice, the balance has improved, as for LPED 2, the maximum difference is within 2.3% after the fill, while before it was 5.7%.

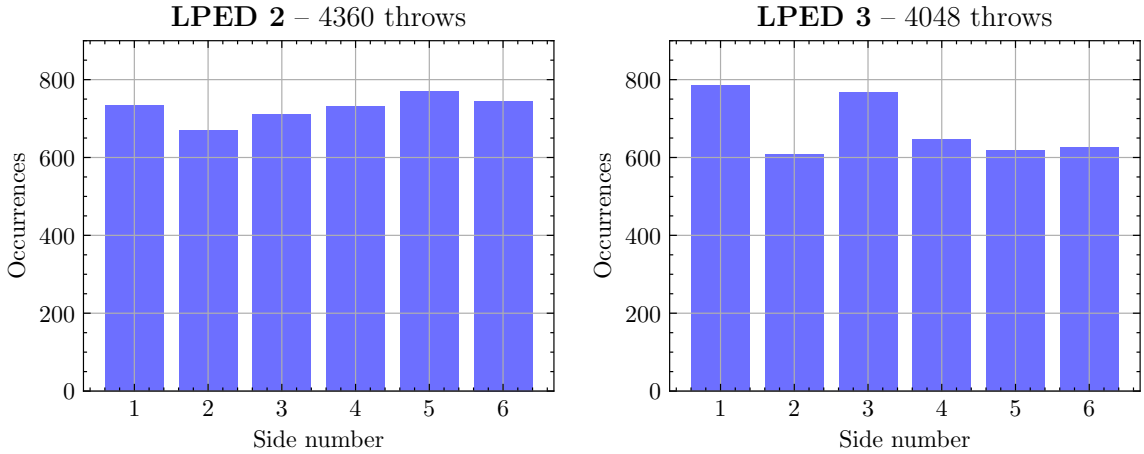


Figure 7.4: Balance test results for two dice after epoxy resin fill.

7.2 Battery life

The dice has an average current consumption of 10mA while transmitting, and 160uA when idle. The current consumption during a throw can be seen in Figure 7.5.

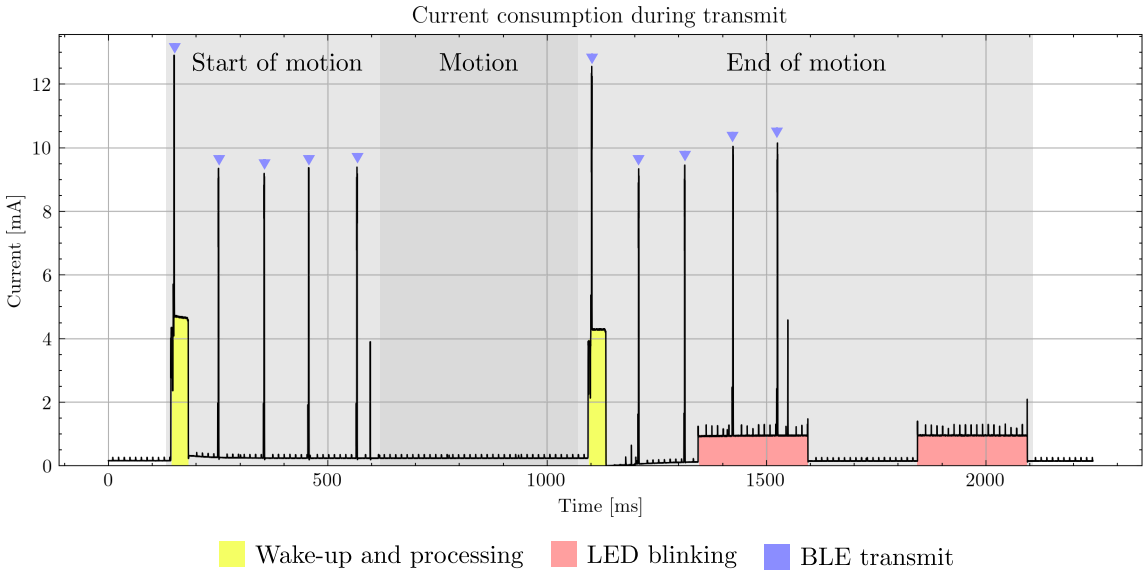


Figure 7.5: Current consumption during a throw.

During a realistic scenario, it reported a battery state between 65% and 75% after four hours of use, suggesting that the battery life of the dice is at least eight hours. Note that the capacitor was charged to around 85% at the start.

The balance test took approximately four hours and forty minutes, and it represents the most extreme usage scenario, where the dice is thrown immediately after the number transmission. Even in this use case, the dice reported capacitor states of 54% and 41%, further confirming the suggested battery life.

Compared to the existing solutions, the full projected battery life is four times longer than the one claimed by GoDice and eight times longer than the battery life of eDice. Even with all LEDs turned off, the eDice has a battery life of ten thousand throws, while the designed dice is projected to have twelve thousand throws with the most demanding scenario and LED mode set to normal for all sides.

7.3 Price

The price for each object is broken down into individual categories. All prices are listed for the minimum required order, and they include VAT. PCBs were manufactured by Gatema using their POOL service with immersion gold plating.

Even with the prototype pricing, which is often much higher than the prices for high-volume runs, the final solution (dice with a charging dock) managed to be two times cheaper than the mentioned eDice and approximately 80 CZK cheaper than a single GoDice with a USB charger.

However, it is important to note that the calculated prices for this project don't include assembly and labour costs that could be significant with regard to the high time complexity of the assembly. They also don't include profit margins.

Playing dice – total cost: 404.07 CZK

The price of the dice could be lowered by not using pogo-pin pads on all sides of the dice. This would decrease the price by 44.08 CZK.

- Electrical components: 275.78 CZK
- Manufactured PCB: 28.8 CZK
- 3D printed case: 0.71 CZK
- Pogo-pin pads: 57.86 CZK
- Connection wires: 0.15 CZK
- Super-capacitor: 40.77 CZK

Charging dock – total cost: 233.53 CZK

- Electrical components: 100.75 CZK
- Manufactured PCB: 50.41 CZK
- 3D printed case: 5.53 CZK
- Pogo-pins: 62.02 CZK
- Magnets: 12 CZK
- Heat insert: 1.32 CZK
- Screw: 1.5 CZK

Programming adapter – total cost: 173.12 CZK

- Electrical components: 47.09 CZK
- Manufactured PCB: 64.08 CZK
- 3D printed case: 2.19 CZK
- Pogo-pins: 57.75 CZK
- Screws: 2.01 CZK

7.4 User feedback

Three fully assembled dice, together with one functional charging dock and a poster (Appendix B), were presented at Excel@FIT. Bystanders were free to try to throw the dice, ask any questions, and provide feedback. The most asked questions were:

1. Is the dice well balanced?
2. How is the dice charged?

3. How long does it last on one charge?
4. Does it support different dice shapes?
5. How much does it cost?

Users had positive responses to the provided details, and the dice received overall approving comments.

Chapter 8

Conclusion

The goal of this project was to design and implement low-power electronic playing dice that communicates with a mobile application via Bluetooth Low Energy. The final implementation provides a complete solution for a playing dice that is ready for real-world use. This solution allows users to configure and use any dice shape, and it provides a practical way of charging the dice with the charging dock. The physical realization achieved its size goal, as the dice has an edge size of 17mm. The designed mobile application provides a clear and solid user interface for not only showing what numbers landed on the dice, but also to change the dice configuration, including defining completely custom dice profiles. This, together with the proposed dice sleeves, offers excellent support for multi-sided or niche game-specific dice. Additionally, thanks to the efficient firmware design and the choice of super-capacitor as the power source, the dice can provide up to eight hours of battery life with a fifteen-minute charging time.

Possible future extensions include improving the charging dock. One way would be to implement a battery-powered charging dock. This would improve flexibility for the more on-the-go users, as the charger would be equipped with its own battery, instead of requiring a separate power bank.

Other improvements could involve shrinking the dice module size by using machine assembly, where it is possible to use components with a BGA package or similar. This would allow, other than making the dice smaller, to fit other peripherals inside the dice, such as a buzzer or more LEDs, It would also make it possible to use a MCU with larger memory, which would allow for adding the OTA DFU capabilities and adding more functionality in the future.

There are also suggestions for enhancement coming from the visitors at Excel@FIT or users who had the opportunity to try the dice out. These suggestions included more flexible side values, which would allow the sides to support other symbols or colors instead of just numbers. This would improve the experience with specific games.

The mobile application could implement a connection to some game APIs, like the ones mentioned in Chapter 4 under GoDice.

Bibliography

[1] *TDK IMU Devices PCB Board Design Guidelines*. TDK Invensense, march 2022. Available at: https://invensense.tdk.com/wp-content/uploads/2022/07/AN-000262-TDK-IMU-Devices-PCB-Board-Design-Guidelines_v1.3.pdf. Revision 1.3.

[2] *2.4 GHz SMD, Above Metal, Low Profile Mini Chip Antenna*. 2450AT42E010B. Johanson Technology, Inc., october 2021. Available at: https://www.johansontechnology.com/docs/1080/2450AT42E010B_C3Pvfnl.pdf. Ver. 5.1.

[3] *Micropower, 3-Axis, ±2g/±4g/±8g Digital Output MEMS Accelerometer*. ADXL367. Analog Devices Inc., october 2024. Available at: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl367.pdf. Rev. B.

[4] *4.0V Low Leakage Lithium Hybrid Supercapacitors*. AHCR-S04R0S. Abracon LLC, september 2024. Available at: https://abracon.com/datasheets/AHCR-S04R0S.pdf. REVISION A.

[5] Ahmad, N.; Raja Ghazilla, R. A.; Khairi, N. and Kasi, V. Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications. *International Journal of Signal Processing Systems*, january 2013, vol. 1, p. 256–262. Available at: https://doi.org/10.12720/ijsps.1.2.256-262.

[6] *2.0 x 1.25 mm SMD Chip LED Lamp*. APTD2012LSURCK. Kingbright, march 2021. Available at: https://www.kingbrightusa.com/images/catalog/SPEC/APTD2012LSURCK.pdf. Rev. no V.5A.

[7] Armenise, M.; Ciminelli, C.; De Leonardis, F.; Diana, R.; Passaro, V. et al. Gyroscope technologies for space applications. *Optoelectronics Laboratory, Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari*, 2003, vol. 6. Available at: https://escies.org/download/webDocumentFile?id=1056.

[8] Ballas, R. The Piezoelectric Effect – an Indispensable Solid State Effect for Contemporary Actuator and Sensor Technologies. *Journal of Physics: Conference Series*, january 2021, vol. 1775, p. 012012. Available at: https://www.doi.org/10.1088/1742-6596/1775/1/012012.

[9] Bhalla, N. *Simulations of MEMS based Piezoresistive Accelerometer Design in COMSOL*. Institute of Electronic Engineering, Chung Yuan Christian University-Taiwan, 2 Institute of NEMS, National Tsing Hua University, Taiwan, october 2011. Available at: https://www.comsol.com/paper/download/100723/bhalla_paper.pdf.

[10] Binali, R.; Demirpolat, H.; Kuntoğlu, M.; Makhesana, M.; Yaghoubi, S. et al. A Comprehensive Review on Low-Cost MEMS Accelerometers for Vibration Measurement: Types, Novel Designs, Performance Evaluation, and Applications. *Journal of Molecular and Engineering Materials*, march 2024, vol. 12. Available at: https://doi.org/10.1142/S225123732430002X.

[11] Bluetooth Special Interest Group. *Bluetooth Low Energy – Regulatory Aspects Document (RAD)* https://www.bluetooth.com/wp-content/uploads/2023/03/bluetooth-le-regulatory-aspects-document.pdf. 1.01. 2023. Available at: https://www.bluetooth.com/wp-content/uploads/2023/03/bluetooth-le-regulatory-aspects-document.pdf. [cit. 2025-05-07]. Prepared by the Regulatory Expert Group.

[12] Bluetooth Special Interest Group. *The Bluetooth® Low Energy Primer* https://www.bluetooth.com/wp-content/uploads/2022/05/the-bluetooth-le-primer-v1.2.0.pdf. 1.2.0. March 2024. Available at: https://www.bluetooth.com/wp-content/uploads/2022/05/the-bluetooth-le-primer-v1.2.0.pdf. Prepared by Ifti Anees.

[13] *3-axes ultra-low power accelerometer*. BMA400. Bosh Sensortec, february 2023. Available at: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bma400-ds000.pdf. Rev. No 2.0.

[14] *Using Decoupling Capacitors*. Cypress Semiconductor Corporation, march 1999. Available at: https://hsi.web.cern.ch/s-link/devices/g-ldc/decouple.pdf.

[15] Dean, R. *Introduction to Microfabriaction*. Electrical & Computer Engineering Department, Auburn University. Available at: https://www.eng.auburn.edu/~deanron/Lecture_81922.pdf. Last visited: 7. 5. 2025.

[16] Eric Beljaars, A. M. USB Type-C® and USB power delivery common use cases and block diagrams. In: *An Engineering's Guide To USB Type-C®*. Texas Instruments Incorporated, 2024, p. 54. Available at: https://www.ti.com/lit/eb/slyy228/slyy228.pdf.

[17] *Ultra-Low-Power SoC with RISC-V Single-Core CPU*. ESP32-C3. ESP32-C3, november 2024. Available at: https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf. Version v2.0.

[18] *Ultra-low-power SoC with RISC-V single-core microprocessor*. ESP32-C6. ESP32-C6, august 2024. Available at: https://www.espressif.com/sites/default/files/documentation/esp32-c6_datasheet_en.pdf. Version v1.2.

[19] Franssila, S. Introduction. In: *Introduction to Microfabrication*. John Wiley & Sons, Ltd, 2010, chap. 1, p. 1–13. ISBN 9781119990413. Available at: https://doi.org/10.1002/9781119990413.ch1.

[20] *3-Axis Low-g Accelerometer*. FXLS8967AF. NXP Semiconductor, march 2022. Available at: https://www.nxp.com/docs/en/data-sheet/FXLS8967AF.pdf. Rev. 1.5.

[21] Ge, C. and Cretu, E. A polymeric piezoelectric MEMS accelerometer with high sensitivity, low noise density, and an innovative manufacturing approach. *Microsystems & Nanoengineering*, Nov 2023, vol. 9, no. 1, p. 151. ISSN 2055-7434. Available at: https://doi.org/10.1038/s41378-023-00628-7.

[22] Hrvoic, I. Magnetometers. In: Gupta, H. K., ed. *Encyclopedia of Solid Earth Geophysics*. Dordrecht: Springer Netherlands, 2011, p. 810–816. ISBN 978-90-481-8702-7. Available at: https://doi.org/10.1007/978-90-481-8702-7_122.

[23] *High Performance 3-Axis Accelerometer*. ICM-42370-P. TDK InvenSense, july 2023. Available at: https://invensense.tdk.com/wp-content/uploads/2023/12/DS-000546-ICM-42370-P-v1.0.pdf. Revision 1.0.

[24] IEEE Spectrum. *Chip Hall of Fame: Texas Instruments Digital Micromirror Device* https://spectrum.ieee.org/chip-hall-of-fame-texas-instruments-digital-micromirror-device. June 2017. [cit. 2025-05-08]. [Photograph (author: Larry Hornbeck)].

[25] *± 2g/4g/8g/16g Tri-axis Digital Accelerometer*. KX132-1211. Kionix Inc., november 2021. Available at: https://fscdn.rohm.com/kionix/en/datasheet/kx132-1211-e.pdf. Revision 4.0.

[26] *High-performance ultralow-power 3-axis femto accelerometer*. LIS2DW12. STMicroelectronics, august 2024. Available at: https://www.st.com/resource/en/datasheet/lis2dw12.pdf. Revision 9.

[27] *Ultra-low power, low-noise, integrated digital output 3-axis accelerometer*. MC3635. Memsic Inc., march 2023. Available at: https://www.memsic.com/Public/Uploads/uploadfile/files/20220119/MC3635Datasheet.pdf. Revision APS-048-0044v1.10.

[28] Mohanty, T. Design and Analysis of MEMS-Based Piezoresistive Accelerometer with Low Cross-Axis Sensitivity. *IOSR Journal of Engineering*, april 2013, vol. 03, p. 54–59. Available at: https://doi.org/10.9790/3021-03415459.

[29] *A Bluetooth® 5.2 SoC supporting Bluetooth Low Energy and 2.4 GHz protocols, optimized for small, cost-effective two-layer PCB designs.* nRF52805. Nordic Semiconductor, december 2023. Available at: https://docs.nordicsemi.com/bundle/nRF52805_PS_v1.4/resource/nRF52805_PS_v1.4.pdf. Version 1.4.

[30] *A 2.4 GHz wireless SoC integrating the nRF52 Series transceiver, an Arm® Cortex®-M4 CPU, and supporting various protocols.* nRF52810. Nordic Semiconductor, december 2023. Available at: https://docs.nordicsemi.com/bundle/nRF52810_PS_v1.5/resource/nRF52810_PS_v1.5.pdf. Version 1.5.

[31] *A 2.4 GHz wireless SoC integrating the nRF52 Series transceiver, an Arm® Cortex®-M4 CPU, and supporting various protocols.* nRF52832. Nordic Semiconductor, december 2023. Available at: https://docs.nordicsemi.com/bundle/nRF52832_PS_v1.9/resource/nRF52832_PS_v1.9.pdf. Version 1.9.

[32] *A SoC with a Bluetooth® 5.1 direction finding radio operates at -40°C to 105°C, supporting various protocols.* nRF52833. Nordic Semiconductor, june 2024. Available at:

https://docs-be.nordicsemi.com/bundle/ps_nrf52833/attach/nRF52833_PS_v1.7.pdf. Version 1.7.

[33] *A 2.4 GHz wireless SoC integrating a multiprotocol 2.4 GHz transceiver and an Arm® Cortex®-M4F CPU for short range networks.* nRF52840. Nordic Semiconductor, october 2024. Available at: https://docs-be.nordicsemi.com/bundle/ps_nrf52840/attach/nRF52840_PS_v1.11.pdf. Version 1.11.

[34] *Dual-core Bluetooth 5.4 SoC supporting Bluetooth LE, Bluetooth mesh, NFC, Thread and Zigbee.* nRF5340. Nordic Semiconductor, june 2024. Available at: https://docs-be.nordicsemi.com/bundle/ps_nrf52840/attach/nRF5340_PS_v1.5.pdf. Version 1.5.

[35] Prouty, M.; Johnson, R.; Hrvoic, I. and Vershovskii, A. Geophysical applications. *Optical Magnetometry*, january 2011, p. 319–336. Available at: https://doi.org/10.1017/CBO9780511846380.018.

[36] Rao, R. *An Introduction to MEMS (Micro-electromechanical Systems).* PRIME Faraday Partnership, 2002. Available at: https://web.stanford.edu/class/cs114/readings/KD-Prime.pdf.

[37] Rashid, W. *Piezoelectric Based MEMS Accelerometer Design and Optimization.* 2021. Master's thesis. Tallinn Univerity of Technology. Available at: https://digikogu.taltech.ee/et/Download/33084236-9864-412a-8253-fcf2d96308a9.

[38] Sandia National Laboratories. *MEMS Video & Image Gallery* https://www.sandia.gov/mesa/mems-video-image-gallery/. N.d. [cit. 2025-05-08]. Courtesy Sandia National Laboratories, SUMMiT™ Technologies, www.sandia.gov/mstc.

[39] *Multiprotocol wireless 32-bit MCU Arm® -based Cortex® -M4 with FPU, Bluetooth® 5.4 radio solution.* STM32WB10CC. STMicroelectronics, november 2023. Available at: https://www.st.com/resource/en/datasheet/stm32wb10cc.pdf. Revision 7.

[40] *Multiprotocol wireless 32-bit MCU Arm® -based Cortex® -M33 with TrustZone® , FPU, Bluetooth ® 5.4 and IEEE 802.15.4 radio solution.* STM32WBA52. STMicroelectronics, february 2024. Available at: https://www.st.com/resource/en/datasheet/stm32wba52cg.pdf. Revision 5.

[41] *Synchronous Step-Down Converter With Advanced Eco-Mode™.* TPS560200. Texas Instruments, february 2015. Available at: https://www.ti.com/lit/ds/symlink/tps560200.pdf. Revision B.

[42] Umoru, S. E. Hybrid Supercapacitor For Energy Storage Devices: A Review. *Journal of Physics and Chemistry of Materials*, 2023, vol. 10. Available at: https://www.isroset.org/pub_paper/JPCM/4-ISROSET-JPCM-09160.pdf.

[43] Veena, S.; Rai, N.; Suresh, H. L. and Nagaraja, V. S. Design, Modelling, and Simulation analysis of a Single Axis MEMS-based Capacitive Accelerometer. *International Journal of Engineering Trends and Technology.* Seventh Sense Research Group Journals, 2021, vol. 69, no. 10. ISSN 2231-5381. Available at: http://dx.doi.org/10.14445/22315381/IJETT-V69I10P211.

[44] Zadehdarrehshoorian, M. *BLE ADVERTISING CHANNEL ANALYSIS*. 2023. Master's thesis. Tampere University. Available at: https://trepo.tuni.fi/bitstream/handle/10024/147694/ZadehdarrehshoorianMaral.pdf.

Appendix A

Dice assembly guide

The assembly process can be challenging due to the small diameters of the used wires and the hard-to-hold-down PCB. This guide provides the most consistent and easiest way to assemble the dice. It takes between thirty and sixty minutes, and it is recommended to prepare a pair of sharp tweezers, a silicone mat, tweezers with plastic tips or similar, and either a small pliers or flat-ended tweezers for the glueing. For the assembly of the dice, several parts are necessary:

1. Case body (3D printed preferably from translucent PETG filament
2. Case cap (3D printed preferably from translucent PETG filament
3. Assembled dice PCB
4. Selected super-capacitor
5. Enameled wire with a diameter of at least 0.1mm (For LED, Button, and Connection pads)
6. Enameled wire with a diameter of at least 0.25mm (For 3V3 and Ground pads)
7. Pogo-pin pads with a diameter of 3mm and a height of 0.3mm
8. Superglue
9. Clear electronic-safe resin (Optional)

The assembly process starts with the connection wire preparation. Cut three sections from the 0.1mm wire and two sections from the 0.25mm wire with a length between two and three centimeters (prefer longer sections for easier soldering later). Strip approximately 2mm of insulation from both ends of the section and tin them.

In the next step, the connection pads are prepared. Solder one end of the section to the pogo-pin pad. This can be done by heating the pad, adding a bit of solder, then placing the wire into the solder and slowly rotating the soldering iron tip while pushing down and moving away from the pad. This should leave most of the solder on the pad. A warning: the pad remains heated for a long time. After all sections have a pad soldered to one side, the next step is to glue the pad to the case cap. The most consistent method is to fit the section through one of the holes in the cap, making sure that the pad can sit flush with the cap surface, rotating the cap so that the pad is facing down on the working surface, and then, aligning the pad with the hole and recess, pushing down on the cap, and adding

a small amount of the super glue to the hole with the wire. Note that the two wires with bigger diameters have to be placed in specific locations, one of them has to be placed in the center, and the location of the other one depends on the selected position of the aligning charging dock key. You can optionally glue the pogo-pin pads to all recesses of the case body.

The next step is soldering the super-capacitor to the board. It should be soldered on the back side (side with the connection pads). The connection lead of the super-cap should be first bent to the side and then down, providing more flexibility and room when positioning it inside the case. Solder the capacitor so that it lies approximately 1mm from the board. After that, bend it to the opposite side to make room for soldering the connection wires.

After soldering the capacitor, the connection wires must be soldered to the corresponding pads on the PCB. Be careful not to leave any spikes of solder on the pads to avoid damaging the super-capacitor. After that, bend the capacitor back to lie on the PCB.

The last required step is to place the soldered assembly in the case body. Ensure that the PCB doesn't have any remaining protrusions from mouse bites. Otherwise, it might not fit well into the body. Carefully push the PCB down, preferably with the plastic tweezers, and check against the light that it lies flat on the bottom of the body.

If you want to fill the dice with resin, now would be the time. Fill the dice with the resin and wait at least a minute for it to fill the small cavities. Then, top it off again, and while managing the connection wires, place the cap on top of the body, and push it down until it clicks into place. Then roll the dice around in your hands. You should see an air bubble rise from the side with the PCB, which means that the resin successfully filled every side. Now open the lid again and add more resin as needed.

The final step is to put the cap on the body, while making sure that the connection wires don't get clamped on the side. Push it down until it clicks.

Appendix B

Excel@FIT poster

This appendix contains the poster design that was presented at Excel@FIT.

Low-power electronic dice

Author: **Sedlák Jiří** 2024/25 Supervisor: **doc. Ing. Zdeněk Vašíček, Ph.D.**

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025

Moving beyond proof-of-concept

A fully operational alpha prototype with custom hardware, firware, and enclosure. No dev kits.

Ultra-low power consumption

1.75uA while idle
7.12mA while constantly transmitting

Very small module size

Compact PCB (14x14mm), fits dice-sized designs from 17mm per side.

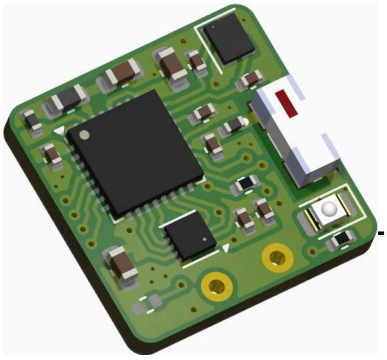


Figure 1: Playing dice PCB

Low-power wireless communication

Utilizes Bluetooth Low Energy with an energy-optimized communication scheme.

Battery-less passive solution

Hybrid super-capacitor that charges quickly and lasts long.

Customizable dice shape

Sleeves can change the look or number of sides the dice has.

Firmware made with Zephyr

Including custom accelerometer libary.

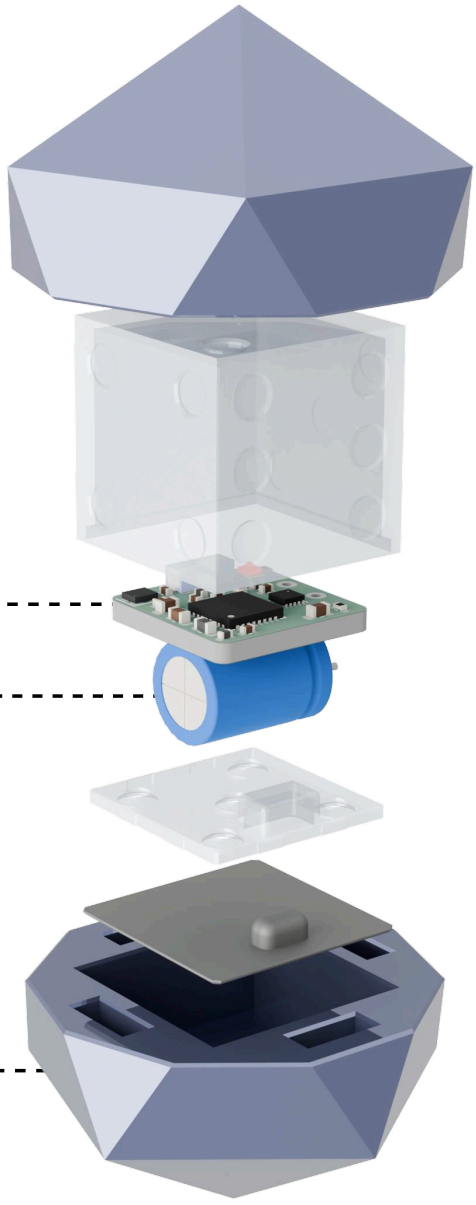


Figure 5: Playing dice assembly with 20-sided sleeve

Accesories

Charging dock

With compact form-factor that connects to the dice via pogo-pins and presses the dice down with lid to keep the connection secure.

Easy to 3D print and customize

No supports required, only heat insert and magnets are necessary.

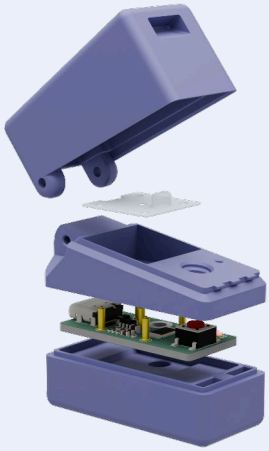


Figure 2: Assembly of charging dock

Programming adapter

Support for development with pogo-pin connected programmer.

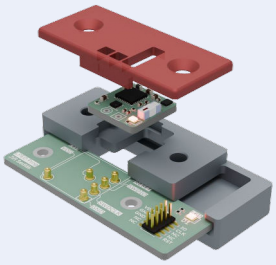


Figure 6: Programming adapter assembly

Companion application

Extensive dice configuration

Parameters can be customized to users liking, including blinking on landing, error blinking, name and currently selected profile.

Support for multiple profiles

Each dice can hold up to 10 dice profiles, that defines the sensitivity and sides.

Up to 60 sides per profile

Side profile tells the dice what number corresponds to physical position of the dice.

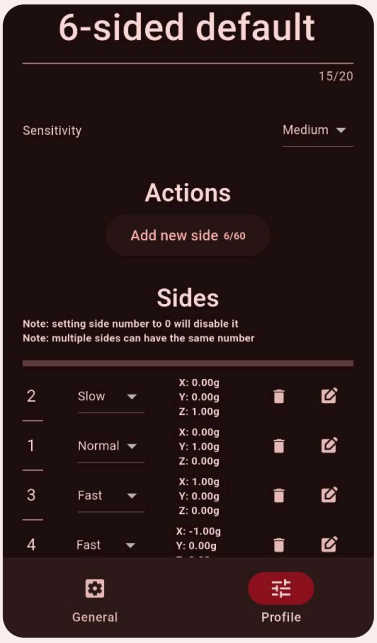


Figure 3: Settings of a dice profile

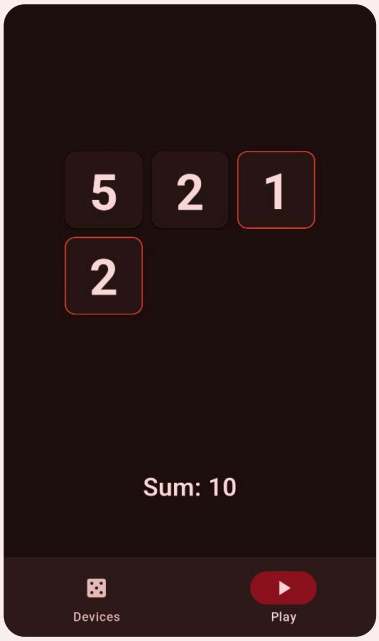


Figure 4: Playing screen

Real-time dice updates

The app will let the user know when a dice is low on power, when it starts moving and when it lands on a number.

Responsive sum

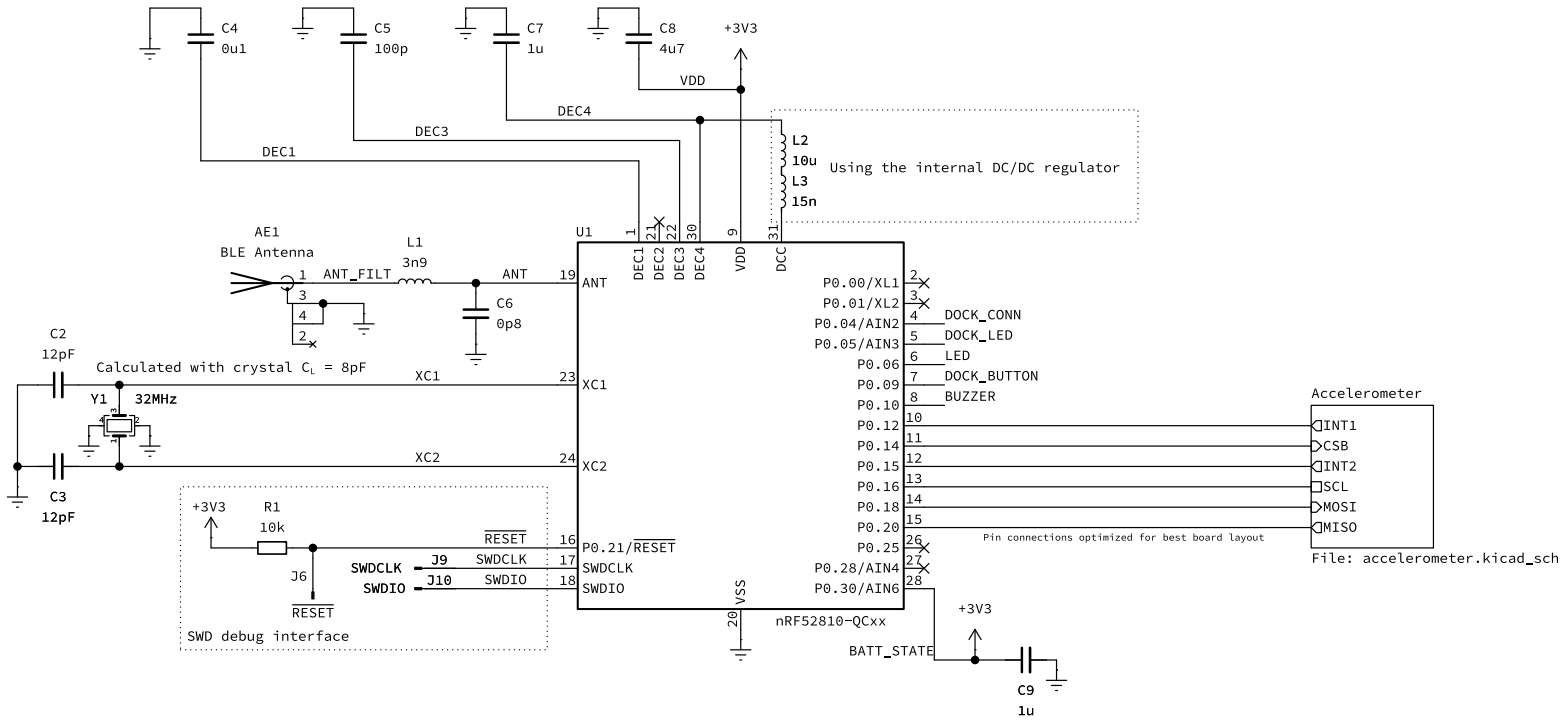
The on-screen sum automatically updates as dice send in numbers. Any dice can be exluded from the sum or hidden completely.

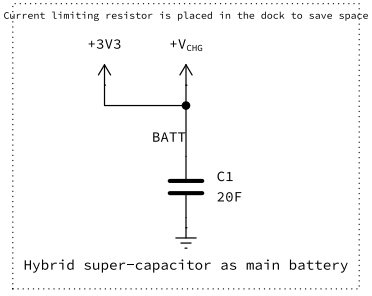
Mobile application written in Flutter

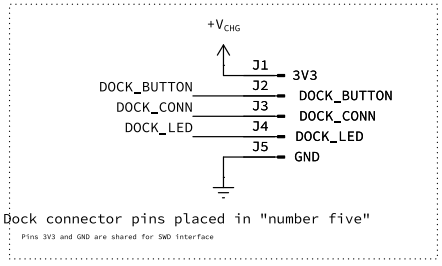
Appendix C

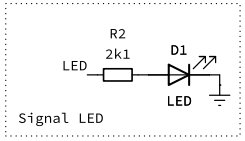
Schematics

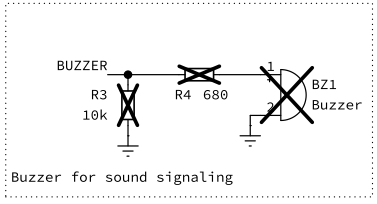
This appendix contains full schematics for all three PCBs. Schematics are listed immediately after each other in this order: dice, charging dock, and programming adapter. The schematic for the dice is separated into two sheets, the first one contains circuitry for the MCU section and the second one for the accelerometer section. The charging dock schematic is separated into five sheets. The first one depicts the block wiring diagram, the second one contains the USB section, the third one provides schematics for the charging voltage regulator, the fourth contains the pogo-pin to dice connections, and the last one shows circuitry for the on-board LED and button. The programming adapter schematic is only one sheet, which comprises the adapter header wiring and the debug LED circuitry.









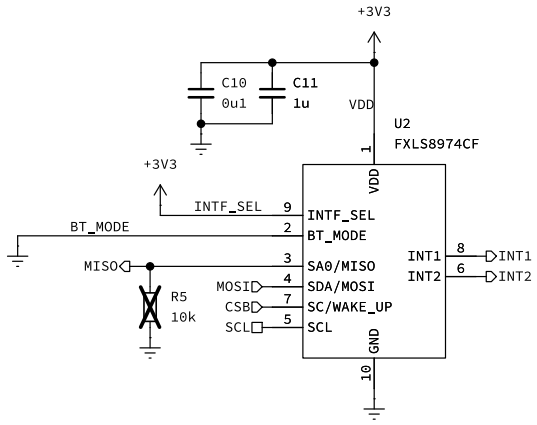


Jiří Sedlák (xsedla2e)

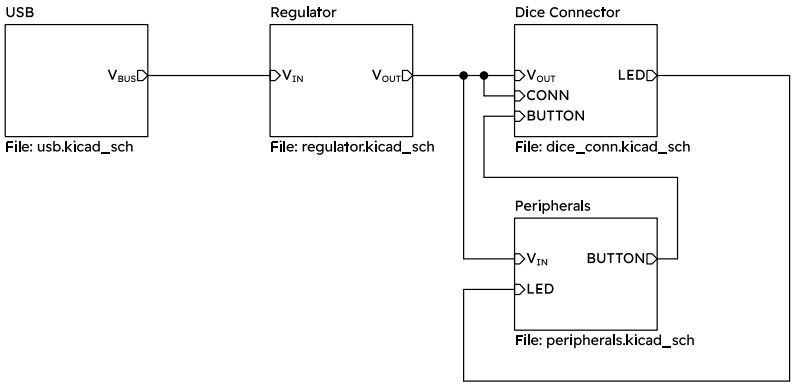
Sheet: /
File: lped.kicad_sch

Title: Low power electronic dice (LPED)

Size: A4	Date: 2025-02-24	**Rev: 1**
KiCad E.D.A. 9.0.1		Id: 1/2



Sheet: /Accelerometer/ File: accelerometer.kicad_sch		
Title:		
Size: A4	Date: 2025-02-24	**Rev:**
KiCad E.D.A. 9.0.1		Id: 2/2



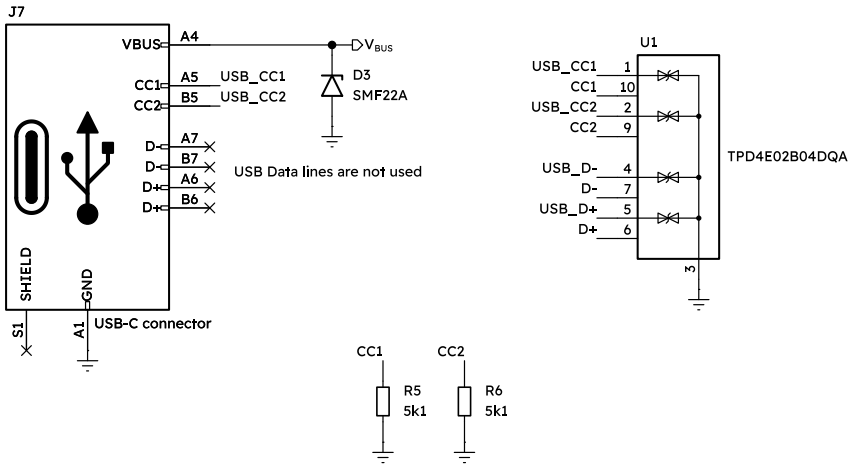
H1
M3

Jiří Sedlák (xsedla2e)

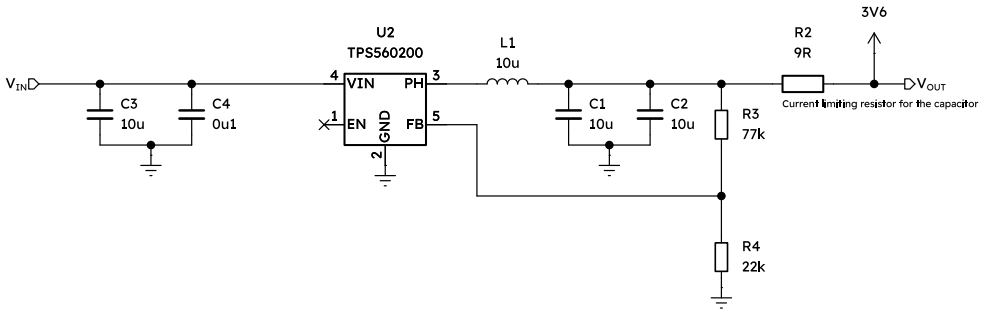
Sheet: /
File: lped_dock.kicad_sch

Title: Charging dock for LPED

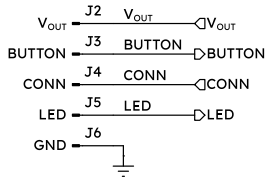
Size: A4	Date: 2024-12-20	**Rev: 1**
KiCad E.D.A. 8.0.7		Id: 1/5



Jiří Sedlák (xsedla2e)		
Sheet: /USB/ File: usb.kicad_sch		
Title: Charging dock for LPED		
Size: A4	Date: 2024-12-20	**Rev: 1**
KiCad E.D.A. 8.0.7		Id: 3/5



Jiří Sedlák (xsedla2e)		
Sheet: /Regulator/ File: regulator.kicad_sch		
Title: Charging dock for LPED		
Size: A4	Date: 2024-12-20	**Rev: 1**
KiCad E.D.A. 8.0.7		Id: 3/5

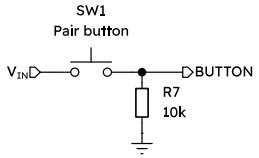


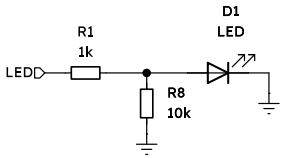
Jiří Sedlák (xsedla2e)

Sheet: /Dice Connector/
File: dice_conn.kicad_sch

Title: Charging dock for LPED

Size: A4	Date: 2024-12-20	**Rev: 1**
KiCad E.D.A. 8.0.7		Id: 4/5



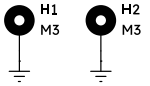


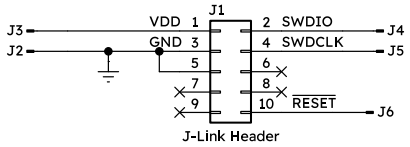
Jiří Sedlák (xsedla2e)

Sheet: /Peripherals/
File: peripherals.kicad_sch

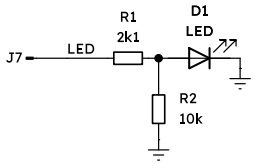
Title: Charging dock for LPED

Size: A4	Date: 2024-12-20	**Rev: 1**
KiCad E.D.A. 8.0.7		Id: 5/5





J-Link Header



Jiří Sedlák (xsedla2e)

Sheet: /
File: lped_prog.kicad_sch

Title: Programming fixture for LPED

Size: A4	Date: 2024-12-19	**Rev: 1**
KiCad E.D.A. 8.0.7		Id: 1/1