VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

SYSTÉM PRO REGISTRACI SPOTŘEBOVANÉ ELEKTRICKÉ ENERGIE

SYSTEM FOR ELECTRICITY CONSUMPTION REGISTRATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MATEJ HOCKICKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VOJTĚCH MRÁZEK, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

156898

Ústav:	Ústav počítačových systémů (UPSY)
Student:	**Hockicko Matej, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Vývoj aplikací
Název:	**Systém pro registraci spotřebované elektrické energie**
Kategorie:	Vestavěné systémy
Akademický rok:	2023/24

Zadání:

1. Seznamte se s technologií Bluetooth Low Energy (BLE) a jejím využitím pro potřeby získávání naměřených dat z vestavěných zařízení. Proveďte rešerši volně dostupných frameworků pro tvorbu multiplatformních mobilních aplikací (např. Apache Cordova, Flutter, React Native apod.). Zaměřte se na dostupnost a úroveň podpory technologie BLE.
2. Navrhněte systém, který bude sestávat z vestavěného zařízení, které bude měřit spotřebovanou elektrickou energii na základě monitorování optického výstupu elektroměru, a mobilní aplikaci umožňující konfiguraci tohoto zařízení, stahování naměřených dat a prohlížení historických dat. Při návrhu se snažte o maximalizaci životnosti baterie tohoto zařízení. Pro datové úložiště využijte kombinaci cloudové služby a lokální cache.
3. Navržený systém implementujte na zvolené platformě. Vestavěné zařízení lze implementovat formou prototypu, např. s využitím existujících modulů a vývojových desek.
4. Ověřte funkčnost navrženého řešení. Vyhodnoťte a diskutujte jeho parametry a možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodu 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Mrázek Vojtěch, Ing., Ph.D.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2023
Termín pro odevzdání:	17.5.2024
Datum schválení:	2.5.2024

Abstrakt

Cieľom tejto práce je navrhnúť systém na zaznamenávanie spotrebovanej elektrickej energie. Tento systém sa bude skladať zo vstavaného zariadenia, ktoré monitoruje optický výstup elektromeru a mobilnej aplikácie, s využitím ktorej bude možné za pomoci technológie Bluetooth sťahovať a zobrazovať namerané údaje. Vstavaný systém sa bude skladať z mikrokontroléru od firmy Nordic Semiconductor. Mobilná aplikácia bude naprogramovaná vo frameworku Flutter. Historicky namerané údaje sa budú ukladať na cloud. Dôraz pri návrhu bude kladený na výdrž vstavaného zariadenia, ktoré bude napájané z batérie CR2032. Hlavným prínosom tejto práce je možnosť monitorovať elektrickú spotrebu za pomoci vstavaného zariadenia a mobilnej aplikácie bez nutnosti vlastnenia smart elektromeru.

Abstract

The aim of this work is to design a system to record the consumed electricity. This system will consist of an embedded device that monitors the optical output of the meter and a mobile application, which will be able to download and display the measured data using bluetooth technology. The embedded system will consist of a microcontroller from Nordic Semiconductor. The mobile application will be programmed in the Flutter framework. Historically measured data will be stored on the cloud. The focus of the design will be on the battery life of the embedded device, which will be powered by a CR2032 battery. The main benefit of this work is the ability to monitor electricity consumption using the embedded device and mobile application without the need of owning a smart meter.

Kľúčové slová

mikrokontrolér, Nordic Semiconductor, Bluetooth, BLE, elektromer, spotreba elektrickej energie, Flutter, BT832, CR2032

Keywords

microcontroller, Nordic Semiconductor, Bluetooth, BLE, electricity meter, electricity consumption, Flutter, BT832, CR2032

Citácia

HOCKICKO, Matej. *Systém pro registraci spotřebované elektrické energie*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vojtěch Mrázek, Ph.D.

Systém pro registraci spotřebované elektrické energie

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Vojtěcha Mrázka Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Matej Hockicko
16. mája 2024

Poďakovanie

Rád by som poďakoval pánovi doc. Ing. Zdeňkovi Vašíčkovi Ph.D. za rady, usmerňovanie a čas strávený na konzultáciách. Vedúcemu práce, pánovi Ing. Vojtěchovi Mrázkovi Ph.D., za pomoc a rady pri písaní tejto práce. Moja vďaka ďalej patrí aj pánovi Ing. Václavovi Šimkovi, za jeho ochotu a pomoc pri návrhu a zhotovení dosky plošných spojov. Nakoniec by som rád poďakoval aj pánovi Zdeňkovi Juříčkovi mladšiemu, za pomoc s návrhom a 3D tlačou krabičky pre vstavané zariadenie.

Obsah

1 Úvod **3**

2 Vstavané systémy **5**
2.1 O jadrách systému 5
2.2 Bezdrôtová komunikácia vo vstavaných systémoch 6
2.2.1 Technológia Wi-Fi 6
2.2.2 Technológia Bluetooth 7
2.3 Príklady platforiem 8
2.3.1 Espressif 8
2.3.2 Nordic Semiconductor 9
2.4 Spotreba 9

3 Využitie smartfónov na interakciu so vstavanými systémami **11**
3.1 Natívne aplikácie 11
3.1.1 Android 11
3.1.2 iOS 12
3.2 Moderné trendy vo vývoji mobilných aplikácií 12
3.2.1 Framework Flutter 13
3.2.2 React Native 13
3.2.3 Apache Cordova 14
3.3 Podpora Bluetooth LE 15

4 Návrh systému **16**
4.1 Elektromer 17
4.2 Návrh vstavaného systému 17
4.2.1 Monitorovanie výstupu na elektromere 18
4.2.2 Spracovávanie pulzov a zápis do pamäte 18
4.2.3 Komunikácia s mobilným zariadením 19
4.2.4 Využitie Bluetooth LE 19
4.2.5 Optimalizácia spotreby batérie 19
4.3 Mobilná aplikácia 21
4.3.1 Prihlasovacia obrazovka 23
4.3.2 Obrazovka s oprávneniami 23
4.3.3 Skenovacia obrazovka 24
4.3.4 Konfiguračná obrazovka 25
4.3.5 Hlavná obrazovka 26
4.4 Vzdialené ukladanie dát 27

5 Implementácia a testovanie vstavaného zariadenia **29**
5.1 Vývoj vstavaného systému . . . 29
5.1.1 Ukladanie nameraných hodnôt do pamäte flash . . . 31
5.1.2 BLE . . . 32
5.1.3 Hlavný beh programu . . . 33
5.1.4 Popis hardvéru a konfigurácia zariadenia . . . 33
5.2 Návrh a vyhotovenie PCB . . . 34
5.3 Návrh a tlač 3D modelu krabičky . . . 36

6 Implementácia a testovanie mobilnej aplikácie **39**
6.1 Dátové štruktúry a poskytovatelia . . . 39
6.1.1 BleProvider . . . 40
6.1.2 DataProvider . . . 41
6.2 Obrazovky . . . 42
6.2.1 Obrazovka prihlásenia . . . 43
6.2.2 Obrazovka s oprávneniami . . . 44
6.2.3 Skenovacia obrazovka . . . 45
6.2.4 Obrazovka s nastavením zariadenia . . . 46
6.2.5 Hlavná obrazovka . . . 46
6.3 Integrácia a využitie cloudových služieb Firebase . . . 49

7 Testovanie a vyhodnotenie systému **50**
7.1 Časová agregácia pulzov a zaplnenie pamäte . . . 50
7.2 Odhadovaná výdrž zariadenia . . . 50
7.3 Výsledná cena vstavaného zariadenia . . . 51

8 Záver **53**

Literatúra **55**

A Nasadenie vstaveného zariadenia na elektromer **58**

B PCB schéma **60**

C PCB layout **61**

D Obsah SD karty **62**

Kapitola 1

Úvod

S nezadržateľným vývojom v oblasti informačných technológii sa množstvo informácií, ktoré sme schopný zúžitkovať, stále zvyšuje. Prílev dát nám umožňuje prijímať lepšie informované rozhodnutia, zefektívňovať procesy a postupy, získavať nové poznatky a v neposlednom rade aj šetriť náš čas a zvyšovať kvalitu života.

Ako príklad môžu poslúžiť vstavané zariadenia, ktoré sa vyskytujú všade okolo nás. V moderných domácich spotrebičoch, ako práčky alebo umývačky riadu, sa nachádzajú senzory, s mocou ktorých sa tieto zariadenia rozhodujú o množstve vody a čase, ktoré bude potrebné na vykonanie programu. S vývojom výrobných procesov a s tým spojenou miniaturizáciou dnes dokážeme vtesnať elektronické zariadenia aj na miesta, kde to inak nebolo možné. To viedlo k vytvoreniu novej oblasti využitia. Vďaka nositeľným zariadeniam máme prehľad o našej fyzickej kondícií, kvalite spánku a ďalších ukazovateľoch, za pomoci ktorých vieme zlepšovať kvalitu nášho života a zdravia. Rôzne smart zariadenia nám poskytujú spôsoby, ako automatizovať opakujúce sa činnosti bez nutnosti našej interakcie a ponúkajú možnosti, ako ovládať ináč manuálne činnosti na diaľku skrz mobilné telefóny a hlasových asistentov.

Príchod smart zariadení zasiahol do rôznych odvetví života a mimo iného aj do spôsobu, akým dnes zaznamenávame spotrebu elektrickej energie ale aj iných komodít. Narozdiel od klasických elektromerov, ktoré zaznamenávajú spotrebu a poskytujú hlavný údaj na displeji, ktorý je nutné fyzicky odčítať, sa dnes stretneme aj s inteligentnými elektromermi. Tie pre prevádzkovateľov služieb poskytujú spôsoby ako odčítať spotrebu zákazníkov na diaľku a zákazníkom zase umožňujú (ak im to prevádzkovateľ sprístupní) získavať lepší prehľad o spotrebe. Aký dopad to môže mať na spotrebiteľov sa môžme dočítať v štúdií [21], ktorá skúmala prechod domácností z klasického elektromera na smart elektrometer. Z výsledkov tejto štúdie vyplýva, že spotrebitelia po nainštalovaní zariadenia, ktoré im poskytovalo informácie o využívaní elektrickej energie v reálnom čase, znížili svoju spotrebu v rozmedzí 11 až 17%. Autori štúdie poukazujú na zistenie, že prechod na smart elektromer a s ním spojené sprostredkovanie údajov o spotrebe elektrickej energie vedie k následným poklesom spotreby domácnosti.

V niektorých krajinách sú pomerne rozšírené smart elektromery, ktoré poskytujú informácie o spotrebe dodávateľovi aj užívateľovi. V Česku sa však tento prechod stále odkladá a v prvotnej fáze pôjde iba o nasadenie pre odberateľov s veľkou spotrebou, ktorá je nad 6MWh [18], pričom priemerná spotreba českej domácnosti sa pohybuje na úrovni 3MWh [17]. V tejto práci sa preto chcem zamerať na spotrebu domácností a konkrétne klasické elektrometre, pre ktoré navrhnem a zostavím systém na registráciu spotrebovanej elektrickej energie, pomocou ktorého bude možné premeniť klasický elektromer na smart

zariadenie. Táto práca je členená následovne. V kapitole 2 popíšeme aktuálne zariadenia a technológie využívané v oblasti vstavaných systémov. V 3. kapitole sa zamieram na trendy v oblasti vývoja aplikácií pre smartfóny ako aj na možnosti a podporu komunikácie a ovládania vstavaných zariadení. Následujúca kapitola 4 sa zameria na návrh celého systému, pričom dôraz bude kladený najmä na čo najdlhšiu výdrž vstavaného zariadenia. V kapitole 5 budem popisovať implementáciu výsledného vstavaného zariadenia. Predposledná kapitola 6 bude pojednávať o implementácií mobilnej aplikácie, ktorá bude slúžiť na konfiguráciu a synchronizáciu údajov zo zariadenia. V poslednej kapitole 7 vyhodnotím zhotovené riešenie a spomeniem možnosti na jeho ďalšie vylepšenie.

Kapitola 2

Vstavané systémy

Vstavané systémy dnes nájdeme v rôznych oblastiach života od priemyslu, cez medicínu, poľnohospodárstvo, až po domáce spotrebiče. Všade tam nájdeme využitia pre systémy, za pomoci ktorých je možné odbremeniť či uľahčiť prácu ľudom. Mnoho jednoduchých úkonov je tak možné zautomatizovať a zefektívniť.

Pod pojmom vstavané systémy rozumieme rôzne zariadenia, ktoré v sebe kombinujú hardvér a softvér ktorý spoločne poskytuje špecifickú funkcionalitu. Často ide o malé a cenovo dostupné zariadenia, ktoré obsahujú mikrokontrolér a vstupno výstupné periférie. Periférie poskytujú prostriedky, vďaka ktorým môže vstavaný systém „vnímať“ svoje okolie, prijímať informácie na vstupoch, ďalej ich uchovávať a spracovávať, a výsledky poskytovať výstupnými perifériami.

Vstavané zariadenie je zväčša určené na jednu konkrétnu úlohu, pre ktorú je navrhnuté a vytvorené. Môže sa vyskytovať aj vo väčších systémoch, v ktorých sa stará o špecifickú úlohu, ktorá môže vyžadovať napríklad prácu v reálnom čase. Vstavané systémy sú často optimalizované na výslednú veľkosť, spotrebu, či cenu.

Činnosť vstavaného systému sa typicky nezaobíde bez mikrokontroléru, ako hlavnej výpočtovej a riadiacej jednotky. Mikrokontrolér sa skladá z nevyhnutných komponentov, ako procesorové jadro, zvyčajne jednojádrové, ale môžeme sa stretnúť s viacerými jadrami, pamäť RAM a GPIO piny. Zároveň sa spolu s týmito prvkami na čipe vyskytuje aj pamäť flash. Rôzne typy mikrokontrolérov môžu ponúkať aj ďalšie hardvérové moduly, ako napríklad ADC[1], DAC[2], časovače, či hardvérovú podporu pre komunikačné protokoly, ako I2C, alebo SPI.

Na pripájanie modulov a drôtovú komunikáciu využívame GPIO piny. Pod pojmom GPIO rozumieme univerzálny vstup/výstup. V programe vieme GPIO pinom softvérovo priradiť daný smer (to, či ide o vstupný alebo výstupný pin). Niektoré GPIO piny na mikrokontrólery môžu poskytovať možnosť pripojenia na pull-up či pull-down rezistor, alebo hardvérovú podporu pre niektorý z protokolov určeného na komunikáciu.

2.1 O jadrách systému

Hlavným rozdielom v mikrokontroléroch dostupných v dnešnej dobe je najmä typ procesorových jadier. Najpopulárnejším a najrozšírenejším typom sú ARM jadrá založené na inštrukčnej sade typu RISC. Táto architektúra je populárna pre svoj výkon a nízku spotrebu,

[1] Analog to digital converter - prevodník analógového signálu na digitálny
[2] Digital to analog converter - prevodník digitálneho signálu na analógový

vďaka čomu sa skvelo hodí do vstavaných systémov, no často ju nájdeme aj v smartfónoch. Spoločnosť ARM priamo vytvára procesorové jadrá, a licencuje ich použitie. Spoločnosti si tak môžu zložiť procesor z týchto jadier podľa potreby, no jednotlivé jadrá si upraviť nemôžu. ARM jadrá sú rozdelené do niekoľkých rodín, pričom na využitie v mikrokontroléroch je zameraná rada Cortex-M. Mikrokontroléry využívajúce ARM Cortex-M jadrá môžme nájsť napríklad v portfóliu firmy Nordic Semiconductor.

V dnešnej dobe ale na popularite získava aj RISC-V architektúra, ktorá je takisto založená na princípoch RISC, avšak oproti ARM architektúre je voľne dostupná a nespájajú sa s ňou žiadne licenčné poplatky. Výrobcovia tak môžu použiť už existujúcu architektúru a zároveň si ju upraviť a vyladiť podľa svojich potrieb.

Procesorové jadrá typu RISC-V môžme nájsť medzi populárnymi mikrokontrolérmi od firmy Espressif, ktorá sa rozhodla výlučne prejsť na toto riešenie [20]. Stále však môžme nájsť mikrokontroléry tejto firmy využívajúce proprietárne riešenia firmy Tensilica a jej procesorové jadrá Xtensa.

2.2 Bezdrôtová komunikácia vo vstavaných systémoch

Funkcionalita vstavaných systémov veľakrát vyžaduje určitú formu komunikácie, či už s užívateľom, okolím, alebo inými zariadeniami. Pri komunikácii s inými zariadeniami môžeme využívať buď drôtovú, alebo bezdrôtovú komunikáciu. Pri drôtovej komunikácii sa zvyčajne jedná o zariadenia v tesnej blízkosti, ktoré prepájame pomocou káblov a na výmenu informácii využívame niektorý z komunikačných protokolov.

Na bezdrôtovú komunikáciu dobre poslúži Wi-Fi alebo Bluetooth modul. Pomocou bezdrôtovej komunikácie tak môžeme ovládať rôzne systémy na diaľku, získavať z nich informácie, ale takisto ich aj prepájať navzájom medzi sebou do väčších celkov, kde každý prvok plný svoju špecifickú úlohu.

V súvislosti so vstavanými systémami sa ďalej môžme stretnúť aj s technológiami ako ZigBee či LoRaWAN, ktoré riešenia potreby a požiadavky systémov, kde by využitie Wi-Fi alebo Bluetooth nebolo vhodným riešením. Môže sa jednať o potrebu na prepojenie viacerých zariadení medzi sebou, či potreba vysielať na dlhšie vzdialenosti. V neposlednom rade nájdeme aj také systémy, kde prebieha bezdrôtová komunikácia skrz telefónne siete.

2.2.1 Technológia Wi-Fi

Najnovší Wi-Fi štandart IEEE 802.11ax je označovaný ako Wi-Fi 6 alebo Wi-Fi 6E. Rozdiel medzi verziami 6 a 6E spočíva vo frekvenciách a šírke pásma, v ktorých pracujú. Wi-fi v štandarde 6 komunikuje na 2,4 a 5 GHz, zatiaľ čo Wi-Fi vo verzii 6E ďalej rozširuje tento štandard o komunikáciu na frekvencii 6 GHz, čo má za následok zvýšenie prenosových rýchlostí, pričom sľubuje nižšiu latenciu a dlhšiu životnosť pri napájaní z batérie pre IoT[3] zariadenia.[16]

Hlavným dôvodom prečo by sme sa pri návrhu vstavaného systému mohli rozhodnúť pre komunikáciu cez štandard Wi-Fi je hlavne rýchlosť bezdrôtového prenosu. To môže byť výhodné, ak plánujeme prenášať veľké objemy dát, ako napríklad pri streamovaní videa v prípade bezpečnostných kamier. Zároveň však môže ísť o nutnosť pripojenia do siete internet za účelom využívania služieb v danej sieti.

[3]Internet of Things - Internet vecí

2.2.2 Technológia Bluetooth

Táto technológia bola vytvorená na prepojenie mobilných zariadení, ako mobilné telefóny a notebooky. Ako sa táto technológia vyvíjala, pridávalo sa sem mnoho ďalších možností využitia. S tým boli spojené zvýšené nároky na šírku pásma, ktorá sa postupne zvyšovala. Oproti WiFi je síce značne pomalšia, no má ďalšie možnosti využitia.

Hlavným dôvodom, prečo sa ale rozhodnúť pre Bluetooth môže byť práve alternatívna verzia Bluetooth Low Energy (LE). Cieľom tejto technológie je poskytnúť bezdrôtovú komunikáciu s čo najnižšou spotrebou energie, čo je výhodne práve pre rôzne smart zariadenia, ktoré vyžadujú napájanie z batérie. Aj keď táto technológia stavia na podobných technológiách ako klasický Bluetooth, Bluetooth LE je zameraná na odlišné ciele a použitia. Bluetooth LE preto nebol vytvorený ako optimálnejšia verzia, ale ako nová samostatná technológia. Predpokladaným použitím sú najmä vstavané zariadenia, ktoré odosielajú iba jednotky dát v krátkych intervaloch s rozostupom od pár sekúnd až do pár dní.

BLE sa skladá z dvoch hlavných častí a to ovládača a hostiteľa. Časť ovládača zahŕňa fyzickú a linkovú vrstvu a zvyčajne je implementovaný ako systém na čipe s integrovaným rádiovým vysielačom a prijímačom. Hostiteľ v sebe zahŕňa funkcie vyšších vrstiev. Komunikácia medzi ovládačom a hostiteľom prebieha pomocou štandardizovaného rozhrania HCI (Host Controller Interface).

Keďže existujú rozdiely medzi Bluetooth Classic a A Bluetooth LE, nie sú tieto technológie medzi sebou kompatibilné. Preto ak zariadenie implementuje iba BLE (označujeme aj ako sigle-mode zariadenie), nemôže komunikovať zo zariadením, ktoré implementuje iba Bluetooth Classic. Dnes sa už však bežne stretneme so zariadeniami, ktoré implementujú oba režimy, BLE aj Bluetooth Classic zároveň. Takéto zariadenia sú označované ako dual-mode.

Fyzická vrstva

BLE na fyzickej vrstve využíva pásmo 2.4 GHz, nad ktorým definuje 40 frekvencií so šírkou 2 MHz. Tieto kanály sa rozdeľujú na 2 kategórie. Tri kanály sú označené ako oznamovacie. Tie sa využívajú na oznamovanie zariadenia na sieti, k nadväzovaniu spojenia a na broadcastové vysielanie, pričom ich frekvencie boli zvolené tak, aby sa minimalizovalo prekrytie s existujúcimi Wi-Fi kanálmi.

Zvyšné kanály sa využívajú na obojsmerný dátový prenos. Nad dátovými kanálmi sa využíva adaptive frequency hopping, aby sa tak čelilo problémom s bezdrôtovým rušením. Tento mechanizmus vždy zvolí jeden z dostupných 37 dátových kanálov na komunikáciu počas daného časového intervalu.

Všetky kanály na fyzickej vrstve zároveň využívajú GSFK (Gaussian Frequency Shift Keying) moduláciu, ktorej modulačný index sa nachádza v rozmedzí 0,45 až 0,55. BLE je na fyzickej vrstve schopné prenášať údaje rýchlosťou 1 Mbps.

Linková vrstva

Obojsmerná dátová komunikácia vyžaduje aby sa obe zariadenia najprv navzájom pripojili. Na to sa využíva asymetrický postup, pri ktorom jedno zariadenie oznamuje postupne na všetkých troch kanáloch, že sa je k nemu možné pripojiť, zatiaľ čo druhé (označené ako iniciátor), naslúcha tomuto vysielaniu na jednom z kanálov. Ak iniciátor nájde oznamovateľa, môže odoslať správu s výzvou na pripojenie a vytvoriť tak point-to-point spojenie.

Tieto dve zariadenia označujeme ako master a slave, pričom master je zariadenie, ktoré iniciuje spojenie. Master zariadenie môže simultánne spravovať viacero pripojených slave zariadení, zatiaľ čo slave zariadenie môže byť pripojené iba k jednému master zariadeniu. Spojenie existuje medzi zariadeniami tak dlho, dokým dochádza k pravidelnej výmene dát. Toto spojenie môže prerušiť ľubovoľná zo strán. [22, 31]

2.3 Príklady platforiem

Na analýzu som si vybral mikrokontroléry od firiem Espressif a Nordic Semiconductor, pretože oba tieto spoločnosti sú známe najmä vývojom mikrokontrolérov s podporou bezdrôtových technológií.

2.3.1 Espressif

ESP mikrokontroléry sú zariadenia od spoločnosti Espressif. Ide o veľmi populárne mikrokontroléry, ktoré sú podporované aj v Arduino IDE. Zároveň pre nich nájdeme kopec knižníc aj open-source systémov. Ako príklad poslúži systém ESPHome[4], s využitím ktorého je možné vzdialene ovládať a automatizovať domácnosť. Ich hlavnou výsadou a rozdielom oproti iným riešeniam je, že disponujú Wi-Fi a Bluetooth modulom a to Bluetooth vo verzii Classic spolu s jeho modernejšou verziou Bluetooth LE. Následujúci popis rôznych verzií bol prevzatý zo stránky výrobcu [2].

- ESP8266 - najzákladnejšia verzia, ktorá ponúka 32bitový jednojádrový procesor s frekvenciou až 160 MHz, 2 alebo 4 MB flash pamäte so spotrebou v spánkovom režime menej ako 20 µA
- ESP32 - táto rada ponúka naviac oproti predchádzajúcej ešte aj Bluetooth a Bluetooth LE modul. Zároveň je dostupná aj verzia, ktorá ponúka dvojjádrový procesor, pričom frekvenciu procesoru je možné upraviť v rozmedzí od 80 MHz do 240 MHz. V tomto prípade spotreba v režime spánku klesla až na 5 µA. Pri tejto variante je možné nájsť modely s veľkosťou pamäti až do 16 MB.
- ESP32-H - Tento modul sa skladá z jednojádrového 32bitového RISC-V procesoru s frekvenciou do 96 MHz. Narozdiel od ostatných sérií neponúka Wi-Fi, ale poskytuje Bluetooth LE a podporu štandardu IEEE 802.15.4[5]. Ďalšie parametre sú 320 kB SRAM, 128 kB ROM, FLASH pamäť o veľkosti 1, 2 alebo 4 MB a 19 programovateľných GPIO pinov.
- ESP32-C - Táto séria rovnako ako H séria ponúka RISC-V procesor. Séria sa delí do troch verzií C2, C3 a C6. Všetky série pozostávajú s jednojádrového procesoru. Zatiaľ čo maximálna frekvencia u S2 dosahuje 120 MHz, C3 a C6 má maximum až 160 MHz. Séria S6 navyše obsahuje Wi-Fi vo verzii 6 a zároveň využíva aj Zigbee modul.
- ESP32-S - S séria, je zameraná najmä na výkon. Využíva 32bitový Xtensa 32-bit LX7 procesor s taktom až 240 MHz a poskytuje bezpečnostné funkcie ako šifrovanie pamäti flash, bezpečné spustenie, overovanie podpisov a integrované algoritmy AES, SHA a RSA. S séria existuje v dvoch verziách - S2 a S3. S2 využíva jedno jadro a disponuje

[4] https://esphome.io/
[5] štandard, ktorý popisuje nízko rýchlostnú bezdrôtovú sieť

až 43 GPIO pinmi. Séria S3 je zostavená z dvojjadrového procesoru a pridáva Bluetooth LE. Navyše tieto mikrokontroléry disponujú vektorovými inštrukciami, vďaka ktorým ich je možné využiť aj na náročnejšie operácie ako výpočet neurónových sietí a spracovávanie signálov.

2.3.2 Nordic Semiconductor

V portfóliu spoločnosti Nordic Semiconductor nájdeme mikroprocesory, ktorých hlavný cieľ je poskytovať mikrokontroléry s bezdrôtovým pripojením s čo najnižšou spotrebou energie. V ponuke nájdeme niekoľko rôznych kategórii produktov. Produkty sú rozdelené podľa spôsobu akým môžu bezdrôtovo komunikovať. Ponuka obsahuje zariadenia komunikujúce pomocou mobilných sietí, Wi-Fi, Bluetooth Low Energy, Zigbee či ANT [24].

Ak sa zamierame napríklad na zariadenia poskytujúce Bluetooth Low Energy, máme na výber niekoľko rôznych variant. Jednotlivé varianty sa líšia najmä vo veľkosti pamäte flash, ktorá sa pohybuje od 192 kB do 1024 kB a pamäte RAM, ktorá je v rozmedzí od 24 kB do 256 kB. Ďalším rozdielom sú protokoly, ktoré jednotlivé verzie podporujú. V základe nájdeme Bluetooth vo verzií 5.3 a k nemu si môžme podľa potreby vybrať zariadenia, ktoré ďalej môžu ponúkať podporu pre Bluetooth Mesh, Zigbee, Matter, či Mesh. Všetky varianty až na jednu ponúkajú rovnaký procesor a to procesor Cortex-M4 typu ARM s taktom 64 MHz a jednotkou na výpočet desatinných čísel. Pre náročnejšie systémy tu nájdeme aj dvojjadrovú verziu, ktorá ponúka dve Cortex-M33 jadrá, pričom jedno z nich je taktované na frekvenciu 64 MHz, zatiaľ co druhé až na 128 MHz.

Čo sa spotreby týka, najvyššiu spotrebu dosahuje vysielač. Ak si ako príklad zoberieme verziu nRF52840, tak sa spotreba pri prenose dát cez Bluetooth pohybe okolo 16.4 mA pri maximálnom výkone antény 8 dBm, zatiaľ čo pri prednastavenom výkone 0 dBm spotreba dosahuje 6.4 mA. Výkon vysielača pritom vieme upraviť v rozsahu 8 dBm až do −20 dBm, čím vieme ďalej znížiť spotrebu. Spotreba samotného systému sa pritom pohybuje okolo 3.16 µA pri zapnutom systéme, udržiavaní pamäte RAM a zapnutom RTC[6]. Najnižšiu spotrebu dosiahneme pri vypnutom systéme a pri vypnutom napájaní pamäte RAM a to 0.4 µA.[1, 25]

2.4 Spotreba

Dôležitým bodom pri návrhu vstavaných systémov býva otázka spotreby. Ak sa výsledné zariadenie bude nachádzať na mieste bez prístupu k elektrickej sieti alebo sa od zariadenia očakáva jeho prenosnosť, tak je dôležité zamyslieť sa nad tým ako budeme dané zariadenie napájať. Pri výbere batérie na napájanie vstavaného systému musíme myslieť na kapacitu batérie, spôsob dobíjania/výmeny batérie, napätie, ktoré je batéria schopná dodávať, ale aj na napájacie napätie zariadia a v neposlednom rade aj na rozmery výsledného systému.

Výslednú spotrebu zariadenia a tým pádom aj výdrž na batériu vieme ovplyvniť zároveň aj správnym výberom mikrokontroléru. Rôzne zariadenia ponúkajú rozličné komponenty, výpočtové prostriedky, ale aj odlišné výrobné procesy a s nimi spojené rozmery tranzistorov, ktoré prispievajú k spotrebe výsledného celku. Pre konkrétnu modelovú radu mikrokontroléru potom môžme nájsť viacero verzií, ktoré sa môžu líšiť napríklad vo veľkosti pamäte na čipe, veľkosti RAM, frekvencií mikroprocesoru, alebo v počte vstupno výstupných portov. Čím výkonnejší mikrokontrolér si vyberieme, tým vyššie nároky na napájanie bude vyžadovať.

[6] Real Time Clock - modul hodín s reálnym časom

Tým však nekončia spôsoby znižovania spotreby energie. U mikrokontrolérov sa často stretávame s rôznymi režimami, v ktorých môžeme dané zariadenie prevádzkovať. Spravidla má miktrokontrolér základný režim, v ktorom je možné využívať všetku funkcionalitu zariadenia, ale zároveň ide o režim, ktorý pochopiteľne vyžaduje najviac elektrickej energie. Následne výrobcovia ponúkajú niekoľko rôznych režimov, do ktorých je zariadenie možné nakonfigurovať. Jednotlivé režimy sú odstupňované podľa tohom akú funkcionalitu dokážu poskytnúť spolu s poklesom spotreby. Čím nižšie pôjdeme, tým nižšia bude spotreba, ale zároveň tým menej možností máme. Často sa dokonca stretávame s režimom, kedy sa mikroprocesor dokonca uspí, aby sme získali čo najväčšiu úsporu, zatiaľ čo nepotrebujeme vykonávať žiadne výpočty. Mikroprocesor potom môžeme znova prebrať pomocou signálu, napríklad pomocou zmeny stavu na niektorom z podporovaných pinov, alebo aj pomocou hodín, ktoré môžu stále bežať a pravidelne prebúdzať zariadenie v určitom intervale.

Kapitola 3

Využitie smartfónov na interakciu so vstavanými systémami

Na interakciu so vstavanými systémami často využívame aj mobilné zariadenia, ako smartfóny, tablety, či notebooky, ktoré využívame na dennej báze a dalo by sa povedať, že sa bez nich už v modernej dobe nezaobídeme. Tieto zariadenia sú výkonnejšie a narozdiel od vstavaných systémov, ktoré nemusia disponovať vhodnými užívateľsky prívetivými prostriedkami na ovládanie, sú vybavené obrazovkami, na ktorých vieme jednoducho zobrazovať užívateľské rozhrania. Pomocou smartfónu sa tak dokážeme vzdialene pripojiť k vstavanému systému, či už skrze Wi-Fi, alebo Bluetooth a pomocou aplikácie bežiacej na tomto zariadení jednoducho ovládať systémy prostredníctvom dotykového displeja.

Mobilné aplikácie sú tak skvelým prostriedkom ako vytvárať užívateľské rozhrania na monitorovanie a ovládanie činnosti vstavaných systémov. Ak sa zameriame na vývoj aplikácií pre smartfóny, tak v súčastnosti sú využívané hlavne dve platformy, Android a iOS. Obe operačné systémy sú v hojnom počte zastúpené medzi užívateľmi. Aby sme sa neukrátili o potenciálnych užívateľov, je nevyhnutné vyvíjať pre obe tieto platformy zároveň. Pri vývoji mobilných aplikácii máme 2 hlavné cesty, ktorými sa môžme vydať. Na výber máme medzi natívnym vývojom a takzvaným cross-platform vývojom. Oba tieto prístupy si rozoberieme v následujúcich podkapitolách.

3.1 Natívne aplikácie

Pod pojmom natívne aplikácie rozumieme také aplikácie, ktorých vývoj je zameraný na jednu konkrétnu cieľovú platformu. V dnešnej dobe ide teda najmä o Android od Googlu a iOS od spoločnosti Apple [28]. Pri vývoji natívnej aplikácie sa zameriavame na jednu platformu a používame preto špecifické nástroje a prvky tohto cieľového operačného systému. Výslednú aplikáciu je následne možné spustiť iba na tejto platforme. Nevýhodou tohto prístupu je, že musíme vyvíjať samostatne pre každý systém, na ktorom chceme aby naša aplikácia bežala. Výhodou však je maximálny výkon, keďže môžme priamo využiť všetky komponenty zvolenej platformy a takisto jednoduchý a priamy prístup k funkcionalitám zariadenia ako napríklad ku kamere, Bluetooth, či GPS.

3.1.1 Android

Spoločnosť Google Inc., ktorá stojí za operačným systémom Android, stojí takisto za integrovaným vývojovým prostredím nazývaným Android Studio. Android Studio je špeciálne

navrhnuté na vývoj aplikácii priamo pre Android zariadenia. V programe nájdeme všetky potrebné nástroje na vývoj, testovanie a ladenie. Ďalej tu nájdeme aj správnu zariadení, v ktorom vieme vytvárať a následne simulovať rôzne zariadenia a tak vyvíjať bez nutnosti fyzického zariadenia. Samozrejmosťou je aj vývoj pre Wear OS, čo je verzia operačného systému Android, ktorá je uspôsobená pre beh na nositeľných zariadeniach. [6]

Vývoj Android aplikácií prebiehal najmä v jazyku Java. Neskôr sa Google rozhodol prejsť na programovací jazyk Kotlin. Ide o open-source jazyk vyvíjaný spoločnosťou JetBrains. Výhodou využívania jazyka Kotlin namiesto Javy je najmä menšie množstvo kódu, ktorý je zároveň čitateľnejší, bezpečnejší a jednoduchší na písanie asynchrónneho kódu [23]. Kotlin kód je zároveň možné spúšťať z java kódu a naopak, čo umožňuje jednoduchší prechod už napísaných Java aplikácií do Kotlin kódu. Android Studio zároveň disponuje nástrojom, ktorý dokáže do editora vložený Java kód automaticky previesť na jeho ekvivalentný zápis v jazyku Kotlin.

3.1.2 iOS

iOS je po Androide druhý najpoužívanejší mobilný operačný systém [28]. Stojí za ním firma Apple Inc. Podobne ako pri Androide aj v prípade iOS poskytuje Apple integrované vývojové prostredie s názvom Xcode. Toto IDE je pritom možné využiť na vytváranie programov a aplikácií nie len pre mobilné zariadenia iPhone, ale takisto aj na vývoj pre iné operačné systémy a zariadenia od spoločnosti Apple, a to konkrétne iPadOS, macOS, tvOS, či watchOS. Vývojové prostredie Xcode je však vyvíjané špeciálne pre zariadenia s operačným systémom macOS, je preto nevyhnutné vlastniť dané zariadenie, za účelom vývoja pre danú platformu.

V minulosti sa na vývoj využíval jazyk Objective-C, no neskôr, podobne ako tomu bolo pri Androide, sa Apple rozhodol pre zmenu, a začal podporovať a pretláčať modernejší jazyk Swift, ktorý zároveň sám vyvinul. Pri tvorbe jazyka bol kladený dôraz najmä na rýchlosť, jednoduchosť používania, čitateľnosť a bezpečnosť. Podľa Apple ide o výkonný programovací jazyk, ktorý sa je jednoduché naučiť [12]. Rovnako ako pri Kotline ide o open-source programovací jazyk. Keďže bol Swift vytvorený ako nástupca jazyka Objective-C, je možné využívať oba jazyky spoločne na vývoj aplikácií.

3.2 Moderné trendy vo vývoji mobilných aplikácií

Pri vývoji mobilných aplikácii máme zároveň možnosť využiť aj rôzne Frameworky[1], ktorých hlavnou výhodou oproti natívnemu vývoju je to, že zaobaľujú špecifické vlastnosti jednotlivých platforiem a pre programátora poskytujú jednotný prístup vrámci jedného kódu naprieč rôznymi zariadeniami. Nie len, že tak môžme vyvíjať zároveň pre Android a iOS, ale dokonca aj pre Windows, alebo pre webové aplikácie. Tým sa výrazne skráti čas a náklady na vývoj. Hoci frameworky umožňujú rýchlejší a efektívnejší vývoj mobilných aplikácií, existujú aj niektoré nevýhody ich používania.

Jednou z najväčších nevýhod je obmedzenie flexibility, keďže vývojári používajú hotové prvky a funkcie z frameworku, čím sa oberajú o možnosť využívať špecifické možnosti cieľovej platformy. Frameworky síce ponúkajú možnosť písať špecifický kód pre cieľovú platformu, no jeho väčšie množstvo prispieva k diverzifikácií a horšej čitateľnosti kódu, čo je presný opak toho, o čo sa frameworky snažia. Ďalšou nevýhodou je, že frameworky

[1]Sada nástrojov a funkcionalít, ktoré pomáhajú pri softvérovom vývoji, s dôrazom na univerzálnosť a znovupoužitie, s pomocou ktorých je ďalej tvoriť vlastné aplikácie

môžu byť veľké a zložité, čo môže spomaliť výkon aplikácie a zvýšiť jej nároky na pamäť v zariadení. Navyše ak spoločnosť, ktorá vytvorila framework prestane podporovať alebo vyvíjať svoj produkt, môže to mať negatívne dôsledky na podporu aplikácie na budúcich verziách danej platformy. Medzi najznámejšie frameworky, ktoré umožňujú vývoj mobilných aplikácii patria Flutter, React Native a Apache Cordova.

3.2.1 Framework Flutter

Flutter je framework, ktorý bol vytvorený a je spravovaný spoločnosťou Google. Ide o open source framework, čo znamená, že na jeho zlepšovaní a vývoji sa môže podieľať každý, kto má o to záujem, keďže jeho zdrojový kód je verejne dostupný. Flutter framework zároveň využíva aj samotný Google na vývoj svojich vlastných aplikácii. Ako príklad môžeme spomenúť aplikáciu Google Play, ktorá bola pôvodne napísaná ako natívna aplikácia, čo však prinášalo problémy pri vývoji, keďže aplikácia bola vyvíjaná zároveň pre operačné systémy Android a iOS. Prepísaním aplikácie do frameworku Flutter sa znížila veľkosť aplikácie. Aplikácia bola zároveň jednoduchšia na spravovanie a aktualizovanie. V porovnaní s originálnou implementáciou šlo dokonca o 35% pokles v počte riadkov kódu, z pôvodných 1,7 milióna na 1,1 milióna. Zároveň tím pracujúci na tomto projekte odhadol 60 až 70% úsporu času inžinierov ako dôsledok jednoduchšej správy projektu. [5]

Flutter využíva Dart ako programovací jazyk. Dart vznikol takisto pod záštitou spoločnosti Google ako open source a ide o moderný jazyk, ktorý sa využíva najmä na programovanie užívateľských rozhraní. Jedná sa o objektovo orientovaný jazyk, ktorý je možné skompilovať do natívneho kódu pre cieľové zariadenie alebo do javascriptu, ak vyvíjame webovú aplikáciu.

Flutter poskytuje obsiahlu dokumentáciu[2], ktorá častokrát obsahuje okrem popisu a ukážok kódu pre daný objekt aj krátke video, kde je vysvetlený základný princíp použitia konkrétneho objektu spolu s vizuálnymi ukážkami a časťami kódu najčastejšie používaných funkcií.

Architektúra aplikácii vytvorených vo Flutter frameworku pozostáva z nezávislých knižníc, z ktorých každá závisí na vrstve nachádzajúcej sa pod ňou. Základom je vrstva Embedder, ktorá je špecifická pre každú platformu. Táto vrstva poskytuje vstupný bod pre prístup k službám operačného systému ako zobrazovacie plochy a vstupy a zároveň riadi slučku udalostí. Nad touto vrstvou sa nachádza Flutter engine, napísaný hlavne v jazyku C++, obsahujúci primitíva nevyhnutné na podporu Flutter aplikácií. Tento engine implementuje knižnice pre animácie, grafiku, prístup k súborom a vstupno výstupným rozhraniam a ďalším. Najvyššou vrstvou je samotný Flutter framework, kde vývojári vytvárajú aplikácie za použitia rôznych balíčkov.

Základným stavebným prvkom vo vývoji Flutter aplikácií sú widgety. Widget predstavuje nemennú deklaráciu konkrétnej časti užívateľského rozhrania. Widgety sa spolu spájajú a vnárajú jeden do druhého, čím tvoria hierarchickú štruktúru, pričom na vrchole sa nachádza koreňový Widget. Ten zaobaľuje fungovanie celej aplikácie. [3]

3.2.2 React Native

React Native je ďalšia z open source možností ako vyvíjať mobilné aplikácie. React Native bol predstavený v roku 2015 spoločnosťou Facebook. Aplikácia Facebook je vytvorená za pomoci React Native Frameworku, ale aj iné spoločnosti, ako napríklad Microsoft využíva

[2] https://api.flutter.dev/

túto technológiu na vývoj svojich vlastných aplikácií pre smartfóny ako Office, Outlook, Teams či skype [13].

Na vývoj v React Native sa využíva Javascript knižnica React, určená na vývoj užívateľských rozhraní. React ako taký sa využíva v spojitosti s vývojom webových aplikácií, no v React Native komponenty React zaobaľujú existujúci natívny kód a interagujú priamo s natívnymi API[3] prostredníctvom deklaratívnej paradigmy používateľského rozhrania React a JavaScriptu. [11]

Pôvodná architektúra v React Native oddeľovala natívnu časť od vrstvy s JavaScript kódom. Tieto dve súčasti boli prepojené pomocou mostu, cez ktorý sa prenášali serializované dáta z vrstvy JS na natívnu vrstvu. Tento most si je možné predstaviť ako zbernicu, na ktorej je producent, ktorý vytvára dáta a konzument, ktorý ich spracúva. To sa však spájalo s nedostatkami. Keďže šlo o nesynchrónnu komunikáciu, udalosti neboli vykonávané v reálnom čase, ale boli plánové ako akcie. Zároveň z prenosom údajov z jednej vrstvy na druhú sa spájajú režijné náklady zo serializáciou a deserializáciou prenášaných dát do formátu JSON.

V súčastnosti však React Native prechádza na novú architektúru. V novej verzii sa upustilo od využitia mostu a prichádza sa s novým komunikačným mechanizmom nazývaným JavaScript Interface (JSI). Toto rozhranie po novom umožňuje komunikáciu nie len JavaScript vrstve s natívnou vrstvou, ale aj naopak. Vďaka tomu je možné aby JS objekt udržoval odkaz na natívny objekt a naopak. Objekty sa tak môžu lepšie synchronizovať. S novým rozhraním je možné vykonávať funkcie synchrónne, ak je to nutné, a zároveň je možné z JavaScriptu spúšťať funkcie, ktoré sa vykonávajú na odlišných vláknach, čo pred tým nebolo možné. [14, 27]

3.2.3 Apache Cordova

Apache Cordova je ďalším open source frameworkom na vývoj mobilných aplikácií. Na vývoj aplikácií využíva webové technológie ako HTML, CSS a JavaScript. Zdrojový kód následne zaobaľuje a spúšťa na cieľovej platforme, využívajúc jej API. Tento prístup vývoja je vhodný najmä pre vývojárov, ktorý už majú existujúcu webovú aplikáciu, z ktorej by chceli vytvoriť aj mobilnú aplikáciu. Následujúca sekcia je prevzatá z dokumentácie [30].

Architektúra aplikácie sa skladá z WebView, webovej aplikácie a pluginov. WebView s podporov Cordova môže poskytovať aplikácií celé jej používateľské rozhranie. Na niektorých platformách sa môže jednať o komponent hybridnej aplikácie, ktorá zároveň využíva natívne komponenty. Aplikácia je implementovaná ako webová stránka. Základom je súbor `index.html`, ktorý sa ďalej odkazuje na štýly v CSS súbore, JavaScript kód, obrázky a ďalšie zdrojové súbory, bez ktorých by nebolo aplikáciu možné spustiť. Dôležitým prvkom je aj súbor `confix.xml`, v ktorom sa definujú parametre aplikácie. Doplnky, ako posledná súčasť architektúry. slúžia na vytvorenie rozhrania, pomocou ktorého môžeme skrz JavaScript kód komunikovať priamo s natívnym API. Bežné funkcionality sú dostupne ako základné doplnky a v prípade potreby špecifických komponent môžeme použiť doplnky tretích strán.

Pri vývoji v tomto frameworku sa môžme vybrať dvoma smermi. Buď budeme vyvíjame na viacero platforiem zároveň, alebo sa zameriame iba na jednu platformu. Ak zvolíme cross-platformový vývoj, tak sa nezaobídeme bez cordova CLI[4]. Pomocou tohto CLI vytvára podpriečinky pre každú cieľovú platformu, kde nakopírováva zdrojový kód a upravuje jeho konfiguráciu podľa potrieb cieľovej platformy, pridáva pluginy a vytvára binárne sú-

[3] Aplikačno programové rozhranie
[4] Command line interface - príkazový riadok

bory aplikácie. Tento vývoj je odporúčaný. V prípade potreby môžme využiť druhý prístup, ktorý umožňuje spolu s Cordova komponentami využívať aj natívne komponenty cieľovej platformy. Cross-platformový vývoj je časom možno zmeniť aj na vývoj zameraný na konkrétnu platformu, avšak opačným smerom to možné nie je.

3.3 Podpora Bluetooth LE

Pri návrhu mobilnej aplikácie bude potreba zvoliť vhodnú platformu a framework. Dôležitým bodom pri návrhu bude podpora Bluetooth Low Energy. Ak sa rozhodneme pre natívny vývoj, tak sa môžme spoľahnúť na to, že priamo vo vývojom prostredí nájdeme podporu pre Bluetooth a Bluetooth LE. Odlišná situácia ale nastane ak zvolíme niektorý z frameworkov.

Vo frameworku Flutter nenájdeme priamo podporu technológie BLE a musíme sa preto obrátiť na nezávislých vývojárov, ktorý sa starajú o vývoj niektorého z dostupných balíkov na implementáciu tejto technológie. Niektoré z týchto balíkov však už neboli aktualizované viac ako 2 roky. Niektoré balíky zase implementujú iba úzko špecifickú časť technológie, alebo nepodporujú všetky platformy. Z ďalších balíčkov je možné spomenúť bluetooth_low_energy, ktorý je pravidelne aktualizovaný, no nepodporuje funkcie ako zmenu veľkosti dátového paketu alebo zmenu módu PHY. Plugin universal_ble podporuje viacero platforiem, ako Web, Windows, Linux, macOS, ako aj oba hlavné mobilné operačné systémy, no pre Web a zariadenia od Apple nepodporuje možnosť párovania. Najpopulárnejšími doplnkami pre framework Flutter sú flutter_blue_plus a flutter_reactive_ble. Prvý spomínaný ponúka širokú škálu funkcií, no nie všetky z nich fungujú na iPhone zariadeniach. Zároveň nepodporuje Web a desktopové aplikácie a je ho možné využívať len v roli centrálneho zariadenia. Druhý spomínaný ponúka tak isto implementáciu mnohých funkcií Bluetooth Low Energy a je vyvíjaný spoločnosťou Philips, no zároveň má kopec otvorených problémov, ktoré ostávajú nevyriešené.

Podpora Bluetooth Low Energy v prípade React Native je o trocha horšia ako v prípade frameworku Flutter. Výber je možné zúžiť na 2 hlavné doplnky, react-native-ble-manager a react-native-ble-plx. Oba podporujú Android a iOS, no druhý spomínaný navyše pridáva aj Web, no ani jeden nepodporuje Mac, Windows, alebo Linux. Prvý podporuje mnoho funkcií, no nie všetky, ako napríklad zmena MTU, alebo spárovanie zariadenia sú podporované na iOS. Druhý balíček síce podporuje funkciu vyjednávania veľkosti MTU aj na Apple zariadeniach, no funkciu párovania nepodporuje ani na smartfónoch so systémom Android.

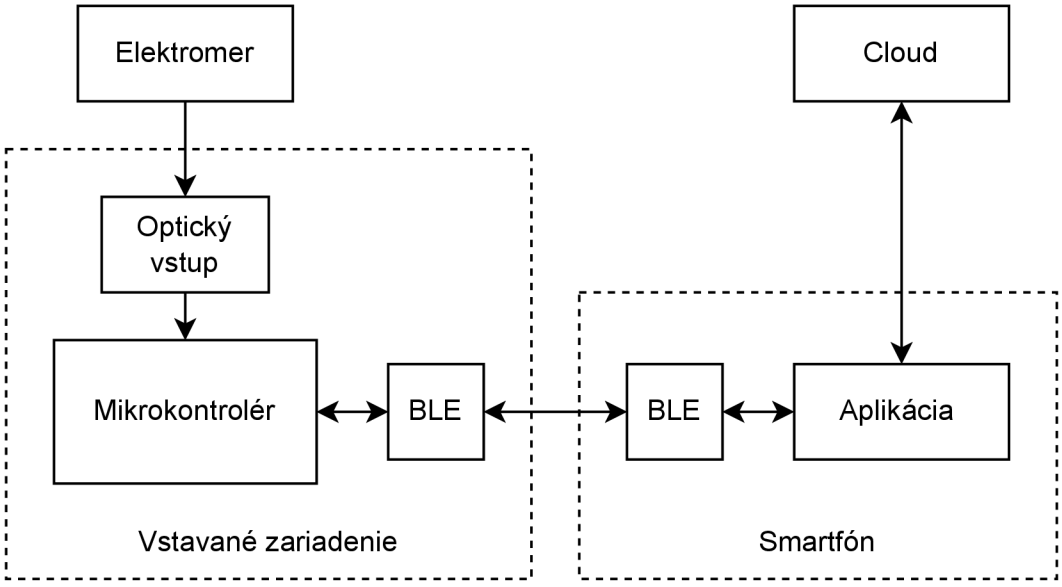
Apache Cordova ponúka niekoľko balíčkov, ktoré implementujú technológiu Bluetooth Low Energy, mnohé však už nie sú ďalej vyvíjané. Ak by sme chceli zariadenie naprogramovať do role periférneho zariadenia, mohli by sme použiť plugin cordova-plugin-ble-peripheral, ten však už nebol aktualizovaný viac ako 6 rokov. Pre centrálne zariadenie môžeme nájsť balíčky cordova-plugin-ble-central a cordova-plugin-bluetoothle. Situácia je tu však podobná ako pri predchádzajúcich frameworkoch. Podporované sú iba systémy Android a iOS, pričom niektoré funkcie je možné využiť len na jednej z týchto platforiem.

Situácia medzi všetkými troma frameworkami je podobná. Na výber je len pár knižníc, ktoré ponúkajú implementáciu technológie Bluetooth LE. V týchto knižniciach často nie sú implementované všetky funkcie, alebo implementované funkcie nie sú podporované na všetkých platformách. Mnoho knižníc je už zastaralých a ďalej nespravovaných. Spravované knižnice síce dostavajú aktualizácie, no aj tak je u nich možné nájsť mnoho nedoriešených problémov. V prípade jednoduchého využitia je podpora dostatočná, no v prípade pokročilejších funkcií naprieč viacerými platformami je podpora problematická.

Kapitola 4

Návrh systému

Cieľom práce je zaznamenávať spotrebu elektrickej energie. Pretože chceme zaznamenávať spotrebu celej domácnosti, ideálne na jednom mieste, zameriame sa priamo na elektromer. Jednou z možností ako zaznamenávať spotrebu je využiť S0 výstup elektromera, ktorý ale nemusí byť dostupný, alebo prístup k nemu môže byť obmedzený dodávateľom elektrickej energie. Preto sa sústredíme na ďalšiu možnosť, ktorou je monitorovanie optického výstupu.



Obr. 4.1: Schéma navrhovaného systému

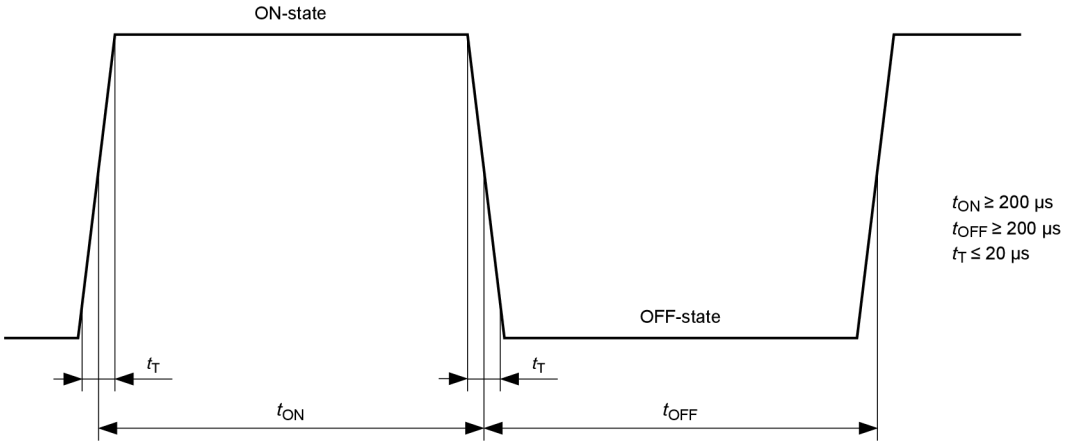
Tento signál bude ďalej nutné zaznamenať a prideliť mu časové razítko, aby bolo neskôr možné zrátať spotrebu podľa rôznych kritérií. Na to bude nevyhnutné využiť mikrokontrolér, ktorý bude súčasťou vstavaného systému. Keďže však mikrokontroléry poskytujú iba obmedzené výpočtové a pamäťové prostriedky, samotné zobrazenie a analýza dát sa musí odohrávať mimo nich. Nezaobídeme sa preto bez komunikácie s ďalším zariadením. Na výmenu informácii využijeme bezdrôtovú komunikáciu pre jej praktickosť a flexibilitu. Z dispozícii máme najmä dve rozšírené a známe štandardy, a to konkrétne Wi-Fi a Bluetooth.

Keďže však je cieľom minimalizovať výslednú spotrebu vstavaného zariadenia, využijeme Bluetooth Low Energy ako vhodnejšiu variantu na danú úlohu.

Na druhej strane budeme potrebovať zariadenie schopné komunikovať so vstavaným systémom pomocou zvoleného štandardu, s možnosťou zobrazovania a analyzovania dát. Na tento účel sme zvolili smartfón, aby sme tak predišli ďalším nákladom spojených so zaobstarávaním ďalšieho zariadenia na tento účel. Pre smartfón bude nutné navrhnúť a zostaviť aplikáciu komunikujúcu so vstavaným zariadením a zároveň schopnú interpretovať a zobrazovať namerané údaje. Ďalej sme sa rozhodli, že využijeme aj niektorú z dostupných cloudových služieb, pričom dôraz bude opäť kladený najmä na cenovú dostupnosť a použiteľnosť. V cloude sa tak budú zhromažďovať údaje za celú dobu merania a prístup k nim bude poskytovaný práve pomocou mobilnej aplikácie. Výslednú schému navrhovaného systému je možné vidieť na obrázku 4.1.

4.1 Elektromer

Pri návrhu systému nesmieme taktiež zabúdať na technické parametre elektromera. Údaje, na ktoré musíme brať ohľad pri návrhu nájdeme v technickej dokumentácií [15]. Zaujíma nás najmä optický výstup, ktorý je popísaný v prílohe A.



Obr. 4.2: Priebeh signálu optického výstupu. Prevzaté z [15]. Upravené.

Z obrázku 4.2 vyplýva, že šírka ON/OFF signálu musí byť najmenej 200µs. Je teda nevyhnutné aby bolo zariadenie schopné reagovať na takéto krátke impulzy. Z dĺžky impulzov viem ďalej odvodiť maximálnu frekvenciu optického výstupu s čím treba neskôr rátať pri návrhu spôsobu, akým sa budú pulzy zaznamenávať.

4.2 Návrh vstavaného systému

Vstavaný systém bude plniť iba jednoduchú úlohu, preto mikrokontrolér, ktorý sa v ňom bude nachádzať nepotrebuje vysoký výkon. Zariadenie sa môže nachádzať na mieste, kde sa nebude vyskytovať možnosť napájať ho z elektrickej siete, preto bude napájané z batérie. Dôležitým aspektom pri návrhu a vývoji tak bude dôraz na čo najnižšiu spotrebu samotného mikrokontroléru a jeho periférií spoločne s ďalšími súčasťami vstavaného systému.

Keďže zariadenia od firmy Espressif poskytujú kopu výkonu a v niektorých verziách navyše aj Wi-Fi modul, s ktorými sa spája aj vyššia spotreba zariadenia, rozhodli sme sa skôr využiť mikrokontrolér od firmy Nordic Semiconductor s Bluetooth modulom, zameraný najmä na čo najnižšiu spotrebu energie.

Samotný mikrokontrolér však nepostačuje na monitorovanie optického výstupu elektromera. Na to bude potreba navrhnúť hardvérový obvod, ktorý bude schopný zaznamenávať tento optický signál a prevádzať ho na elektrický, ktorý bude ďalej prenášaný a zaznamenávaný na jednom z GPIO pinov mikrokontroléru.

Na napájanie sme sa rozhodli zvoliť gombíkovú batériu typu CR2032. Táto batéria je pomerne lacná a dá sa jednoducho zohnať a zároveň má malé rozmery no dostatok energie. Batéria má napätie 3V a je ju teda možné priamo napojiť na mikrokontrolér bez nutnosti regulátora napätia. Kapacita takejto batérie sa pohybe zvyčajne okolo 220 mAh.

4.2.1 Monitorovanie výstupu na elektromere

Jednou z možností, ktorou by sme mohli zaznamenávať pulzy optického výstupu elektromera, je využite fotorezistoru. Toto riešenie má síce prijateľnú spotrebu ak a neho nesvieti svetlo, no pri každom impulze LED diódy sa zároveň zníži odpor na fotorezistore, čo vedie k citeľnému zvýšeniu spotreby [4]. Rozhodli sme sa preto využiť iný postup, ktorý bude úspornejší.

Na prijímanie signálu využijeme obvod popísaný v jednom z návodov na možné aplikácie súčiastky TLV7031 [29]. V tomto dokumente sa popisuje využitie komparátoru spolu s fotodiódou na zaznamenávanie infračervených signálov. Komparátor je ďalej napojený na mikrokontrolér, ktorý prebúdza pri zmene signálu. Prúd pretekajúci týmto obvodom je odhadovaný na 2 µA, pričom je v ňom už zahrnutá aj spotreba vypnutého mikrokontroléra (tá by mala byť nižšia ako 1 µA).

Tento obvod využijeme na zaznamenávanie vstupu, no narozdiel od infračervenej diódy využijeme fotodiódu, ktorá bude schopná zaznamenávať aj viditeľné svetlo aby tak zachytila červené svetlo vyžarované diódou na elektromere.

4.2.2 Spracovávanie pulzov a zápis do pamäte

Spôsob zaznamenávania jednotlivých hodnôt bude spočívať v agregácií pulzov za jednotku času. Táto jednotka času by mala byť zvolená tak, aby umožňovala dostatočné rozlíšenie spotreby, zároveň však musíme počítať s obmedzeným úložným priestorom, ktorý bude limitujúci v počte jednotlivých záznamov, ktoré bude možné uložiť. Z obrázku 4.2 vyčítame, že jeden pulz trvá najmenej 400µs. Maximálna frekvencia výstupného signálu je teda 2500Hz. V prípade, že si zvolíme agregáciu jednotlivých pulzov v 5 minútových intervaloch, tak zistíme, že budeme potrebovať každých 5 minút ukladať hodnotu o maximálnej veľkosti 10 000. Znamená to, že budeme zapisovať 2B. Nami zvolený mikrokontrolér disponuje 512kB pamäte. Za predpokladu, že kód zariadenia zaberie 50% pamäte, tak na ukladanie dát ostáva 256kB. Po prepočte by teda zariadenie malo byť schopné ukladať údaje po dobu zhruba 444 dní.

Na ukladanie do pamäte poslúži cyklický buffer. Pri inicializácií zariadenia sa najskôr do pamäte uloží časové razítko a od tohto momentu sa začnú sčítavať pulzy v jednotlivých časových oknách. Súčet z každého takéhoto okna sa zapíše do pamäte hneď za následujúci záznam. Do pamäte zariadenia sme schopný zapisovať najmenej 4B, preto vždy budeme zapisovať 2 hodnoty spoločne. Samotné číslo ukladáme bez metadát, keďže sémantika údajov je dáta spôsobom akým údaje do pamäte zapisujeme. Sme tak schopný ukladať údaje

efektívnejšie. Zároveň sa takýmto spôsobom pamäť zariadenia opotrebováva rovnomerne. V prípade, že sa pamäť vyčerpá, je možné postupovať dvoma spôsobmi. Buď zariadenie prestane zapisovať ďalšie údaje, alebo je nutné začať prepisovať najstaršie hodnoty. V tomto prípade však netreba zabúdať na to, že sa nám zároveň mení aj časové razítko prvého záznamu. Počas aktualizácie dát cez mobilné zariadenie vždy odošleme všetky údaje, uvoľníme pamäť a pokračujeme od následujúceho miesta v pamäti.

4.2.3 Komunikácia s mobilným zariadením

Pri komunikácií s mobilným zariadením musíme dbať hneď na niekoľko vecí. V prvom rade musíme pri prenose dát zabezpečiť, aby sa údaje prenášali v takom poradí v akom boli zapísané do pamäte, aby sa tak zachovala časová postupnosť záznamov. Je na ďalšom spracovaní v mobilnej aplikácií, aby sa jednotlivým hodnotám priraďovali časové údaje.

Ďalej chceme pri posielaní dát stráviť čo najmenej času samotným prenosom a to z dôvodu aby sme čo najmenej využívali anténu a obmedzili tak spotrebu batérie. Na to bude dôležité správne nastaviť MTU, aby sme v jednom pakete dokázali poslať viacero hodnôt zároveň a zvýšili tak pomer dát k údajom v hlavičke paketu. Zrýchliť komunikáciu však vieme aj využitím 2M PHY módu tam kde je to možné a kde to zariadenia podporujú.

V neposlednom rade potrebujeme zabezpečiť správne riadenie komunikácie. Existuje niekoľko možností akým môžeme zasielať údaje medzi zariadeniami. Najrýchlejšia možnosť ako zasielať údaje, ak nepotrebujeme komunikovať obojstranne, je práve využitie notifikácií. Vstavané zariadenie by tak malo poskytovať možnosť prihlásiť sa k odberu. Ak dôjde k prihláseniu k odberu, zo zariadenia by sme mali odoslať všetky do tej doby uložené hodnoty. Na koniec je potrebné zariadeniu oznámiť, že došlo k ukončeniu prenosu. Po tomto oznámení by sa mobilné zariadenie malo odhlásiť z odberu a ukončiť spojenie, aby sa nespotrebovávala energia na udržiavanie spojenia, ak to nie je nutné. Dĺžka paketu sa môže líšiť, preto môžeme využiť prázdny paket na signalizáciu, že už vstavané zariadenie nemá žiadne ďalšie údaje na odoslanie, a že je možné ukončiť spojenie. To je možné využiť v prípade, že zariadenie už odoslalo všetky svoje dáta, rovnako tak ako aj na prípad, kedy zariadenie neodoslalo žiadne údaje a chce tak oznámiť, že ani žiadne údaje na odoslanie nemá.

4.2.4 Využitie Bluetooth LE

Bluetooth modul by mal byť neustále zapnutý, aby tak bolo možné pripojiť sa k zariadeniu kedykoľvek na diaľku. Aby sme však zabezpečil čo najnižšiu spotrebu, budeme experimentálne testovať rôzne úrovne výkonu antény, aby sme našli vhodný kompromis medzi dosahom signálu a spotrebou batérie.

Ak chceme neustále oznamovať prítomnosť nášho zariadenia na sieti, nemôžeme využívať najúspornejší mód mikrokontroléru - System OFF. V tomto režime nie je dostupný Bluetooth LE, keďže procesor nebeží a všetky hodiny a periférie sú vypnuté. Zariadenie je možné opäť spustiť len pomocou signálu na GPIO pine alebo pomocou resetu.

Využijeme teda režim System ON, v ktorom je možné systém prebudiť pomocou akéhokoľvek prerušenia. V tomto móde procesor nevykonáva žiadnu akciu. Jednotlivé periférie sú automaticky vypínané, ak sa nevyužívajú. [8]

4.2.5 Optimalizácia spotreby batérie

Mikrokontrolér bude väčšinu času v úspornom režime System ON, aby sa čo najviac šetrila energia, a aby zároveň bolo možné využívať Bluetooth LE. Tento režim je zároveň predna-

stavený automaticky. Režim System OFF, ktorý je úspornejší, nie je možné využiť, pretože by sme tak kompletne vypli procesor a všetky jeho periférie. Nevedeli by sme tak reagovať na pripojenie pomocou BLE. Zároveň by sa mikrokontrolér pri každom prebudení restoval a ukladať stav by tak bolo dosť náročné. V režime Power ON bude procesor nečinný, ak nebude mať nič na práci, no bude reagovať na prerušenia. Prerušenia budú prichádzať od senzora a z Bluetooth modulu. Ak systém zaznamená prerušené vzniknuté rozsvietením LED diódy na elektromere, zariadenie sa prebudí, spracuje sa udalosť a aktualizujú sa hodnoty v pamäti, po čom sa zariadenie opäť vráti do predchádzajúceho stavu. Pri príchode prerušenia od modulu Bluetooth zariadenie odošle namerané hodnoty a uvoľní miesto v pamäti pre ďalšie merania, alebo príjme zmenu svojej konfigurácie z pripojeného zariadenia a následne, po ukončení výmeny údajov, prejde späť do režimu úspory energie.

Okrem procesoru zariadenie najviac vybíja komunikácia pomocou modulu BLE. Na zníženie spotreby tohto modulu je dôležité nakonfigurovať rôzne parametre chovania modulu. Spoločnosť Nordic poskytuje webstránku[1], na ktorej je možné simulovať spotrebu zariadenia na základe rôzne zvolených parametrov pre BLE komunikáciu. Úpravou parametrov môžeme výrazne znížiť spotrebu zariadenia.

Na začiatok si môžeme všimnúť, že zariadenie sa môže nachádzať v rôznych modoch. Zariadenie môže oznamovať svoju prítomnosť na sieti, alebo byť pasívne, či vykonávať inú funkciu. Môžeme si všimnúť, že spotreba pri oznamovaní prítomnosti na sieti sa môže vyšplhať až na 12mA, čo je rádovo vyššia spotreba, než pri nečinnosti zariadenia. Túto spotrebu môžeme znížiť tak, že upravíme interval, ktorý udáva, ako často zariadenie vysiela do okolia. Tento interval môžeme nastaviť v rozmedzí od 20ms až po $10,24$s a to po krokoch o dĺžke $0,625$ms. Čím kratší interval, tým jednoduchšie je nájsť zariadenie na sieti a pripojiť sa k nemu, no samozrejme o to vyššia spotreba. Na zníženie spotreby skúsim nastaviť tento interval na 2s a zistiť, či je tento interval dostatočný na spoľahlivé pripojenie v rade jednotiek sekúnd. Vyššie hodnoty by už značne znížili schopnosť pripojenia k zariadeniu a rozdiel v spotrebe by nebol až tak markantný.

Ďalšie nastavenie, pomocou ktorého vieme znížiť spotrebu zariadenia je rádio mód. Zariadenie by malo využívať mód 2M PHY, ktorý zaručí, že sa údaje medzi zariadeniami budú posielať vyššou rýchlosťou, čo znamená, že celkový čas vysielania, a teda využívania antény sa skráti, a to zníži celkovú spotrebu batérie.

Keďže sa medzi zariadením a mobilom bude presúvať väčšie množstvo dát, je nutné, aby zariadenie takisto využívalo maximálnu možnú veľkosť PDU paketu. Na to je najskôr nutné povoliť možnosť rozšírenia dĺžky paketu a následne upraviť jeho dĺžku na najväčšiu možnú. Tým docielime to, že v jednom pakete prenesieme viac údajov a zníži sa tým réžia spojená s vypĺňaním hlavičiek paketu. V konečnom dôsledku tak prenesieme rovnaký počet údajov s menším počtom dát.

Spotrebu zariadenia pri využívaní Bluetooth Low Energy môžeme ešte ďalej znížiť, a to tak, že budeme využívať externý nízko frekvenčný kryštál, namiesto vnútorného oscilačného RC obvodu. Tento kryštál je ďaleko presnejší a podľa simulácií tak môže znížiť spotrebu zariadenia. Interný RC obvod by zároveň bolo potrebné neustále kalibrovať, čo stojí ďalšiu energiu naviac.

Spotrebu energie môžeme ďalej znižovať úpravou výkonu, akým bude anténa vysielať. Ak nepotrebujeme veľký dosah na zariadení, tak môžeme znížiť výkon antény a znížiť tak aj spotrebu. Konkrétna hodnota však už bude závisieť na testovaní, aby sa dosiahla dostatočná spoľahlivosť spojenia na požadovanú vzdialenosť.

[1] https://devzone.nordicsemi.com/power/w/opp/2/online-power-profiler-for-bluetooth-le

Aby sme znížili spotrebu zariadenia a celkové využitie BLE modulu, tak by sa mal mobil k zariadeniu pripájať iba na minimálnu dobu nevyhnutnú na to, aby sa medzi zariadením a mobilom preniesli všetky potrebné údaje, a neodkladne po prenose údajov by malo dôjsť k odpojeniu zariadení. Týmto spôsobom sa obmedzí komunikácia na minimum.

Ďalšiu minimalizáciu je možné dosiahnuť pomocou odporúčaní na optimalizáciu spotreby v dokumentácií nRF Connect SDK [10]. Najväčšiu úsporu spotreby vieme zaistiť tým, že v konfigurácií vypneme logovanie. Pri vývoji bude dôležité, aby sme mali dostupné logy a ľahšie sme tak odhalili chyby. Preto bude potrebné vytvoriť dve odlišné konfigurácie, jednu na vývoj a druhú, ktorá bude optimalizovaná na nasadenie do produkcie.

Medzi ďalšie rady, ktoré sú spomenuté v tejto dokumentácií patrí aj vypínanie nevyužívaných pinov a periférií na zariadení. To môžeme spraviť v overlay súbore tak, že nastavíme status periférie na disabled. V konfigurácií zariadenia môžeme ďalej povoliť modul správy napájania `CONFIG_PM_DEVICE`, ktorý informuje ovládače o tom, že sa systém pokúša prejsť do niektorého z úsporných režimov a tie tak môžu vykonať nevyhnutné operácie na zníženie spotreby. Pri implementácií by tak malo dosť k zmapovaniu všetkých nevyužitých periférií a modulov a ich vypnutiu, a zároveň povoleniu modulu na správu napájania.

Časť spotreby energie môžeme znížiť aj vypnutím napájania pamäte RAM, pre tú časť, ktorú nebudeme používať vrámci systému. Zároveň vieme optimalizovať vykonávanie kódu tak, aby sme používali čo najmenej pamäte RAM. Zníženie využitia RAM vieme docieliť viacerými spôsobmi, no ide najmä o to zakázať periférie, ktoré sú predvolené povolené, ak ich nebudeme využívať, ďalej môžeme zakázať vlastnosti Bluetooth, ktoré nepotrebujeme, ako napríklad šifrovanie a nakoniec ide aj o vyladenie veľkostí zásobníkov, ktoré potrebujeme pre jednotlivé vlákna a úlohy systému. [7, 9]

4.3 Mobilná aplikácia

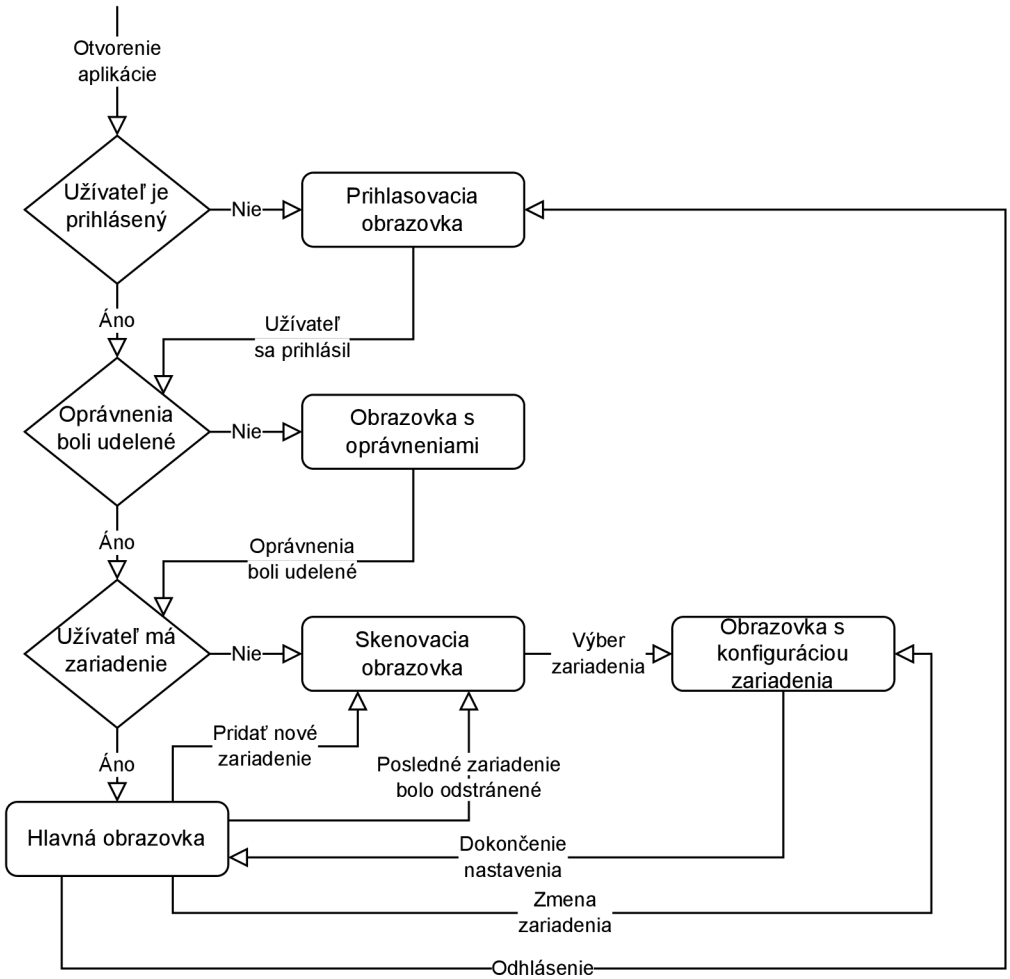
Pri vývoji mobilnej aplikácie sme sa rozhodli ísť cestou cross-platform vývoja, aby tak bolo možné zároveň možné vytvárať aplikáciu rovnako tak pre Android, ako aj iOS zariadenia. Ako framework sme si zvolili Flutter, pretože poskytuje mnoho doplnkov a rozsiahlu dokumentáciu. V mobilnej aplikácií sa budú vykonávať všetky prípadné použitia, narozdiel od vstavaného systému, ktorý bude údaje iba zaznamenávať, ukladať a ďalej odosielať do mobilného zariadenia.

Prenos dát sa bude uskutočňovať práve pomocou mobilnej aplikácie, vďaka ktorej bude možné zo vstavaného zariadenia získavať namerané údaje rovnako tak, ako bude možné toto zariadenie konfigurovať.

Nemenej dôležitou úlohou mobilnej aplikácie bude zobrazovanie a analýza nameraných hodnôt a to buď priamo číselnými údajmi za určité obdobie, ako aj grafické znázornenie trendu v grafe.

Mobilná aplikácia by mala zároveň ukladať namerané údaje do cloudu, kde k ním bude mať prístup iba prihlásený užívateľ. Užívateľ tak má možnosť sledovať spotrebu na rôznych zariadeniach, ktoré by však medzi sebou mali ostať vzájomne synchronizované. Užívateľovi by malo byť umožnené spracovávať viacero zariadení zároveň, pridávať/odoberať zariadenia, prepínať sa medzi nimi a to všetko v jednej prehľadnej aplikácií.

Aplikácia sa bude skladať z niekoľkých obrazoviek. Každá obrazovka bude plniť zvláštnu funkciu. Obrazovky je možné rozdeliť podľa ich funkcie na následovné: prihlasovacia obrazovka, obrazovka oprávnení, obrazovka na vyhľadávanie a pridávanie zariadení, obrazovka na konfiguráciu zariadenia a hlavná obrazovka na zobrazovanie údajov.



Obr. 4.3: Prechody medzi jednotlivými obrazovkami v aplikácií

Na obrázku 4.3 je možné vidieť ako by mali fungovať prechody medzi jednotlivými obrazovkami. Prvá obrazovka, ktorá sa používateľovi zobrazí po nainštalovaní aplikácie je prihlasovacia obrazovka. Užívateľ sa tu prihlási k účtu ku ktorému sa ďalej budú viazať zariadenia a údaje z nich. Po prihlásení užívateľ prejde na obrazovku, kde bude vyzvaný aby povolil využívanie Bluetooth pre aplikáciu. Bez tohto povolenia totiž nebude možné aplikáciu využívať. Následne by mala aplikácia overiť, či užívateľ už má zaregistrované nejaké zariadenie na svojom účte. Ak áno, prejde sa na hlavnú obrazovku, kde sa už následne budú zobrazovať údaje zo zariadenia. Ak nie, užívateľovi sa zobrazí skenovacia obrazovka, na ktorej môže vyhľadať a pridať nové zariadenie. Ak užívateľ vyberie konkrétne zariadenie, prechádza na obrazovku s konfiguráciou, kde je nutné nastaviť potrebné hodnoty pre správne fungovanie. Po konfigurácií sa užívateľ dostáva na hlavnú obrazovku.

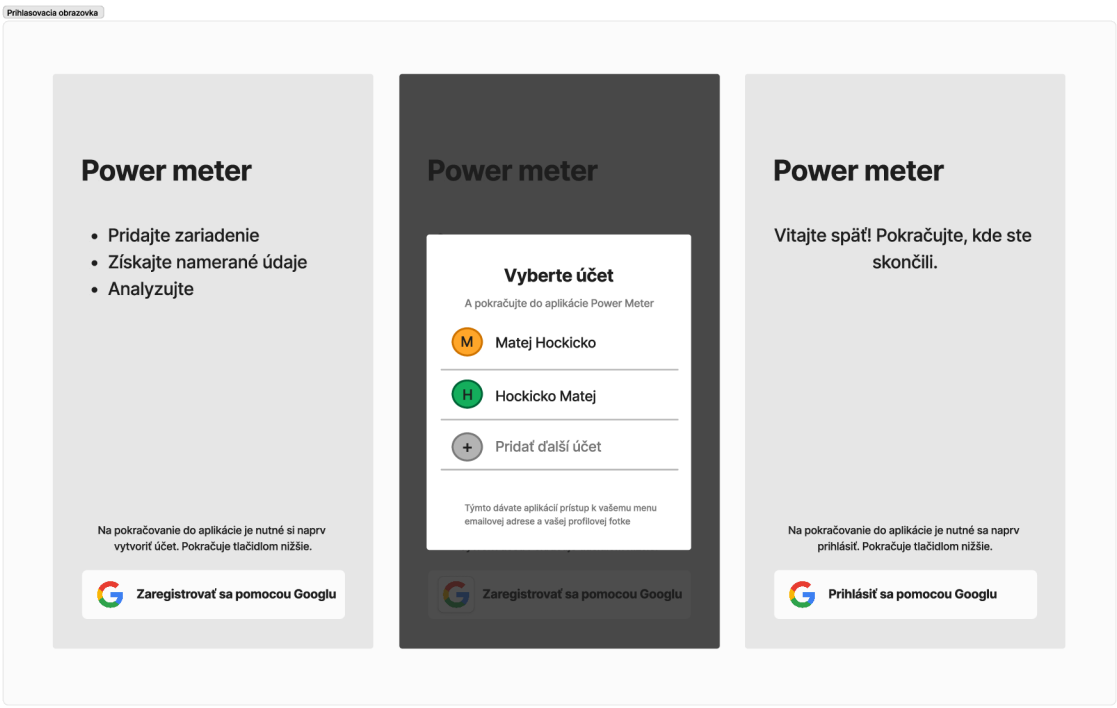
Z hlavnej obrazovky sa užívateľ vie dostať na konfiguračnú obrazovku, ak sa rozhodne zmeniť aktuálne zariadenie, poprípade na skenovaciu obrazovku, v prípade, že by sa rozhodol pridať nové zariadenie, alebo odstránil zariadenie a žiadne iné zariadenie, na ktoré

by sa bolo možné prepnúť, už na účte nemá. Užívateľovi bude umožnené aj odhlásenie, aby sa mohol mimo iného prihlásiť aj na iný účet. Táto akcia ho presmeruje naspäť na prihlasovaciu obrazovku.

Po opätovnom otvorení aplikácie by sa užívateľ mal už dostať priamo na hlavnú obrazovku. Aplikácia však bude aj tak overovať prihlásenie, oprávnenia a zariadenia, a v prípade zmeny užívateľa presmeruje na odpovedajúcu obrazovku.

4.3.1 Prihlasovacia obrazovka

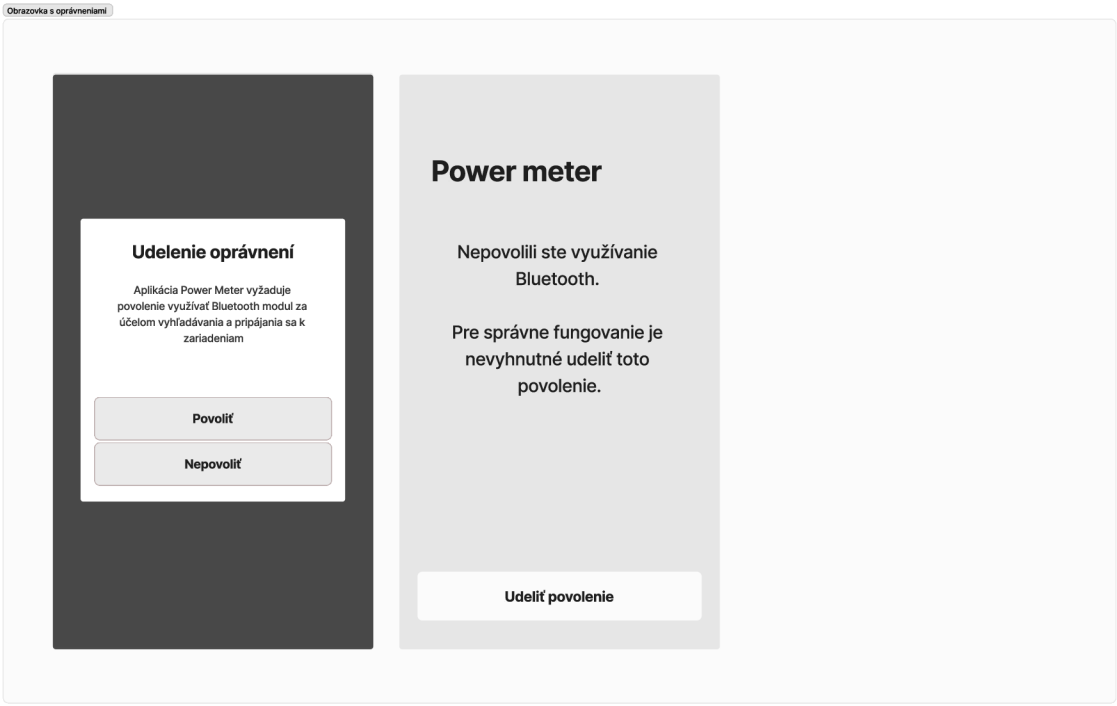
Aby bolo možné spravovať údaje pre užívateľa a ukladať ich v cloude je nutné aby mal užívateľ vytvorené konto, ktorým sa bude autentifikovať. Na tejto obrazovke by tak mal mať užívateľ na výber prihlásenie do niektorého z kont, ktorými je už prihlásený v zariadení. A to napríklad pomocou služby Google, ale aj iných služieb. Vizáž tejto obrazovky by mala závisieť od toho, či ju užívateľ vidí prvýkrát, a teda či sa registruje niektorým z účtov, alebo či bol odhlásený, a má sa teda znova prihlásiť do už pred tým použitého konta, poprípade vybrať iný účet. Jednotlivé varianty obrazovky a výber účtu je možné vidieť na obrázku 4.4



Obr. 4.4: Návrh prihlasovacej obrazovky

4.3.2 Obrazovka s oprávneniami

Táto obrazovka vyzýva užívateľa na udelenie oprávnení pre aplikáciu aby mohla využívať Bluetooth. Bez toho povolenia aplikácia fungovať nebude. Ak voľba ešte nebola uskutočnená, zobrazí sa prvá aplikácia s výzvou. V prípade, že užívateľ zamietol povolenie, zobrazí sa mu druhá obrazovka, ktorá ho nepustí do aplikácie, dokým neudelí povolenie. Užívateľovi už nie je možné zobraziť výzvu na udelenie oprávnení, keďže ich zamietol. Namiesto toho



Obr. 4.5: Návrh obrazovky s oprávneniami.

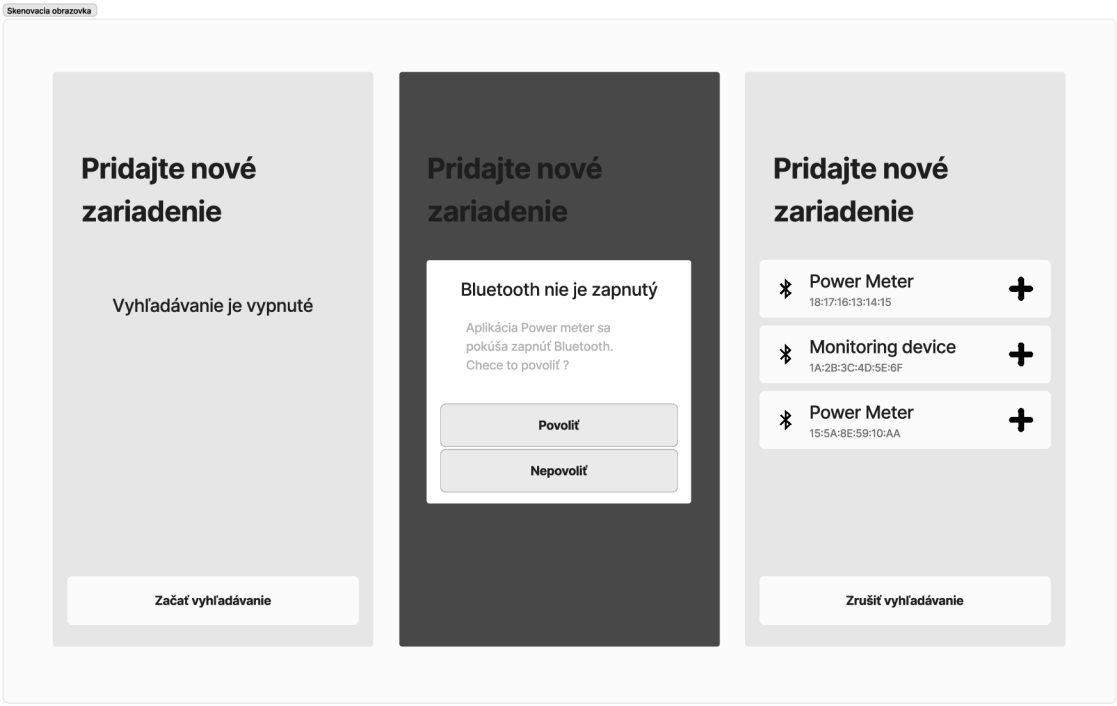
sa mu zobrazí tlačidlo, ktoré ho presmeruje priamo do nastavení aplikácie, kde môže svoje rozhodnutie zmeniť a udeliť povolenie. Návrh obrazovky je na obrázku 4.5.

4.3.3 Skenovacia obrazovka

Na tejto obrazovke užívateľ bude pridávať nové zariadenia. Na prvej obrazovke sa užívateľovi zobrazí možnosť začať vyhľadávanie. Ak užívateľ klikne na toto tlačidlo, mala by sa najprv vykonať kontrola, či je zapnutý bluetooth a či je vôbec možné spustiť skenovanie. Ak bluetooth nie je zapnutý, aplikácia by mala umožniť jednoduché zapnutie bluetooth priamo v aplikácií bez nutnosti opustenia aktuálnej obrazovky. V prípade, že užívateľ nepovolí zapnutie Bluetooth tak sa obrazovka nemení a stále bude ponúkať možnosť na začatie vyhľadávania.

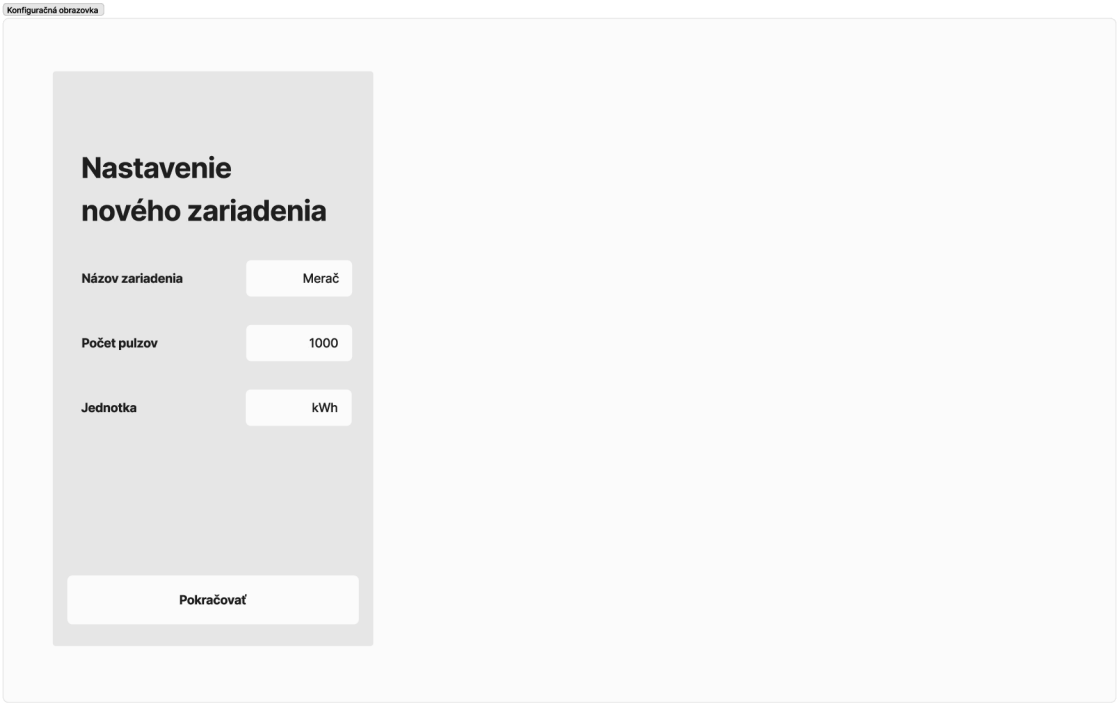
Po povolení zapnutia bluetooth alebo v prípade, že už bol bluetooth zapnutý sa zmení stav obrazovky. V strede obrazovky sa budú zobrazovať postupne novo nájdené zariadenia. Malo by ísť iba o zariadenia, ktoré sú určené na zaznamenávanie elektrickej spotreby a aplikácia by nemala umožňovať pripojenie k iným zariadeniam, ktoré neumožňujú takúto funkcionalitu. V zozname zariadení sa bude zobrazovať názov jednotlivých zariadení a ich MAC adresa. Kliknutím na konkrétne zariadenie užívateľ zvolí toto zariadenie. Aplikácia sa pokúsi o pripojenie a po úspechu prechádza na konfiguračnú obrazovku. V prípade neúspechu aplikácia ďalej pokračuje vo vyhľadávaní, užívateľa by ale mala informovať o chybe, napríklad formou widgetu toast.

Na obrazovke sa zároveň zmení aj spodné tlačidlo, ktoré prejde zo stavu začať vyhľadávanie do stavu prerušiť vyhľadávanie. Po stlačení sa preruší skenovanie a obrazovka prejde do svojho prvotného stavu. Na obrázku 4.6 je možné vidieť návrh obrazovky.



Obr. 4.6: Návrh obrazovky na skenovanie zariadení v okolí

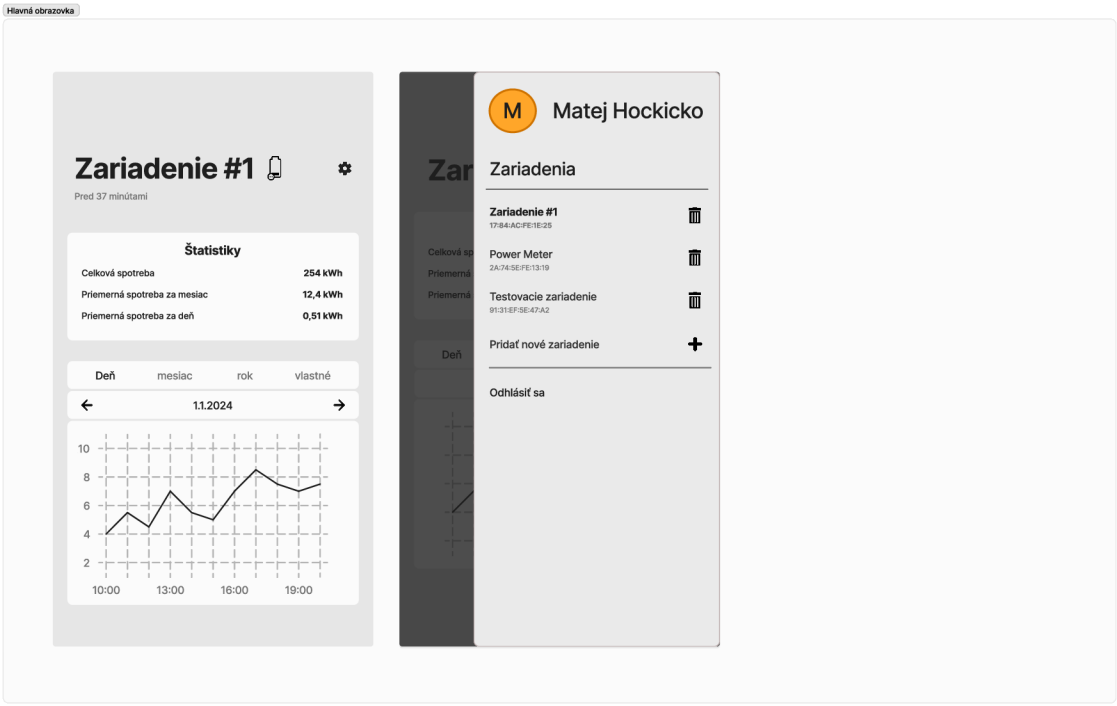
4.3.4 Konfiguračná obrazovka



Obr. 4.7: Návrh konfiguračnej obrazovky

Po tom, čo užívateľ zvolí zariadenie, sa aplikácia k nemu pokúsi pripojiť a po úspechu sa prechádza na konfiguračnú obrazovku. Tu je nutné vyplniť niekoľko polí. Pre správny výpočet a zobrazovanie štatistík spotreby elektrickej energie je nutné vyplniť počet pulzov a jednotku spotrebovanej energie. Tieto údaje je možné nájsť priamo na elektromere. Užívateľ má možnosť zmeniť aj názov zariadenia. Tento názov sa mu bude neskôr zobrazovať pre dané zariadenie v zozname zariadení a na hlavnej obrazovke. Po kliknutí na tlačidlo, ktoré sa nachádza v spodnej časti, aplikácia nakonfiguruje potrebné údaje, odošle potrebné informácie do zariadenia, ktoré od tohto momentu začína vykonávať meranie a zároveň uloží údaje o konfigurácií zariadenia na cloud, odkiaľ ich ďalej bude čerpať. Ukážka obrazovky je na obrázku 4.7.

4.3.5 Hlavná obrazovka



Obr. 4.8: Návrh hlavnej obrazovky

Väčšina interakcie s mobilnou aplikáciou sa bude odohrávať na jej hlavnej obrazovke. Tu bude užívateľ sledovať štatistiky, či už vo forme čísel alebo grafu, prepínať sa medzi jednotlivými zariadeniami, či prípadne upravovať konfiguráciu zariadení alebo pridávať ďalšie zariadenia.

Hlavná obrazovka bude pozostávať z jednoduchého rozhrania. Na vrchu sa bude zobrazovať názov aktuálne vybraného zariadenia. Hneď vedľa názvu zariadenia by sa malo zobrazovať prípadné upozornenie na slabú batériu. Celkom v pravo sa v tomto riadku ešte bude nachádzať možnosť na zmenu konfigurácie zariadenia. Pod týmto riadkom sa bude nachádzať informácia o tom, kedy došlo k poslednej aktualizácií údajov zo zariadenia.

Následujúca sekcia na obrazovke sa bude venovať hodnotám spotreby. Bude sa tu zobrazovať celková spotreba, ale zároveň aj priemerná spotreba za obdobie, či už za deň, týždeň alebo mesiac.

V ďalšej sekcii sa bude zobrazovať graf znázorňujúci spotrebu za určité obdobie. Na vrchu tejto sekcie si môže užívateľ zvoliť obdobie, za ktoré chce zobraziť spotrebu, ako deň, mesiac, rok, alebo si môže zvoliť vlastné obdobie. Pod týmto výberom sa zobrazí aktuálne zvolené obdobie. Po kliknutí na toto obdobie môže užívateľ zvoliť konkrétne obdobie. Zároveň je možné využiť tlačidlá po stranách, ktorými sa bude dať jednoducho prepínať medzi predchádzajúcim a následujúcim obdobím. V prípadne výberu vlastného obdobia môže užívateľ zvoliť počiatočný a koncový dátum, počas ktorých chce na grafe vidieť spotrebu. Užívateľ by mal mať možnosť prechádzať prstom po grafe a tak vidieť detailnú hodnotu v konkrétnom čase.

Táto obrazovka bude ešte zobrazovať bočné menu, kde bude možné vykonávať ďalšie akcie vrámci aplikácie. V tomto menu sa na vrchu bude zobrazovať účet pod ktorým je aktuálne užívateľ prihlásený. Ďalej sa pod ním zobrazí zoznam jednotlivých zariadení, ktoré má užívateľ vo svojom účte pridané. Aktuálne zariadenie, ktoré je zvolené bude zvýraznené od ostatných. Užívateľ bude mať možnosť prepínať sa medzi jednotlivými zariadeniami, odstraňovať zariadenia, alebo pridávať nové zariadenia. V spodnej časti tohto menu sa ešte zobrazí možnosť na odhlásenia z aktuálneho účtu. Vizualizáciu návrhu je možné vidieť na obrázku 4.8.

4.4 Vzdialené ukladanie dát

Potreba ukladania dát vyžaduje určitú službu, na ktorej si budú môcť jednotlivý užívatelia uchovávať svoje údaje, aby sa k nim vedeli dostať aj v prípade zmeny zariadenia. Táto služba by mala byť prepojená s mobilom, a to tak, aby bolo možné aplikáciu využívať aj v prípade aktuálnej nedostupnosti internetového pripojenia. Mobilná aplikácia by mala byť schopná ukladať údaje do lokálnej pamäte a v prípade obnovenia internetového spojenia synchronizovať údaje. Fungovanie aplikácie by nemalo byť závislé na neustálom internetovom spojení.

Výhodou cloudovej databázy bude, ak bude vedieť reagovať v reálnom čase. Pri synchronizácií údajov v jednom zariadení tak bude možné hneď vidieť aktuálne hodnoty aj na ostatných zariadeniach prihlásených pod rovnakým účtom.

V cloudovej databáze sa budú ukladať údaje o zariadeniach používateľa a zároveň aj namerané hodnoty. Aplikácia by mala byť schopná synchronizovať tieto údaje nezávisle od seba. Nechceme, aby pri každej aktualizácií zariadenia dochádzalo aj k sťahovaní všetkých nameraných hodnôt.

Jednotlivé zariadenia a ich údaje by sa mali nachádzať v DB pod konkrétnym užívateľom. S týmto súvisí tvorba užívateľských účtov, ktoré by mali byť prepojené aj s fungovaní databázy. Oprávnenia v databáze by mali byť nastavené tak, aby sa ku konkrétnemu zariadeniu vedel dostať len taký užívateľ, ktorý má na to právo. Teda taký, ktorý zariadenie pridal a nakonfiguroval. Rovnako tak údaje zo zariadenia by mal vidieť iba jeho vlastník.

Rozhodol som sa pre využívanie cloudových služieb Firebase od Googlu. Firebase poskytuje služby databázy v reálnom čase. Ide o NoSQL databázu, ktorá ukladá dáta vo formáte JSON. Na jednotlivé údaje sa odkazujeme pomocou ich cesty v hierarchii. Na najvyššej úrovni by sa v hierarchii mali nachádzať samotný užívatelia. Pod každým užívateľom budú následne uložené jeho údaje, ku ktorým bude pristupovať. Každý užívateľ bude mať oprávnenia iba na svoj záznam. Každý užívateľ bude identifikovaný svojim ID, ktoré bude kľúčovým záznamom. Ak bude užívateľ vytvárať žiadosť na dáta, tak na začiatku cesty sa vždy bude nachádzať jeho ID. Pre toto ID sa overí, že je pod ním užívateľ prihlásený a že má pridelené oprávnenia.

Pod každým užívateľom sa následne bude hierarchia deliť na 2 oblasti, a to na dáta a zariadenia. Údaje zo zariadení budú oddelené od konfigurácie zariadenia, aby sme pri aktualizácií údajov o konfigurácií zároveň nemuseli znova sťahovať všetky namerané údaje. V oboch oblastiach sa na vrchu budú nachádzať ID zariadení. Ako ID zariadení poslúži ich MAC adresa. V oblasti s nameranými hodnotami sa pod týmto zariadením budú nachádzať páry kľúčov a hodnôt. Ako kľúč poslúži časové razítko, pre ktoré bol vytvorený záznam. Podľa kľúčov tak bude možné ďalej filtrovať hodnoty do ľubovoľných časových oblastí. V druhej oblasti so zariadeniami sa bude ukladať ich konfigurácia. Konfigurácia bude opäť v tvare kľúč - hodnota. Kľúč bude predstavovať názov vlastnosti a k nemu bude konkrétna hodnota. Ukladať sa tu bude informácia o tom, ako sa zariadenie nazýva, či je aktívne a informácia o množstve pulzov na jednotku času.

Kapitola 5

Implementácia a testovanie vstavaného zariadenia

Počas implementácie riešenia bolo nevyhnutné postupne testovať funkčnosť a spoľahlivosť v jednotlivých fázach vývoja. Vývoj bol rozdelený na menšie celky, v ktorých vždy došlo k implementácií určitej funkčnosti. Jednotlivé oblasti boli otestované samostatne, a následne po úspešnom overení ešte došlo k otestovaniu celého systému ako celku.

Samotný vývoj prebiehal v niekoľkých fázach. Na najvyššej úrovni možno vývoj rozdeliť podla oblasti na vývoj vstavaného zariadenia a vývoj mobilnej aplikácie. Implementácia vstavaného zariadenia predchádzala mobilnej aplikácií, aby bolo neskôr možné pri vývoji mobilnej aplikácie využiť už čiastočne hotové hardvérové riešenie bez nutnosti simulovania daného zariadenia a dát. Neskorší vývoj však už prebiehal simultánne, a to najmä z dôvodu vyladenia parametrov systému na oboch stranách.

Ak sa pozrieme detailne na už spomínané dve oblasti vývoja, vieme ich ďalej rozčleniť na menšie samostatné celky. Oblasť vstavaného zariadenia môžme ďalej deliť na spracovávanie vstupu, funkčnú logiku spracovávania a ukladania dát, a komunikáciu s mobilným zariadením pomocou BLE. Takéto členenie je vidieť aj v rozdelení zdrojových kódov na jednotlivé súbory, ktoré spolu prepája a riadi hlavný program.

5.1 Vývoj vstavaného systému

Prvým ktorom pred samotným programovaním mikrokontroléru bolo vytvorenie hardvérového obvodu na zaznamenávanie optického vstupu. Obvod na zaznamenávanie optického vstupu, spomínaný v návrhu systému v sekcii 4.2.1, bol zostavený na nespájkovacom kontaktnom poli. Na simuláciu optického výstupu elektromera bola použitá LED dióda, ktorá bola zapojená do obvodu s tlačidlom. Na testovanie tohto obvodu bola ešte na výstup umiestnená ďalšia LED dióda, na základe ktorej bolo možné detektovať funkčnosť zapojenia. Narozdiel od rozoberaného zapojenia v návrhu, kde bola využitá LED dióda za účelom zaznamenávania signálov, je v tomto zapojení využitá priamo fotodióda. Počas testovania sa na jednom konci pomocou tlačila rozsvecovala LED dióda, ktorej svetlo bolo detekované pomocou obvodu. Na výstupe sa sledovala druhá dióda, ktorá signalizovala spracovanie vstupu.

Po úspešnom otestovaní funkčnosti obvodu s komparátorom nasledoval ďalší krok, kde bola LED dióda nahradená za mikrokontrolér. Testovaný obvod s komparátorom sa pripojil na VDD a GND pin na mikrokontroléri. Tak som dosiahlo to, že celý systém funguje na

jednej úrovni napätia a nemusí sa tak upravovať výstupné napätie komparátora, aby ho bolo možné priamo zaznamenávať. Na mikrokontroléri bežal jednoduchý kód, ktorého úlohou bolo detektovať zmeny na GPIO pine. Pri detekcii zmeny z logickej nuly na logickú jednotku sa mala vyvolať výnimka, ktorá ikrementovala čítač detekcií zmeny stavov a vypísala jeho aktuálnu hodnotu. Pri testovaní funkčnosti tohto zapojenia došlo k problému, ktorý sa pred tým pri testovaní bez mikrokontroléra nevyskytoval. Prerušenie sa v určitých chvíľach vyvolávalo aj bez stlačenia tlačidla a to dokonca v rade stoviek či tisícok udalostí za sekundu. Dôvodom týchto prerušení bolo nedostatočné odizolovanie fotodiódy, ktorá tak reagovala na zmeny v intenzite okolitého svetla. Tento problém sa vyriešil pridaním fyzickej zábrany, ktorá odtienila fotodiódu a LED diódu od vplyvov okolitého svetla.

Následne bolo potrebné otestovať registráciu prerušení manuálnymi stlačeniami tlačidla o rôznych dĺžkach trvania. Pri rýchlejších stlačeniach systém nedokázal spoľahlivo reagovať na všetky zmeny. Problém sa vyriešil po odstránení spätnoväzbového odporu, ktorý mal slúžiť na pridanie hysterzie. Použitý komparátor sa líšil od komparátoru použitého v obvode, z ktorého som vychádzal. Použitý komparátor mal väčšiu hysterziu, než pôvodný, preto došlo k vynechaniu tohto odporu. Po opätovnom otestovaní už systém dokázal správne reagovať na všetky manuálne vstupy.

Po manuálnom testovaní bolo na rade testovanie teoretickej schopnosti zaznamenávať jednotlivé pulzy z elektromeru pomocou počítačovej simulácie. Na to bolo možné využiť program TINA-TI[1] od spoločnosti Texas Instruments. Do tohto programu bol prenesený rovnaký obvod, ktorý už existoval na nespájkovacom kontaktnom poli. Do programu bolo potrebné pridať model komparátoru[2], aby tak bolo možné vykonať jeho simuláciu. LED dióda a fotodióda bola nahradená za generátor prúdu, ktorého parametre boli nastavil tak, aby jeho priebeh odpovedal maximálnej možnej frekvencii blikania optického výstupu na elektromere podľa technickej normy. Zo simulácie vyplynulo, že daný obvod nie je schopný správne reagovať na tak rýchle zmeny na vstupe. Hodnoty odporov bolo potrebné znížiť tak, aby bolo možné v simulácií dosiahnuť požadovaného výsledku.

V obvode následne k výmene odporov podľa výsledkov v simulácií. Tento obvod bol ešte ďalej otestovaný za pomoci ďalšieho mikrokontroléra, aby tak bolo možné presne ovládať časové úseky, počas ktorých bude vstupná LED dióda svietiť a kedy svietiť nebude. Na testovanie sa využil modul ESP32 s rozhraním RMT, ktorého primárna úloha je generovanie a dekódovanie infračervených signálov. Rozhranie je schopné generovať impulzy podľa predom definovaného priebehu. Táto funkcionalita sa využila na to, aby sa stav diódy menil presne s periódou 400µs, pričom pracovný cyklus bol nastavený na 50%. Testovalo sa niekoľko tisíc opakovaní a sledoval sa výstup mikrokontroléru, na ktorom bolo možné overiť očakávaný počet zaznamenaných impulzov. Testovalo sa nie len to, či dochádza k zaznamenávaniu správneho počtu impulzov, ale zároveň aj to, či pri chýbajúcich impulzoch ostáva hodnota na nule. Spomínaná hodnota 400µs vychádza z minimálnej dĺžky periódy signálu, ktorý je možné generovať na optickom výstupe elektromera.

Týmto spôsobom bolo otestované, že je zariadenie schopné zaznamenávať maximálnu možnú frekvenciu vstupov. Aby sa však overilo fungovanie obvodu na reálnejšom vstupe, rozhodol som sa upraviť kód, tak, že v ňom ostala pevná šírku pulzu zapnutej LED diódy a náhodne sa menila šírka času LED diódy vo vypnutom stave. Podobným spôsobom ako v predchádzajúcom prípade sa overila funkčnosť vstavaného zariadenia.

[1] https://www.ti.com/tool/TINA-TI
[2] https://www.ti.com/lit/zip/sbom853

5.1.1 Ukladanie nameraných hodnôt do pamäte flash

V návrhu som sa rozhodol, že počet pulzov sa bude sčítavať po dobu 5 minút, a následne sa uloží. Na meranie týchto časových úsekov sa využíva časovač. Ten každých 5 minút vyvolá prerušenie, ktoré zaistí, že sa do pracovnej fronty zaradí funkcia na ukladanie hodnôt. Vykonávanie samotného ukladania sa odohráva v pracovnej fronte z dôvodu, aby toto prerušenie neblokovalo prioritnejšie vlákno, ktoré nutne potrebuje reagovať na prerušenia od obvodu na zaznamenávanie optického výstupu. Zápis do pamäte je zdĺhavejšia operácia, ktorá sa môže vykonávať aj v menej prioritnom vlákne. Ako bolo ukázané v návrhu, na ukladanie hodnôt postačujú 2B. Do pamäte je však možno zapísať najmenej 4B, preto hodnoty zapisujem naraz po dvojiciach.

Funkcie, ktoré slúžia konfiguráciu obsluhu prerušenia časovača sa nachádzajú v súbore `rtc.c`. Hlavná funkcia s názvom `start_counter` nakonfiguruje a spustí časovač. Táto funkcia očakáva ako argument funkciu, ktorá sa bude volať pri vzniku prerušenia. Na začiatok definujeme zariadenie `counter_dev`, ktoré využíva modul RTC. Toto zariadenie následne spustíme zavolaním funkcie `counter_start`. Ďalej je potreba nastaviť konfiguráciu časovača. Konfigurácia časovača sa nachádza v globálnej premennej `alarm_cfg`. Do tejto premennej nastavíme zariadenie a čas po akom chceme vyvolať prerušenie. Tento čas je definovaný ako makro v súbore `const.h` pod názvom `COUNTER_SECONDS`. Ďalej nastavíme funkciu, ktorá sa vyvolá po prerušení, a to na funkciu `alarm_cb`, a do užívateľských dát ešte predáme funkciu, ktorá bola vložená do atribútu. Nakoniec vo funkcii `start_counter` zavolaním funkcie `set_alarm`. V tejto funkcii dôjde zapnutiu prerušenia zavolaním funkcie `counter_set_channel_alarm`, ktorej predáme zariadenie, ID kanálu a konfiguráciu časovača. Po vyvolaní prerušenia sa zavolá funkcia `alarm_cb`. Na začiatok táto funkcia opätovne nastaví ďalšie prerušenie pomocou funkcie `set_alarm` a následne zavolá funkciu, ktorá bola predaná v užívateľských údajoch pri konfigurácií počítadla.

Na ukladanie a spracovávanie údajov v pamäti slúžia funkcie v súbore `data.c`. Na začiatok mám v tomto súbore definované konštanty a štruktúry, ktoré sú ďalej potrebné pre prácu s úložiskom. Za nimi mám ďalej definované premenné, ktoré využívam vrámci viacerých funkcií ako napríklad adresu, na ktorej začínajú záznamy v pamäti, ďalej miesto, kam sa majú ukladať aktuálne záznamy ale aj počet zaznamenaných pulzov. Prvá funkcia v tomto súbore má názov `init_nvs`. Tá slúži na inicializáciu knižnice NVS. Táto knižnica ukladá do pamäte hodnoty v podobe dvojíc kľúč - hodnota. Pamäť vyčlenená tejto knižnici tvorí kruhový buffer, do ktorého sa postupne ukladajú údaje. V prípade, že dôjdeme nakoniec, tak sa hodnoty ukladajú opäť od začiatku. S týmto spôsobom ukladania je spojená réžia 8B pre každú uloženú hodnotu. Táto knižnica tak nie je vhodná na ukladanie nameraných hodnôt, pretože by využívala príliš veľa miesta, ale hodí sa na ukladanie premenných, ktoré sa tak často nemenia a je ich potrebné uchovávať aj po vypnutí zariadenia ako napríklad počiatok hodnôt v pamäti a časové razítko označujúce čas prvej uloženej hodnoty. Na fingovanie tejto knižnice je v pamäti vyhradené miesto o veľkosti 2 sektorov.

Na ukladanie nameraných hodnôt do pamäte sa využíva priamy prístup do pamäte pomocou flash map API. Toto API sa inicializuje a konfiguruje vo funkcii `init_fa`. V tejto funkcii otvárame oddiely pamäte, ktoré sú vyhradené na ukladanie počtu pulzov a na uloženie informácie o počte uložených záznamov. Po otvorení oddielov sa ešte inicializujú hodnoty počiatku dát v pamäti a počet záznamov.

Pri zaregistrovaní prerušenia od počítadla sa volá funkcia `counter_callback`. Tá nespracováva prerušenie priamo v prioritnom vlákne, aby sa neblokovali dôležitejšie úkony, ale naplánuje prácu do poradia. Práca, ktorá sa vykoná sa nachádza vo funkcii `counter_work`.

V tejto funkcii sa uloží aktuálny počet pulzov a počítadlo sa vynuluje. Počet pulzov sa uloží do záznamu pomocou funkcie `add_val_to_rec`. Ak sa kapacita záznamu naplnila, tak sa sa ďalej zavolá funkcia `save_record`. Táto funkcia vykoná zápis záznamu do flash pamäte na pozíciu `offset` a upraví posun ukazateľa do pamäte. Nakoniec sa vykoná ešte funkcia `add_offset`, ktorá ešte zapíše posun do pamäte. Táto funkcia postupne nuluje jednotlivé bity v pamäti, pričom každý bit udáva zápis jedného záznamu do pamäte. Zápis offsetu do pamäte sa využíva na to, aby po reštarte zariadenia bolo možné zistiť, kde končia záznamy a následne prečítať hodnoty, ktoré sa do pamäte uložili pred reštartom zariadenia.

Prerušenie na GPIO pine, ktorý reaguje na optický výstup elektromera, vyvolá funkciu `pulse_callback`. Táto funkcia len jednoducho inkrementuje hodnotu počítadla a ukončí sa.

Dvojice funkcií `read_timestamp`, `save_timestamp` a `read_start`, `save_start` slúžia na získavanie a ukladanie premenných s rovnakým názvom do NVS pamäte zariadenia.

Funkcia `read_offset` slúži na čítanie hodnoty offset z pamäte zariadenia. Táto funkcia postupne prechádza pamäť a spočítava počet núl, ktoré udávajú počet záznamov uložených v pamäti. Následne táto vráti počet bajtov, o ktoré je posunutý koniec záznamov v pamäti oproti začiatku. Na výmaz zmien bitov v pamäti a opätovné počítanie offsetu od nuly slúži funkcia `clear_offset`.

Poslednou funkciou je `records_count`. Tá slúži na výpočet počtu záznamov, ktoré sú aktuálne uložené v pamäti.

5.1.2 BLE

Na overenie základnej funkčnosti bola počas vývoja využívaná aplikácia od firmy Nordic Semiconductor s názvom nRF Connect. Pomocou tento aplikácie je možné overiť, že zariadenie je viditeľné, má správnu konfiguráciu a že sa k nemu je možné pripojiť. Zároveň je možné skrze aplikáciu získať zoznam dostupných služieb, odoslať potrebné informácie, či stiahnuť dáta. Počas celej doby komunikácie je možné v aplikácií prezerať jednotlivé správy, ktoré si medzi sebou vstavané zariadenie a mobil vymieňali, takže v prípade chyby bolo jednoduché zistiť, v čom bol problém.

Konfigurácia Bluetooth Low Energy a obslužné funkcie sa nachádzajú v samostatnom súbore `ble.c`. Na začiatku súboru sú deklarované funkcie, ktorých úlohou je spracovávať a poskytovať údaje pre charakteristiky. Jednotlivé Bluetooth charakteristiky sú postupne ďalej definované za pomoci makier. Ďalej sú v súbore definované štruktúry `bt_data` s názvami `ad` a `sd`. Štruktúra `ad` definuje údaje, ktoré zariadenie vysiela pri oznamovaní na sieti a `sd` definuje odpoveď vstavaného zariadenia na skenovanie, kde informuje o poskytovanej službe. Ďalej sa tu ešte nachádza štruktúra `bt_le_adv_param`, ktorá udáva, že k zariadeniu sa je možné pripojiť a nastavuje interval oznamovania na sieti na dve sekundy, aby sa tak šetrila batéria.

Makro `BT_CONN_CB_DEFINE` nastavuje funkcie, ktoré sa vykonajú po tom, čo sa zariadenie pripojí alebo odpojí. Po pripojení sa volá funkcia `on_connected`, ktorá uloží aktuálne spojenie do premennej aby s ním bolo možné ďalej pracovať a vyvolá funkciu `update_phy`, ktorá upraví parametre spojenia na využívanie režimu 2M. Po odpojení funkcia `on_disconnected` nastaví aktuálne spojenie na hodnotu `NULL`.

Konfigurácia Bluetooth prebieha pomocou funkcie `config_ble`. Tá si nastaví callback funkcie, ktoré bude ďalej využívať ak dôjde ku konfigurácií zariadenia, alebo ak bude potrebné získať údaje zo zariadenia a poslať ich cez Bluetooth do mobilu. Povolí sa Bluetooth

a začne sa vysielanie zariadenia na sieti pomocou funkcie `bt_le_adv_start`, do ktorej predáme skôr definované štruktúry.

Do zariadenia je možné zapísať časové razítko, čo vyvolá funkciu `set_timestamp`. V tomto momente zariadenie overuje, či sa zariadenie nespustilo po prvý krát alebo nereštartovalo. V tomto prípade volá callback funkciu `setup_cp`, ktorá pripraví zariadenie na zaznamenávanie pulzov. Časové razítko sa uloží a funkcia končí. Pri pokuse o prečítanie časového razítka sa zavolá funkcia `get_timestamp`, prečíta sa hodnota z pamäte a odošle sa do mobilu.

Zo zariadenia je zároveň možné zistiť, či nedošlo k reštartu zariadenia a to tak, že mobil prečíta charakteristiku, kde o tom zariadenie informuje. Táto charakteristika pri pokuse o prečítanie zavolá funkciu `get_restart_flag`, ktorá vráti požadovanú hodnotu.

Najdôležitejšia charakteristika je ale tá, pomocou ktorej je možné stiahnuť zo zariadenia údaje. Táto charakteristika povoľuje prihlásenie sa k odberu notifikácií. Pri zmene stavu notifikácii sa volá funkcia `get_data_cfg_changed`. Ta po prihlásení mobilu k notifikáciam ďalej volá funkciu `on_notification_enabled`, v ktorej sa pridá do porodia práca na vykonanie. Práca na vykonanie je vo forme funkcie `send_all_data_work`. Tá postupne do jednotlivých bufferov na odoslanie pripraví za pomoci funkcie `prepare_data_for_nitification` všetky údaje, ktoré sú uložené v zariadení. Príprava prebieha zistením počtu záznamov ktoré je potrebné preniesť. Následne sa zistí maximálny počet záznamov, ktoré je možné preniesť v jednom pakete a pomocou callback funkcie `record_cb` sa potrebné údaje vyčítajú z pamäte, vyplnia do paketu a odošlú pomocou funkcie `bt_gatt_notify_cb`. Ak sa už pripravili a odoslali všetky údaje, tak sa na záver ešte odošle prázdny paket, na ktorý čaká mobil, aby sa odhlásil z odberu a ukončil spojenie. Po potvrdení prijatia posledného paketu sa ešte vyvolá funkcia `write_start`, ktorá prepíše potrebné údaje ako časové razítko, adresu začiatku údajov v pamäti a vymaže offset.

5.1.3 Hlavný beh programu

Vykonávanie programu vstavaného zariadenia začína v programe `main.c`. Po spustení aplikácie sa postupne volajú funkcie, ktoré nastavia a spustia potrebné moduly. V prvom rade sa nakonfiguruje a zapne LED. Definície funkcií `configure_led` a `led_on` je možné nájsť v súbore `led.c`. V tomto momente LED svieti, aby tak užívateľovi signalizovala, že zariadenie je funkčné a pripravené na konfiguráciu. Ďalej sa nastavia callback funkcie pre Bluetooth a ten sa zapne. Inicializujú sa aj knižnice na prístup do pamäti a pripraví sa aj nastavenie prerušenia na GPIO pine vo funkcii `input_setup`, no zatiaľ sa nespustí. V tomto momente končí práca hlavného programu a ďalšie operácie sa už vykonávanú v samostatných vláknach. Väčšinu času je už zariadenie ďalej nečinné a reaguje už iba na podnety z vonku. Jedným z takýchto podnetov je pripojenie mobilného zariadenia a konfigurácia, ktorá vyvolá funkciu `monitor_start`. Táto funkcia vypne LED, spustí časovač, uloží si časové razítko a spustí zaznamenávanie prerušení. Od tohto momentu zariadenie monitoruje optický výstup elektromera a je pripravené odosielať namerané údaje do mobilu ihneď po jeho pripojení.

5.1.4 Popis hardvéru a konfigurácia zariadenia

Hardverové moduly a komponenty pripojené k zariadeniu je nutné nakonfigurovať v súbore overlay. Ako základ sa využíva dts súbor pre vývojovú dosku nrf52, na ktorú bol vyvíjaný software. Následne sú vo vlastnom overlay súbore nadefinované hardverové komponenty ako pripojená LED dióda alebo GPIO pin, na ktorom sa budú zaznamenávať prerušenia.

Ďalej sa tu vypínajú nepotrebné moduly ako napríklad ADC. V neposlednom rade sa tu vymedzujú aj oddiely v pamäti na zápis programu, ukladanie nastavení a hodnôt.

Ďalšia konfigurácia sa ešte odohráva v súboroch `prj.conf` pre vývojovú verziu programu a `prj_release.conf` pre distribučnú verziu. V konfigurácií je možné vidieť povolenie všetkých potrebných knižníc a modulov ako napríklad BT modul, povolenie FLASH pamäte, nastavenie názvu zariadenia, povolenie logovania ale aj úprava niektorých veľkostí registrov. Na správne využívanie externého kryštálu je potrebné nakonfigurovať zdroj hodín pomocou konfigurácie `CONFIG_CLOCK_CONTROL_NRF_K32SRC_XTAL`. Tým určíme externý kryštál ako zdroj hodín. Pomocou konfigurácie `CONFIG_CLOCK_CONTROL_NRF_K32SRC_20PPM` definujeme presnosť týchto hodín. Pod touto konfiguráciou sa nachádza povolenie DCDC regulátora na vstavanom zariadení, ktorý taktiež pomáha zo znížení spotreby batérie. Na záver konfigurácie je ešte povolené riadenie spotreby, ktoré v nečinnosti procesoru vypína nepotrebné moduly na zariadení.

Konfigurácia `prj_release` sa od vývojovej líši v tom, že pri nej už dbáme na spotrebu zariadenia. Preto je možné vidieť rozdiely, kde v tejto konfigurácií veci ako logovanie, konfiguráciu konzole, seriálového rozhrania, ale aj niektoré funkcionality zariadenia vypíname, aby sa tak znížila výsledná veľkosť firmwaru, ale najmä konečná spotreba zariadenia.

5.2 Návrh a vyhotovenie PCB

Aby som zariadenie otestoval v praxi a viac sa priblížil hotovému produktu, rozhodol som sa navrhnutý obvod preniesť z nespájkovacieho kontaktného poľa na PCB. Tu som mal na výber dve možnosti. Buď som si mohol navrhnúť celé zariadenie sám s využitím mikrokontroléru nrf52832, alebo využiť už hotový modul, ku ktorému by som ďalej napojil komponenty pre moju konkrétu aplikáciu. V prvom prípade by som sa nevyhol problémom s návrhom a ladením antény pre Bluetooth, preto som sa rozhodol ísť druhou cestou, a zvolil som modul BT832. Tento modul v sebe zahŕňa mikrokontrolér NRF52832 aj s komponentami nevyhnutnými na správne fungovanie, ako napríklad kryštál, a zároveň sa na doštičke nachádza aj vyladený obvod pre anténu so samotnou anténou na PCB.

K tomuto modulu som ďalej potreboval doplniť obvod na zaznamenávanie optického výstupu a pár ďalších komponent, ako napríklad tlačidlo na resetovanie zariadenia, port na naprogramovanie zariadenia, LED diódu na signalizáciu a testovanie funkčnosti, a v neposlednom rade aj 32kHz kryštál.

Do schémy zapojenia som postupne pridával potrebné komponenty na požadované fungovanie zariadenia, pričom som vychádzal najmä z referenčného zapojenia modulu BT832 [19]. Časť obvodu na zaznamenávanie signálov a hodnoty jednotlivých rezistorov som volil podľa výsledkov teoretického aj praktického testovania obvodu na nespájkovacom kontaktnom poli. Niektoré komponenty som najskôr nepripájal na GPIO piny, respektíve som ich pripojil na dočasné piny a toto zapojenie neskôr menil, keďže som ešte nevedel, ako budú umiestnené, a ku ktorým pinom ich bude najlepšie pripojiť.

Pri napájaní kryštálu je nutné používať správne kondenzátory. Ich hodnotu som vypočítal pomocou vzorca v dokumentácií k mikrokontroléru NRF52832 na strane 107 [26].

$$CL = \frac{C_1' \cdot C_2'}{C_1' + C_2'} \tag{5.1}$$

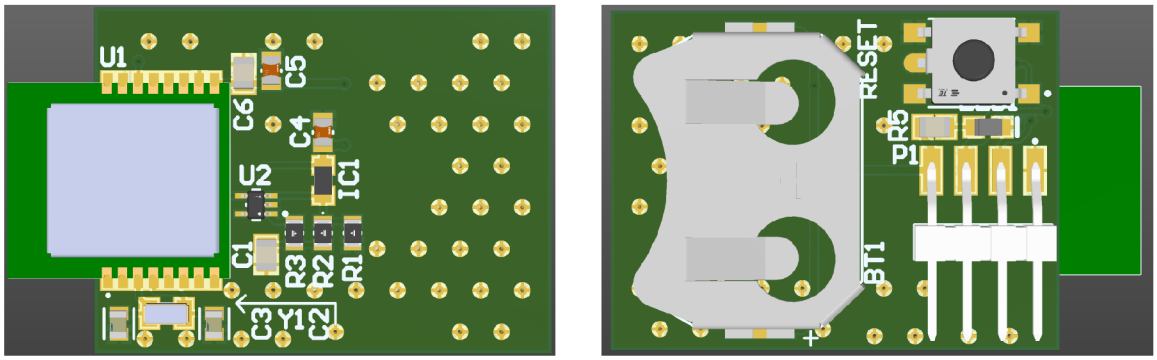
$$C_1' = C_1 + C_{pcb1} + C_{pin} \tag{5.2}$$

$$C_2' = C_2 + C_{pcb2} + C_{pin} \tag{5.3}$$

Výsledná hodnota CL v vzorci (5.1) značí záťažovú kapacitu. Túto kapacitu vyčítame z dokumentácie kryštálu. Zvolil som kryštál, ktorý má kapacitu 7pF. Kapacitu C_1 a C_2 vo vzorcoch (5.2) a (5.3) sa snažíme spočítať. Ide o kapacitu, ktoré by mali mať kondenzátory. Tieto kapacity by mali byť rovnaké. C_{pcb1} a C_{pcb2} značia kapacitu trás k jednotlivým spojom kryštálu. Túto kapacitu som ignoroval. Kapacita C_{pin} udáva, akú kapacitu majú piny, na ktoré budeme napájať kryštál. Podľa dokumentácie na strane 112 [26] je táto kapacita 7pF. Kapacita pre kondenzátory ku kryštálu tak z týchto vzorcov vychádza na 10pF.

Na pripojenie reset tlačidla som využil GPIO pin, ktorého funkciu je možné nakonfigurovať na reset zariadenia. Tento pin som zapojil na tlačidlo, ku ktorému som z druhej strany napojil zem. Ďalšie súčiastky neboli potrebné, pretože sa v dokumentácií k modulu BT832 môžeme dočítať, že tento modul už vo vnútri obsahuje interný RC resetovací obvod. Podobne tak nájdeme rovnaké zapojenie na schéme ukážkového zapojenia.

Ďalším krokom v návrhu bolo umiestnenie komponentov na PCB a vytvorenie trás pre obvody. Pri umiestnení modulu BT832 som sa opieral o dokumentáciu, ktorá odporúča modul umiestniť tak, aby anténa presahovala hranu PCB o najmenej 6mm [19]. Všetky komponenty potrebné na snímanie optického výstupu som sa rozhodol umiestniť na spodnú stranu dosky. Približne v strede dosky by sa mala nachádzať dióda, aby tak došlo k obmedzeniu prípadných odrazov okolitého svetla po okrajoch dosky. Na zadnú stranu dosky som sa rozhodol umiestniť všetky komponenty, ktoré vyžadujú pristúp a nemôžu tak byť na strane, ktorá bude v kontakte s elektromerom. Ide o držiak na batériu, tlačidlo na reštart zariadenia, LED dióda a samozrejme vstup na naprogramovanie zariadenia. Pri umiestňovaní komponentov som sa snažil o to, aby mala výsledná PCB doska čo možno najmenšie rozmery. Najväčšiu rolu v tomto zohrával držiak na batériu, ktorý som pri voľbe komponentov vyberal tak, aby mal čo najmenšiu plochu a zbytočne nepridával na rozmeroch zariadenia, keďže by šlo o zbytočný a nevyužitý priestor.



Obr. 5.1: Predná a zadná strana 3D modelu výslednej PCB dosky.

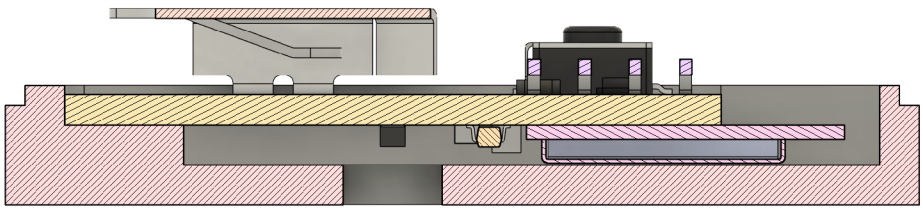
Komponenty som volil tak, aby ich puzdro bolo aspoň o veľkosti 0805, alebo najväčšie možné z dostupných variant, aby tak bolo možné dosku osadiť ručne.

Po umiestnení komponentov, ktoré som zatiaľ nemal pripojené som podľa ich pozície vybral najbližší možný pin, na ktorý ich bolo možné pripojiť a toto zapojenie som najskôr upravil v schéme, a neskôr preniesol aj na PCB návrh. Výsledný návrh PCB je možné vidieť na obrázku 5.1.

5.3 Návrh a tlač 3D modelu krabičky

Pri návrhu 3D modelu krabičky som dbal na jednoduchosť a funkčnosť výsledného riešenia. Pre uľahčenie inštalácie zariadenia som sa rozhodol krabičku rozdeliť na 2 časti, ktoré budú spolu tvoriť jednotný celok. Každá časť krabičky bude mať svoje špecifiká. Spodná časť krabičky musí obsahovať otvor, cez ktorý bude možné snímať optický výstup. Spodná strana by mala byť hladká, aby tak umožňovala dobré dosadnutie na povrch elektromeru, a nevytvárala zbytočne možnosť ovplyvňovať funkčnosť senzora okolitým svetlom. Táto strana by sa mala dať jednoducho pripevniť na povrch elektromeru, napríklad pomocou obojstrannej lepiacej pásky, a to tak, aby táto páska pokrývala okolitý priestor medzi senzorom a okolitým svetom. Už spomínaný otvor bude slúžiť aj na jednoduchú inštaláciu, keďže podľa neho bude možné jednoducho zarovnať optický výstup elektromera so senzorom na PCB. po pripevnení a zarovnaní spodnej časti krabičky by táto časť mala umožňovať jednoduché vloženie dosky plošných spojov, na túto časť a jej bezpečné upevnenie na mieste. Vrchná časť krabičky by mala byť ľahko odnímateľná, aby bol k zariadeniu ľahký prístup, na prípadnú reštart pomocou tlačidla, či výmenu batérie. Celá krabička by mala tvoriť jednotný celok, ktorý by okrem otvoru na snímanie optického výstupu nemal obsahovať iné nežiadúce otvory, cez ktoré by sa do vnútra mohlo dostať svetlo a ovplyvniť tak zaznamenávanie impulzov.

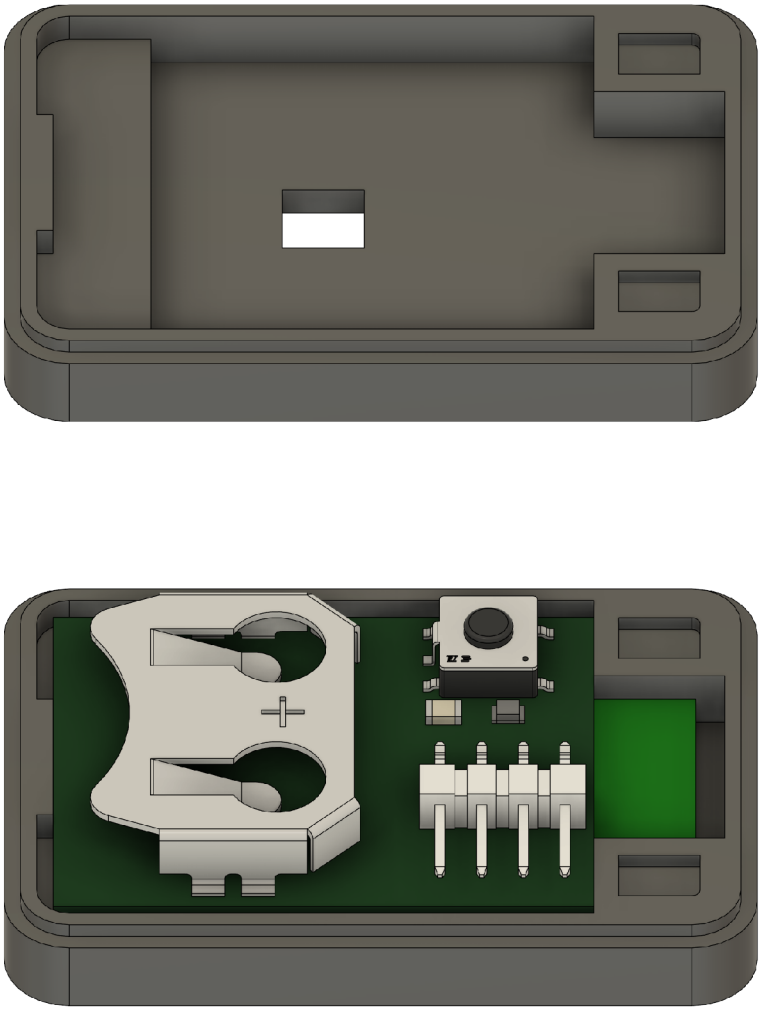
Návrh krabičky som zhotovoval v programe Autodesk Fusion. Z programu Altium designer som si vyexportoval 3D model PCB, ktorý som importoval do Fusion aby som mal podklad, na základe ktorého môžem vytvoriť 3D model krabičky. Najprv som vytvoril spodnú časť, tak aby sa do nej vošla doštička, a aby mala priestor zo všetkých strán. Spodnú hranu krabičky som vytvoril o hrúbke 2mm. Okraje hrán som zaoblil polomerom 4mm. Po okraji som vytiahol hranu o hrúbke 2mm. Hranu som následne rozdelil na vnútornú a vonkajšiu presne na polovicu. Vnútornú hranu som nastavil o 1mm vyššie. K tejto hrane príde protikus, ktorý uzavrie krabičku. Výšku hrany som volil tak, aby pri otvorenej vrchnej časti bolo možné vyberať a vkladať batériu do zariadenia. V mieste, kde sa nachádza fotodióda som vytvoril otvor na optický vstup. Rozmer otvoru je $5 * 5$mm.



Obr. 5.2: Prierez 3D modelom krabičky a PCB. Na obrázku je možné vidieť podstavec, ktorý slúži ako opora pre PCB, aby sa doštička nachádzala vo vodorovnej polohe. Na priereze je zároveň možné vidieť umiestnenie otvoru pre optický vstup, nad ktorého stredom sa nachádza fotodióda.

Keďže spodná strana PCB nemá rovnakú výšku vo všetkých miestach, rozhodol som sa ešte pridať podstavec v mieste, kde sa nenachádzajú žiadne komponenty na PCB, aby tak PCB dosadlo rovnobežne na spodnú stranu krabičky. Na jednej strane sa bude PCB opierať o modul BT832 a na druhej strane sa bude nachádzať podporná štruktúra. Ako by to malo vyzerať v praxi je možné vidieť na obrázku 5.2. Po stranách, kde do priestoru PCB

vystupuje anténa som pridal štruktúru, ktorá bude slúžiť na uchytenie PCB na mieste. Rovnako tak som na protiľahlú stranu pridal výstupok, ktorý taktiež pomôže s usadením PCB na mieste. Do štruktúr vedľa antény som ešte vytvoril diery, do ktorých prídu výstupky na vrchnej časti krabičky. Model spodnej časti krabičky je možné vidieť na obrázku 5.3.



Obr. 5.3: Model 3D krabičky spodnej časti krabičky bez/s vloženým PCB. Na modeli je možné vidieť hranu po obvode krabičky, na ktorú sa bude umiestňovať vrchná časť, otvor na fotodiódu, podstavec pre PCB, ako aj miesta na uchytenie PCB na mieste a diery na kolíky, do ktorých sa bude umiestňovať vrchná časť krabičky.

Obdobným spôsobom som ešte vytvoril vrchnú časť takým spôsobom, aby zapadala do spodnej. Zo spodnej časti som okopíroval plochy a následne som im pomocou nástroja extrude pridal tretí rozmer. Výšku som upravil tak, aby vo vrchnej časti bolo dostatok priestoru na komponenty nachádzajúce sa na PCB. Model vrchnej časti krabičky je možné vidieť na obrázku 5.4.

Obr. 5.4: Model 3D krabičky vrchnej časti krabičky. Vrchná časť má narozdiel od spodnej časti vyvýšenú vonkajšiu hranu a narozdiel od dier má naopak kolíky, ktoré zapadnú do týchto dier na ukotvenie vrchnej časti na mieste.

Takto vytvorenú krabičku som nechal vytlačiť. Časti, ktoré do seba mali zapadnúť však boli príliš natesno. Následne som upravil obvod hrany spodnej časti krabičky a diery na kolíky tak, aby medzi nimi a protikusom bola medzera o veľkosti $0,2$mm. Túto verziu som znova vytlačil, no tento krát boli časti krabičky príliš volné. Ako najlepšia sa ukázala verzia, v ktorej som upravil iba veľkosť dier na kolíky o $0,2$mm, a obvodovú hranu som nechal na pôvodnom nastavení.

Kapitola 6

Implementácia a testovanie mobilnej aplikácie

Vývoj mobilnej aplikácie nastal v momente, kedy bolo vstavané zariadenie z väčšej miery funkčné a dokázalo nadväzovať spojenie a prijímať/odosielať potrené údaje. Na začiatok šlo najmä o to, vedieť sa k tomuto zariadeniu pripojiť, správne s nim komunikovať, dokázať s neho získať informácie o spotrebe, vedieť ich správne vyhodnotiť a ďalej zúžitkovať. Po vyriešení týchto cieľov bolo potrebné ďalej zabezpečiť ukladanie údajov do cloudu, s čím je zároveň úzko spojené autentifikácia používateľov.

Vývoj aplikácie prebiehal predovšetkým pre systém Android, avšak framework Flutter poskytuje jednoduchú možnosť ako túto aplikáciu rozšíriť aj na iné systémy, ako napríklad iOS. Aplikácia je písaná tak, aby ju bolo možné spustiť na viacerých platformách, no tie sa líšia v tom, akým spôsob spracúvajú oprávnenia. Pre každú platformu je tak nutné definovať tieto povolenia zvlášť. Aplikácia aktuálne má nakonfigurované povolenia iba pre systém Android.

Prvým krokom vývoja aplikácie bolo zobrazovanie dostupných zariadení v okolí. Samozrejme sa v okolí zobrazilo hneď niekoľko rôznych Bluetooth zariadení. Zoznam Bluetooth zariadení, ktoré boli nájdené v okolí, sa obmedzil tak, aby obsahoval iba zariadenia, ktoré majú definovanú službu s konkrétnym Uuid, ktorú poskytuje vstavané zariadenie. Tým docielime to, že sa zobrazia len zariadenia, ktoré užívateľ skutočne očakáva a zároveň tým bránim užívateľovi, aby sa pripojil k nechcenému zariadeniu, či už vedome alebo nevedome.

Ďalším krokom implementácie bolo vytvorenie konfiguračnej obrazovky. Na tejto obrazovke môže užívateľ zmeniť názov zariadenia, ale čo je hlavné, musí nastaviť počet pulzov na jednotku spotrebovanej energie. Aplikácia zároveň v tomto momente odosiela do zariadenia časové razítko, ktoré ďalej slúži na priradenie dátumu a času k neskôr stiahnutým údajom.

Posledným krokom bolo vytvorenie hlavnej obrazovky, kde bolo možné synchronizovať údaje zo zariadenia, a zobrazovať ich v prehľade a grafoch. Takto mala aplikácia vytvorenú základnú funkcionalitu, ku bolo možné obohatiť o ďalšie vymoženosti.

6.1 Dátové štruktúry a poskytovatelia

Na implementáciu dátových štruktúr využívam dve triedy, `Device` a `Value`. Trieda `Device` uchováva informácie o zariadení ako ID, čo pre konkrétne zariadenie znamená jeho MAC adresu, názov zariadenia, označenie, či ide o aktívne zariadenie, teda to, s ktorým užívateľ aktuálne pracuje a počet pulzov, a jednotku, ktorá určuje, akej spotrebe odpovedá daný

počet pulzov. Na uchovávanie jednotky spotrebovanej elektrickej energie typ `enum` s názvom `Units`, ktorý k IDčku jednotky prihadzuje popis. Konštruktom triedy `Device` vyžaduje, aby boli vyplnené všetky hodnoty pre zariadenie. Na tvorbu objektov ešte ďalej využívam továreň, ktorej úlohou je premapovávať záznamy získané z realtime databázy Firebase, na instancie objektu `Device`.

Trieda `Value` reprezentuje dátové záznamy, ktoré prichádzajú zo vstavaného zariadenia. Záznamy obsahujú 2 hodnoty a to časové razítko a počet impulzov za danú dobu pre tento čas. Podobne ako pri triede `Device` aj tu sú vyžadované obe hodnoty a rovnako tak existuje továreň, ktorá spracováva hodnoty získane z databázy a prevádza ich na objekty.

Vytváranie a spracovávanie objektov týchto tried riadia dvaja poskytovatelia. Poskytovateľ `BleProvider`, ktorý je zodpovedný za Bluetooth komunikáciu so vstavaným zariadením a výmenu údajov a poskytovateľ `DataProvider`, vyžívaný na synchronizáciu údajov medzi aplikáciou a cloudovým úložiskom. Obaja poskytovatelia slúžia na komunikáciu s ďalšími službami, od ktorých získavajú a pripravujú dáta, ktoré následne sprostredkúvajú jednotlivým obrazovkám, ktoré ich zobrazujú. V istých prípadoch si títo poskytovatelia vymieňajú a využívajú funkcií toho druhého.

6.1.1 BleProvider

Poskytovateľ `BleProvider` rozširuje triedu `ChangeNotifier`, ktorá poskytuje API na oznamovanie poslucháčom, že došlo k zmene hodnoty niektorej zo sledovaných premenných. Poslucháči sú v tomto prípade pohľady, ktoré reagujú na zmeny hodnôt zmenou vykresľovaných widgetov. Keďže tento poskytovateľ komunikuje zo vstavaným zariadením a musí tak rozumieť dátam a ich forme, nachádza sa tu hneď niekoľko konštánt, ktoré s týmto súvisia. Ide o konštanty `recSize`, `valSize`, `valInRec` a `valsRange`. Tieto konštanty po poradí udávajú to, koľko bajtov má jeden záznam a koľko bajtov má jedna hodnota, ďalej celočíselný počet hodnôt v jednom zázname a nakoniec hodnota udávajúca časový úsek v milisekundách, ktorému zodpovedá jedna hodnota.

Trieda `BleProvider` obsahuje niekoľko privátnych premenných s ktorými pracuje. Prvá skupina premenných udržiava stream pre vyhľadávanie zariadení a stream informujúci o zmenách stavov počas pripájania. Ďalšia premenná udržiava zoznam zariadení, ktoré sa našli počas vyhľadávania. Nakoniec premenné, ktoré uchovávajú MAC adresu aktuálne pripojeného zariadenia, jeho názov a hodnoty synchronizované zo zariadenia. Ďalšia skupina premenných sú booleanovské premenné, ktoré udávajú aktuálny stav Bluetooth, na ktorý reagujú pohľady. Pohľady naslúchajú na zmeny týchto premenných pomocou ich `get` metód. Poskytovateľ `BleProvider` ešte obsahuje pár ďalších pomocných premenných.

Pri vytvorení objektu triedy `BleProvider` sa hneď v konštruktore spúšťa privátna metóda `_setStatusStream`. Táto metóda využíva objekt triedy `FlutterReactiveBle`, na spustenie streamu, ktorý načúva na zmeny v statuse Bluetooth. Statusy, na ktoré táto metóda reaguje sú `poweredOff` a `ready`, pri ktorých mení stavy premenných, ktoré sa ďalej propagujú na pohľady. Pri zmene premenných sa volá funkcia `notifyListeners`, ktorá ma za úlohu práve upozorniť načúvajúce pohľady na zmenu, ktorá nastala.

Metóda `scanForDevices`, ako už názov napovedá, slúži na vyhľadávanie BLE zariadení v okolí. Táto metóda nie je privátna a spúšťa sa tlačidlom z jedného z pohľadov. Ak ešte nebolo vyhľadávanie spustené, tak táto metóda spúšťa stream na vyhľadávanie zariadení v okolí. Tento stream obsahuje parametre, ktoré udávajú, že hľadáme iba také zariadenia, ktoré podporujú nami definované Uuid, pretože sa nechceme pripájať k zariadeniam, ktoré neslúžia na meranie spotreby elektrickej energie. Pri každom novo nájdenom zariadení táto

metóda pridá dané zariadenie do zoznamu nájdených zariadení a informuje poslucháčov. Ak je táto metóda spustená po tom, čo už bolo spustené vyhľadávanie, tak sa toto vyhľadávanie ukončí, zmenia sa stavové premenné a odošle sa informácia o tejto zmene.

Sada metód `connect`, `disconnect`, `_onDisconnect` slúžia na pripájanie a opájanie od vstavaného zariadenia. Pri pripájaní metóda `connect` najskôr overí, či už mobil nie je pripojený k požadovanému zariadeniu, poprípade či náhodou nie je mobil pripojený k inému zariadeniu. Ak by už existovalo nejaké spojenie, tak toto spojenie sa uzavrie, aby bolo možné pripojiť sa na zariadenie. V prípade, že sa pokúšame pripojiť po tom, čo sme zariadenie našli pomocou vyhľadávania okolitých zariadení, tak ešte ukončíme toto vyhľadávanie. Následne sa smartfón pokúsi o pripojenie k vstavanému zariadeniu v časovom limite 15 sekúnd. Pri pripájaní sa otvorí stream spojenia, na ktorom načúvame na zmeny stavu, Táto metóda sa vykonáva asynchrónne. Pri úspechu upravíme potrebné premenné a hneď požiadame pripojené zariadenie o zmenu veľkosti MTU metódou `_requestMTUchange`. Touto metódou sa pokúsime o nastavenie MTU na dĺžku 250 znakov aby sme v prípade ďalšieho prenosu dokázali preniesť čo najviac dát a čo najefektívnejšie, a šetrili tým spotrebu zariadenia. V prípade chyby alebo neúspechu prepošleme informáciu o neúspechu ďalej na poslucháčov.

V prípade odpájania je situácia jednoduchšia. Stačí uzavrieť stream spojenia, čím sa ukončí spojenie so zariadením. Táto metóda interne volá privátnu metódu `_onDisconnect`, ktorá upraví stavové premenné a informuje pohľady. Kód tejto metódy je oddelený od hlavnej metódy ktorá ju volá, pretože odpojenie môže nastať aj iným spôsobom a v takom prípade nie je potreba uzatvárať stream, ktorý sa uzavrel už sám.

Po prvom pripojení k vstavanému zariadeniu je potrebné nakonfigurovať čas pre modul RTC, aby sme vedeli k akému času sa viažu jednotlivé hodnoty. Na to slúži metóda `setupDevice`. Tá ďalej zavolá internú metódu `updateDeviceTime`, ktorá vytvorí odpoveď pre vstavané zariadenie a odošle aktualizovanú hodnotu časového razítka.

Metóda s názvom `_subscribeToRead` slúži na synchronizáciu údajov zo zariadenia. Zo zariadenia si najskôr prečítame časové razítko, od ktorého začínajú záznamy uložené v pamäti. Prečítame si aj ďalšiu charakteristiku, ktorá informuje o tom, či nedošlo k reštartu zariadenia a o prípadnom reštarte ďalej informujeme. Následne začneme odoberať notifikácie na požadovanej charakteristike. Po prihlásení na odoberanie začne vstavané zariadenie automaticky odosielať všetky údaje, ktoré má uložené vo svojej pamäti. Na konci tohto odosielania odošle paket s prázdnym telom, ktorý signalizuje, že už boli odoslané všetky údaje a nie je čo ďalej prijímať. V tomto momente sa mobil odhlasuje z odberu. Počas prijímania dát sa z údajov vytvárajú objekty triedy `Value` pomocou funkcie `_procesData` a vkladajú do zoznamu. Metóda `_processData` skladá prijaté bajty do hodnôt a každej hodnote priraďuje časové razítko. Prvá hodnota v poradí má časové razítko, ktoré sme získali zo zariadenia a každá ďalšia hodnota má posunuté časové razítko o hodnotu, ktorá je uložená v konštante `valsRange`. Táto hodnota musí pre správne fungovanie zostať rovnaká v stavanom zariadení aj mobilnej aplikácií. Udáva časový rozsah, počas ktorého sa sčítavajú pulzy. Tento zoznam nakoniec ešte predáme poskytovateľovi `DataProvider`, ktorý údaje nahrá do databázy. Táto metóda je volaná z vnútra metódy `readAllData`, ktorá ešte pred získavaním dát overuje a zabezpečuje, že máme zapnutý Bluetooth a že sme pripojený k zariadeniu.

6.1.2 DataProvider

Poskytovateľ `DataProvider` funguje v princípe rovnako ako jeho predchodca. Rozširuje triedu `ChangeNotifier` a slúži na synchronizáciu údajov medzi clodovým úložiskom a mo-

bilnou aplikáciou. Obsahuje niekoľko atribútov, ktoré uchovávajú napojenie na databázu, informácie o aktuálnom používateľovi, zoznam zariadení používateľa a ID aktuálne aktívneho zariadenia a aj dáta všetkých zariadení. Pomocou funkcie `get` môžu obrazovky získavať tieto údaje a zobrazovať ich ďalej užívateľovi.

V konštruktore triedy `DataProvider` nájdeme volanie metódy `setPersistenceEnabled` s atribútom `true`. Toto volanie zabezpečí, že všetky údaje sa budú odkladať do zariadenia aj lokálne a tým pádom nie je aplikácia závislá na neustálom pripojení na internet. V prípade, že zariadenie nie je pripojené k internetu, tak sa zobrazujú údaje, ktoré sú uložené priamo v zariadení. Ak dôjde k synchronizácií nameraných hodnôt zo vstavaného zariadenia, tak sa tieto hodnoty v režime offline uložia lokálne a po opätovnom pripojení k internetu dôjde k ich odoslaní do cloudu. Po tomto volaní ešte voláme metódu `_listenToDevices`. Tá sa hneď napojí na databázu a začne načúvam zmenám zariadení užívateľa, ktorý je aktuálne v aplikácií prihlásený. Pri zmene upraví zoznam zariadení a informuje ďalej všetkých, ktorý naslúchajú na zmeny.

Rovnako funguje metóda `_listenToData`. Tá reaguje na zmeny hodnôt v databáze. Odkazuje sa na cestu k dátam pomocou funkcie `_getPathToData`, ktorá do cesty k údajom ešte pridáva ID užívateľa. Prijaté dáta aktualizuje a zoraďuje, aby sa s nimi následne jednoduchšie pracovalo. Táto verejná metóda je zaobalená vo verejnej metóde `ListenToDataStream`, ktorú volajú obrazovky. Na ukončenie načúvania slúži metóda `cancelDataStream`.

Metódu `uploadData` volá poskytovateľ `BleProvider` po tom, čo získa nové namerané hodnoty zo zariadenia. V tele metódy je kód, ktorý má za úlohu transformovať dáta do podoby JSON štruktúry, ktorá sa odosiela na cloud na aktualizáciu údajov.

Nové zariadenie nastavíme pomocou metódy `setNewDevice`. Vo vnútri metódy uložíme do JSON objektu informácie o zariadení a tie odošleme na cloud. Následne ešte nastavíme toto zariadenie ako aktuálne aktívne a ostatné zariadenia na neaktívne a to pomocou metódy `setActiveDevice`.

Pri odstraňovaní zariadenia využijeme metódu `removeDevice`. Tá vyhľadá zariadenie, odstráni ho, odstráni jeho namerané údaje a ako aktívne zariadenie nastaví prvé v zozname ostatných zariadení.

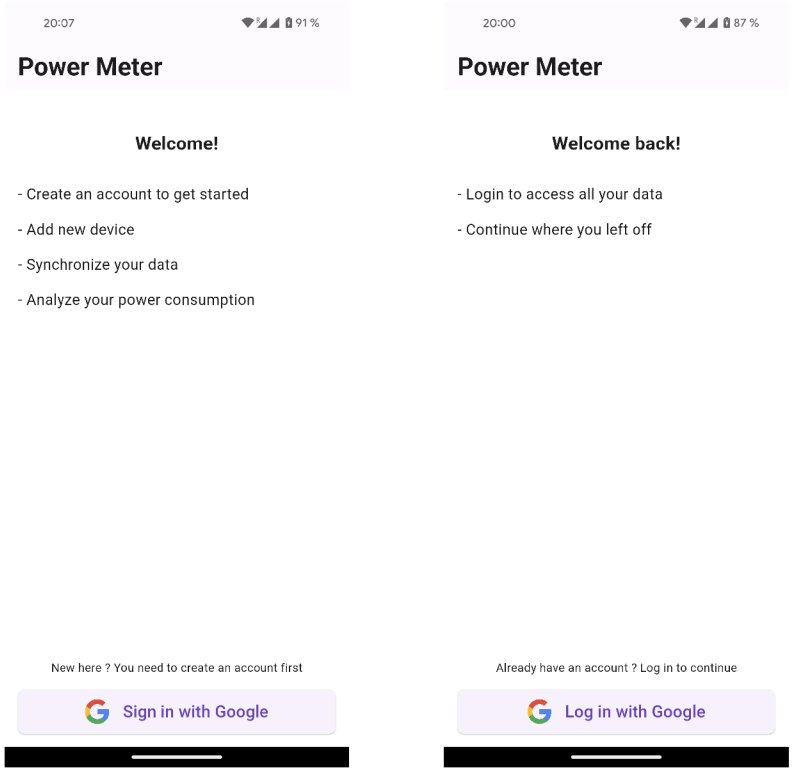
6.2 Obrazovky

Pre každú obrazovku, tak ako som si ich rozdelil v návrhu som vytvoril samostatný súbor. Keďže mali tieto obrazovky niektoré komponenty spoločné, vytvoril som dva widgety, ktoré som následne používal naprieč celou aplikáciou. Umiestnil som ich do samostatnej zložky a nazval som ich `base_scaffold` a `bottom_button`. `base_scaffold` má atribúty `body` a `bodyTitle`. Dátový typ widgetu `bodyTitle` je string, ktorý udáva názov obrazovky, ktorý sa zobrazí vo widgete `AppBar`. Atribút `body` je zoznamom widgetov. Tieto widgety sa pridajú do tela widgetu `base_scaffold` a vykreslia sa v ňom. `base_scaffold` tvorí základnú štruktúru, ktorá sa používa naprieč všetkými obrazovkami a je ju možné zmeniť na jednom mieste. Podobne tlačidlo s názvom `bottom_button` sa využíva na viacerých obrazovkách a v jeho triede si uchováva štýl ako zaoblenie hrán a rozmery, ktoré sa vrámci obrazoviek nemenia a vychádzajú z tohto miesta. Čo sa mení je text tlačidla a jeho chovanie, čo sa určuje pomocou atribútov `text` a funkcie `onPressed`, ktorá sa zavolá po stlačení tlačidla.

Program `main` určuje, ktorá obrazovka sa zobrazí ako prvá po zapnutí aplikácie. Po otvorení aplikácie sa najskôr inicializuje Firebase aplikácia a hneď na to sa spúšťa aplikácia s využitím poskytovateľov `BleProvider` a `DataProvider`, ku ktorým sa pristupuje naprieč celou aplikáciou. Program `main` sa ďalej odkazuje na triedu `AuthSevice`, ktorá v metóde

`handleAuthState` overuje, či je užívateľ prihlásený. Ak je užívateľ prihlásený, tak je ďalej presmerovaný na obrazovku oprávnení, v opačnom prípade putuje na obrazovku s prihlásením.

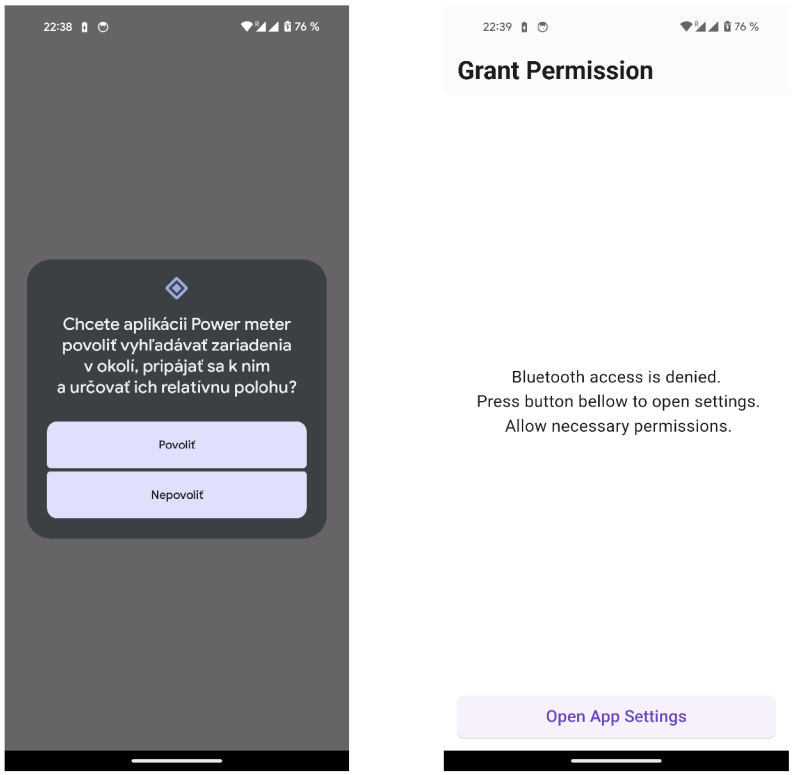
6.2.1 Obrazovka prihlásenia



Obr. 6.1: V ľavo obrazovka, ktorá sa užívateľovi zobrazí pri prvom prihlasovaní a v pravo obrazovka zobrazujúca sa užívateľovi, ktorý sa opätovne prihlasuje do aplikácie.

V inicializácií stavu obrazovky sa najskôr pozeráme na to, či sa už užívateľ niekedy prihlásil do aplikácie. V závislosti na výsledku sa užívateľovi môžu zobraziť dve odlišné obrazovky. Ak užívateľ vidí túto obrazovku po prvý krát, zobrazí sa mu krátky prehľad na čo aplikácia slúži a ako má užívateľ ďalej pokračovať. V prípade, že užívateľ už bol niekedy v minulosti prihlásený, zobrazí sa mu privítanie. V oboch prípadoch sa na spodnej časti obrazovky zobrazí tlačidlo `bottom_buttom` s výzvou na prihlásenie. Rozdiel medzi obrazovkami je možné vidieť na obrázku 6.1.

Po kliknutí na tlačidlo sa vykoná kód metódy `signInWithGoogle` triedy `AuthService`. V tejto metóde sa pokúsime o prihlásenie požívateľa pomocou Google emailu a očakávame, kým sa autentifikuje. V prípade, že sa prihlásil po prvý krát, tak si tom uložíme informáciu pomocou triedy `SharedPreferences` ako `bool` premennú s názvom `firstLogin` a hodnotou `false`. Ak prihlásenie prebehlo úspešne, tak sa metóda `handleAuthState` postará o to, že užívateľa presmeruje na obrazovku s oprávneniami.



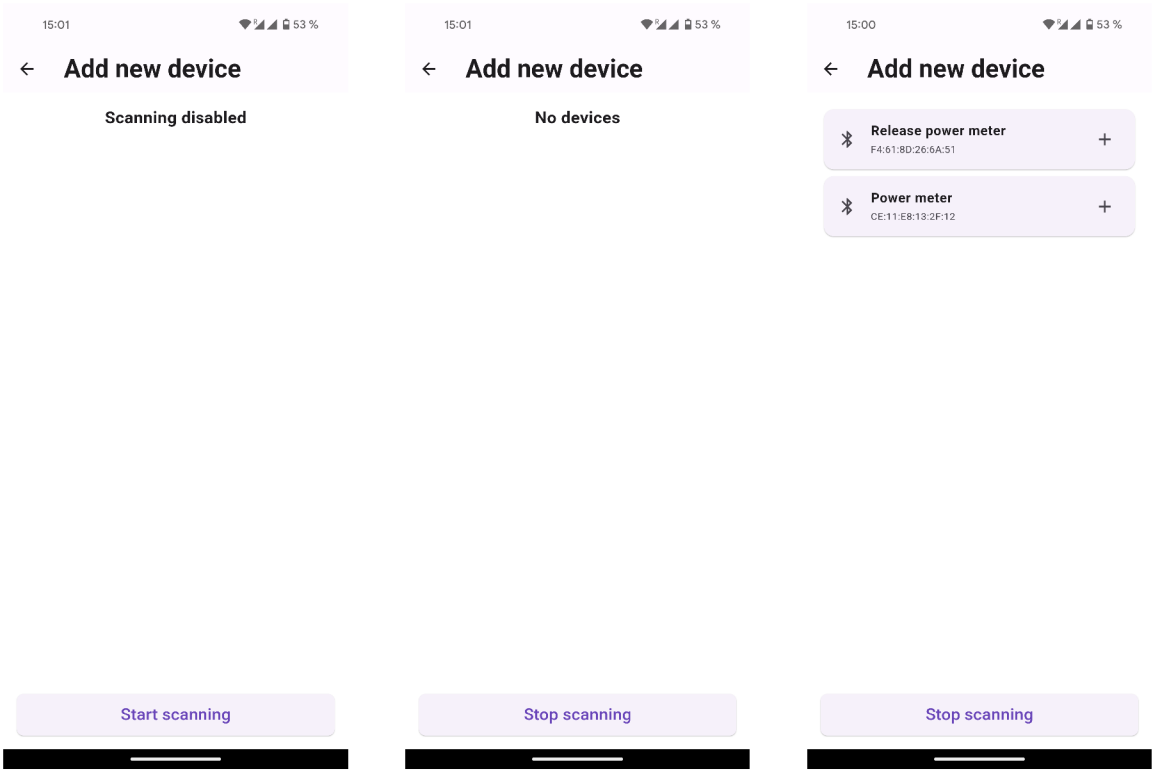
Obr. 6.2: V ľavo obrazovka, ktorá vyzýva užívateľa na udelenie oprávnení a v pravo obrazovka po tom, čo užívateľ nepovolil tieto oprávnenia.

6.2.2 Obrazovka s oprávneniami

`PermissionPage` rieši to, či užívateľ udelil nevyhnutné oprávnenia na využívanie Bluetooth. Táto trieda má spočiatku nastavený atribút `_waitingForPermission` na `true`, na ktorom detekuje, či sa aktuálne čaká na vyhodnotenie žiadosti o oprávnenia. Žiadosť o oprávnenia sa volá hneď pri inicializácií obrazovky pomocou metódy `permissionRequest`. Táto metóda žiada o oprávnenia na skenovanie a pripájanie sa k Bluetooth zariadeniam. V prípade, že sú oprávnenia udelené, tak táto funkcia presmerováva užívateľa na hlavnú obrazovku, ak tomu tak nie je, mení sa hodnota premennej `_watingForPermission`. Táto premenná je zodpovedná za to, čo sa užívateľovi vykresľuje na obrazovke. Ak aplikácia stále čaká na udelenie oprávnení, tak sa zobrazí jednoduchý `CircularProgressIndicator` widget, ktorý symbolizuje, že sa na niečo čaká. V prípade, že už čakanie skončilo a oprávnenia neboli udelené, tak sa namiesto tohto widgetu zobrazí obrazovka, ktorá informuje užívateľa o nutnosti udelenia oprávnení pre správne fungovanie aplikácie a ponúka mu v spodnej časti obrazovky tlačidlo na zmenu oprávnení v nastaveniach. Zároveň táto obrazovka pri inicializácií registruje observer na to, aby dokázala reagovať na zmeny stavu aplikácie. V prípade, že užívateľ klikne na tlačidlo je presmerovaný do nastavení, kde môže zmeniť svoje rozhodnutie a udeliť potrebné povolenia. Po návrate do aplikácie aplikácia zistí, že sa užívateľ vrátil a automaticky opäť požiada oprávnenia, aby si overila, či užívateľ tieto oprávnenia v nastaveniach udelil. V prípade, že k tomu došlo, tak ho presmeruje na hlavnú obrazovku. Obrazovku a jej varianty je možné vidieť na obrázku 6.2.

6.2.3 Skenovacia obrazovka

Na tejto obrazovke sa užívateľ ocitne, ak sa pokúša o pripojenie prvého zariadenia, alebo následne aj každého ďalšieho. Obrazovka sa skladá z jedného tlačidla, ktorým sa zapína a vypína Bluetooth skenovanie. Stlačením tlačidla sa zavolá metóda poskytovateľa `BleProvider` s názvom `scanForDevices`. Táto obrazovka načúva na zmeny premenných, ktoré poskytuje `BleProvider`. Obrazovka mení svoj dizajn na základne toho, či je skenovanie zapnuté alebo nie a či existujú nejaké nájdené zariadenia. Rozdiel je možno vidieť na obrázku 6.3.

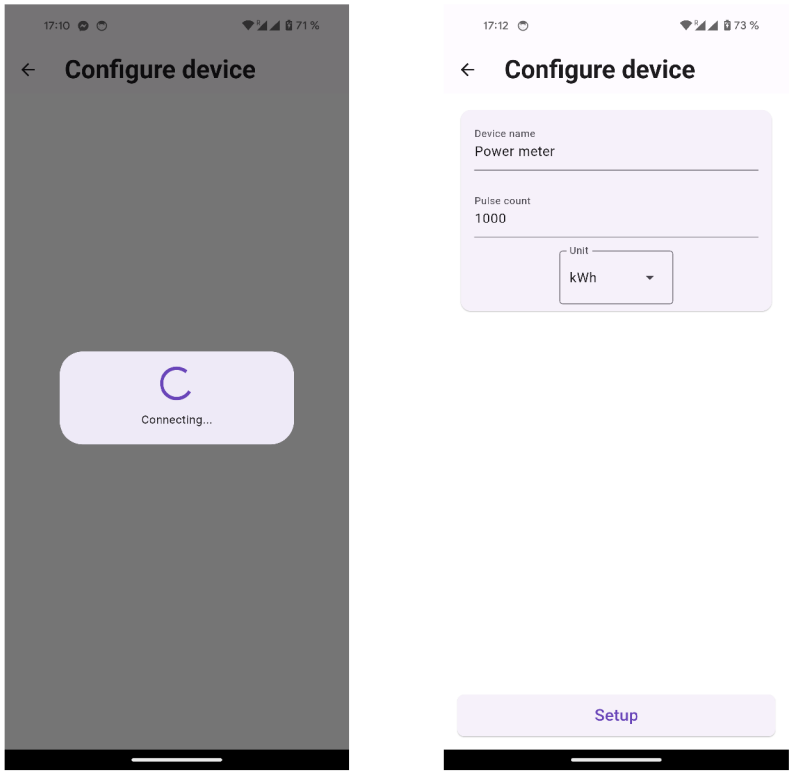


Obr. 6.3: Vľavo skenovacia obrazovka, ak nie je zapnuté skenovanie, v strede ak sa zatiaľ nenašlo žiadne zariadenie a vpravo sú zobrazené nájdené zariadenia.

V prípade, že skenovanie nie je zapnuté, tak text tlačidla vyzýva na začiatok skenovania a na obrazovke sa zobrazuje jednoduchý text s oznamom, že skenovanie je vypnuté. Po zapnutí skenovania tlačidlom sa najskôr zobrazí výzva na zapnutie Bluetooth, ak ešte nie je zapnutý a následne sa postupne začnú zobrazovať jednotlivé zariadenia, o ktorých nájdení informuje `BleProvider`. Počas skenovania sa mení text tlačidla, aby tak vyjadroval, že po opätovnom stlačení dôjde k prerušeniu skenovania. V prípade, že sa zatiaľ nenašlo žiadne zariadenie, tak o tom informuje jednoduchý text. V opačnom prípade sa vyžíva metóda `_devices`. Tá ma za úlohu vykresliť zoznam jednotlivých zariadení. Ja vykreslenie sa využíva widget `ListView`, ktorý má metódu `builder`. Táto metóda má atribúty `itemCount`, aký počet položiek sa má vykresliť a `itemBuilder`, ktorý udáva, ako sa má vykresliť konkrétny záznam. Počet položiek udáva dĺžka zoznamu zariadení. Jednotlivé položky sa vykresľujú ako widgety `Card`, vo vnútri ktorých sa nachádza ďalši widget `ListTile`. V tomto widgete nastavujeme počiatočnú ikonku, ktorá zobrazuje Bluetooth logo, ďalej titulok, ktorý zobrazuje meno zariadenia a podtitulok, v ktorom je jeho MAC adresa a nakoniec sa zobrazuje ikona plus, ktorá simbolizuje pridanie zariadenia. Celá táto položka reaguje a kliknutie, po ktorom sa

volá metóda `connect` s atribútom `id`. Počas pripájania sa zobrazuje animácia načítania, ktorú po úspešnom pripojení strieda ďalšia obrazovka, ktorá nastavuje vybrané zariadenie.

6.2.4 Obrazovka s nastavením zariadenia



Obr. 6.4: Vľavo obrazovka, ktorá sa zobrazuje kým prebieha pripájanie k zariadeniu, vpravo obrazovka na ktorej sa nastavuje konfigurácia zariadenia.

Pred používaním zariadenia je ešte nutné ho najprv nakonfigurovať. To sa odohráva na konfiguračnej obrazovke, ktorú je možné vidieť na obrázku 6.4. Na tejto obrazovke sa vo widgete `Card` nachádzajú v stĺpci dve wdigety `TextFormField` na zadávanie textu. Prvý slúži na zadanie názvu zariadenia. Jeho hodnota sa predvyplní menom, pod ktorým sa zariadenie oznamuje na sieti. Druhé textové pole slúži na zadanie počtu pulzov. Do tohto poľa je možné zadať iba číselný údaj. Ako predvyplnená hodnota je tu číslo 1000. V stĺpci sa ešte ďalej nachádza widget `DropDownMenu` v ktorom je možné vybrať jednotku spotrebovanej energie. Záznamy sa do neho získavajú z dátového typu `enum` s názvom `Units`. V spodnej časti obrazovky sa ešte nachádza tlačidlo na potvrdenie hodnôt a nastavenie zariadenia. Toto tlačidlo najskôr volá metódu `setupDevice` poskytovateľa `BleProvider` na pripojenie a nastavenie zariadenia a následne odosiela údaje o konfigurácií zariadenia do cloudu pomocou poskytovateľa `DataProvider` a jeho metódy `setNewDevice`. Po tomto nastavení je užívateľ Ďalej presmerovaný na hlavnú obrazovku.

6.2.5 Hlavná obrazovka

Hlavná obrazovka na začiatok očakáva odpoveď od poskytovateľa `DataProvider` o tom, či sa podarilo načítať zariadenia prihláseného užívateľa. Počas čakania na krátky moment

zobrazuje užívateľovi načítaniu animáciu. V prípade, že užívateľ aktuálne nemá žiadne aktívne zariadenie, tak je presmerovaný na skenovaciu obrazovku, kde si môže pridať nové zariadenie.

Po synchronizácií zariadení sa vyberie aktívne zariadene, ktorého údaje sa začnú zobrazovať na obrazovke. V titulku obrazovky sa zobrazí názov zariadenia a v tele aplikácie sa následne zobrazujú textové údaje a graf. Celé telo aplikácie je zaobalené vo widgete `RefreshIndicator`. Tento widget poskytuje funkcionalitu, pomocou ktorej je možné spustiť akciu po dostatočnom vysunutí indikátoru. Tento indikátor následne ostáva zobrazený do momentu, dokým nedôjde k dokončeniu akcie. Tento widget využívam na aktualizáciu údajov zo zariadenia. Akcia, ktorá sa spustí je metóda `ReadAllData` triedy `BleProvider`. V prípade, že sa nepodarí pripojiť na zariadenie a načítať údaje, tak sa vráti error, na čo obrazovka reaguje zobrazením widgetu `Fluttertoast` s textom chyby. V opačnom prípade sa zobrazí hláška o úspešnej synchronizácií údajov. Widget `RefreshIndicator` má ďalej potomka `ListView`. V ňom sa nachádzajú ďalšie časti obrazovky, ktoré sú postupne vykresľované. Každá časť sa nachádza v samostatnej karte a vykresľuje sa samostatnou metódou.

Prvá metóda je `restartWaring`. Tá na karte vykresľuje informáciu o tom, či sa počas aktualizácie údajov nezistil reštart zariadenia. Ak áno, na obrazovke sa zobrazí varovanie v podobe oranžového textu a krížiku, pomocou ktorého je možné oznámenie odstrániť. Toto oznámenie sa zobrazuje len v prípade, že došlo k reštartu.

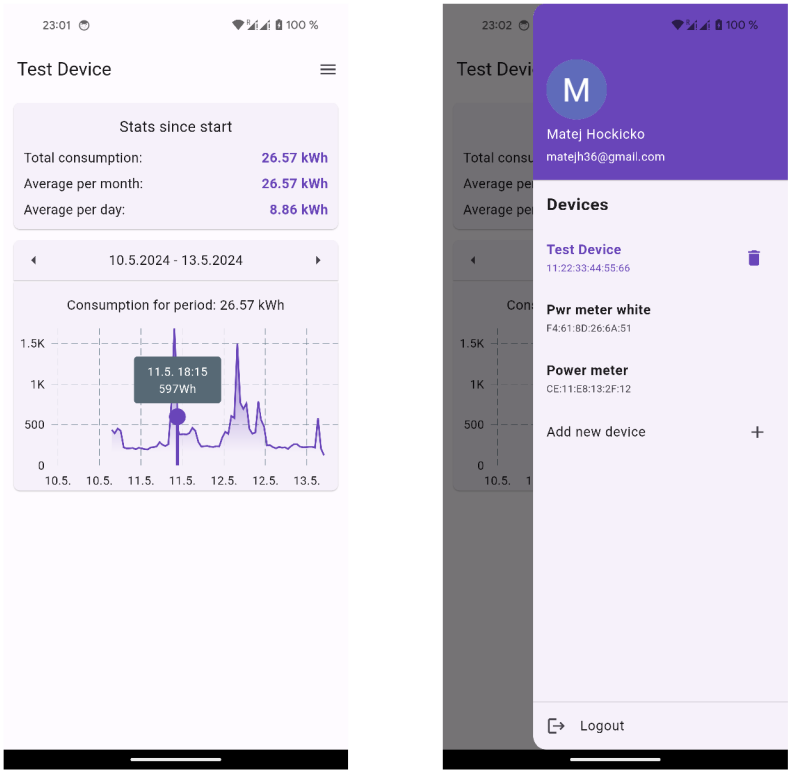
Za týmto oznámením nasleduje metóda s názvom `infoCard`. Ide o kartu, na ktorej sa zobrazujú údaje ohľadom spotreby. Každý údaj o spotrebe sa nachádza v jednom riadku a všetky tieto riadku sú vložené do stĺpca. Údaje sa získavajú pomocou metód `totalUsage`, `averagePetMonth` a `averagePerDay`. V týchto metódach sa spočíta celkový počet pulzov a prípadne sa ešte vydelí počtom období. Všetky metódy nakoniec volajú ďalšiu metódu `formatValue`, ktorá ešte výsledné číslo upraví tak, aby nebolo príliš malé alebo veľké a to tak, že posunie jednotku jedným alebo druhým smerom, aby výsledné číslo, ktoré sa bude zobrazovať ležalo v rozsahu 1 až 1000.

Ďalšia karta v zozname je vykresľovaná pomocou funkcie `dateRangePickerButton`. Ide o riadok, v ktorom sa po stranách nachádzajú šípky na zmenu dátumu a v strede sa zobrazuje text s aktuálne zvoleným dátumovým rozsahom. Na zvolenie dátumu sa využíva metóda `_showDateRangePicker`. Tá volá rovnako nazývanú funkciu ktorá vracia premennú typu `DateTimeRange`. Táto funkcia zobrazí užívateľovi kalendár, pomocou ktorého si môže zvoliť dátumový rozsah, pre ktorý chce zobraziť namerané hodnoty. Prednastavená hodnota zobrazuje aktuálny deň. Ak z funkcie dostaneme výsledok, tak ten upravíme tak, aby koncový dátum končil v čase 23:59:59, aby sa nám tak zobrazili hodnoty do konca daného dňa. Pomocou funkcie `setState` ešte oznámime prípadnú zmenu a vynútime tak prekreslenie obrazovky s novými údajmi. V prípade kliknutia niektorého z tlačidiel po stranách sa volá metóda `shiftDateRange` a atribútom `forward`, ktorý udáva, či sa má zvolený rozsah posúvať dopredu alebo dozadu. Dátumy sa posunú o rozdiel medzi počiatočným a koncovým dátumom a opäť sa zavolá funkcia `setState`.

Posledná karta zobrazuje graf s nameranými hodnotami pre zvolené časové obdobie pomocou metódy `chartCard`. Táto metóda do karty vykreslí textové pole, s hodnotou celkovej spotreby za zvolené obdobie. Pod týmto textom sa ďalej zobrazuje widget `AspectRatio`, ktorý zabezpečí, že pomer strán bude 2:1. Následne je volaná metóda `chart`, ktorá vykreslí samotný graf. Pred vykresľovaním sa najskôr zavolá metóda `calculateFilteredValues`. Tá najskôr vyfiltruje tie hodnoty, ktoré sa nachádzajú vo zvolenom období a následne ešte volá metódu `aggregateValues`. V nej sa ešte údaje upravia tak, aby v grafe nebolo príliš veľa

hodnôt, ale aby sa zobrazovalo zhruba 100 agregovaných hodnôt. To sa docieli tak, že sa najskôr zráta počet sekúnd v požadovanom intervale. Pomocou metódy `findNiceRoundNumber` sa ďalej hľadá najbližšie pekné číslo zo zoznamu, na ktoré sa budú hodnoty agregovať. Vytvorí sa nový zoznam, ktorý obsahuje tieto agregované hodnoty, s ktorým ďalej pracuje graf.

Na vykreslenie grafu sa využíva widget `LineChart`. Prvý atribút tohto widgetu s názvom `data` popisuje pomocou widgetu `LineChartData` ako vyzerajú údaje. Udáva sa tu minimum a maximum na oboch osách, a parametre toho, čo a ako sa má vykresľovať po osách grafu. Na ose X sa zobrazujú popisky, ktoré sa prevádzajú z kľúča záznamu, ktorý obsahujú časové razítko, na ľudsky čitateľný text obsahujúci čas alebo dátum zaznamenanej hodnoty podľa šírky zvoleného časového rozpätia. V atribúte `lineBarsData` sa popisuje spôsob akým sa budú zobrazovať údaje. Dôležité je najmä nastavenie atribútu `spots`, do ktorého prídu jednotlivé body, z ktorých sa ďalej vytvára krivka na grafe. Do týchto bodov sa mapujú hodnoty zo zoznamu agregovaných údajov typu `Value`. Tie sa mapujú do triedy `FlSport` tak, že hodnota X udáva časové razítko záznamu a hodnota Y udáva agregovaný údaj.



Obr. 6.5: Vľavo hlavná obrazovka, na ktorej je možné vidieť namerané údaje v grafe, vpravo bočné menu halvnej obrazovky.

Na tejto obrazovke sa ešte zároveň nachádza aj bočné menu. Na jeho vykreslenie sa využíva metóda `_drawer`. Bočné menu slúži na prepínanie medzi viacerými zariadeniami, prípadne na pridanie nového zariadenia. Toto menu tvorí widget `Column`, na vrchu ktorého sa nachádza funkcia `UserAcountDrawerHeader`. Tá vykresľuje informácie o aktuálne prihlásenom účte užívateľa ako jeho meno, emailovú adresu a profilovú fotku. Ďalej sa v zozname nachádza položka `ListTile`, ktorá slúži ako nadpis sekcie a udáva, že sa pod ňou budú zobrazovať jednotlivé zariadenia. Následne sa do widgetu `Expanded` vykresľuje

zoznam zariadení, aby tak zabral zvyšnú časť obrazovky. Pomocou metódy `builder` widgetu `ListView` sa postupne pridáva jednotlivé zariadenia. Zariadenia v zozname obsahujú názov a MAC adresu. Ak je zariadenie aktívne, zobrazuje sa odlišnou farbou od ostatných. Zároveň má aktívne zariadenie naviac na konci widget `IconButon`, ktorý po stlačení volá metódu `_deleteActiveDevice`. Táto funkcia najskôr pomocou dialógu overí, či užívateľ chce skutočne odstrániť zariadenie, a následne volá metódu `removeDevice` poskytovateľa `DataProvider`, ktorá sa postará o zmazanie zariadenia a jeho údajov. Do atribútu `itemCount` metódy `builder`, ktorá vykresľuje zariadenia, je nastavená hodnota o 1 väčšia, než je počet zariadení. To sa následne overuje vo funkcii v atribúte `itemBUilder`, kde ak sme narazili na koniec zoznamu, tak ešte vykreslíme jednu položku. Táto položka slúži na pridanie nového zariadenia. Po kliknutí volá metódu `_addNewDevice`, ktorá presmeruje užívateľa na obrazovku `ScanPage`, kde si môže pridať nové zariadenie. Na záver stĺpca sa ešte nachádza oddeľovač `Divider`, po ktorom nasleduje widget `ListTile`. Jeho úlohou je odhlásiť užívateľa po kliknutí. Po odhlásení je užívateľ vrátený na prihlasovaciu obrazovku, kde sa môže prihlásiť akýmkoľvek ďalším účtom. Výslednú obrazovku je možné vidieť na obrázku 6.5.

6.3 Integrácia a využitie cloudových služieb Firebase

Jedným z dôležitých krokov vývoja aplikácie bolo prepojenie so službami Firebase. Firebase sa využíva na autentifikáciu užívateľov a ukladanie údajov do cloudovej databázy.

Vo Firebase konzole je najskôr potrebné vytvoriť si projekt. Projekt nesie názov Power Meter. V rámci tohto projektu sa využívajú dve služby a to autentifikáciu a Realtime databáza.

Autentifikácia funguje skrz prihlasovanie cez google účet. Táto možnosť mi prišla ako najviac prívetivá pre používateľov, ktorý už nevyhnutne používajú tento google účet vo svojich zariadeniach, aby tak vôbec mali prístup k aplikáciam v obchode google play. Zároveň je to jednoduché riešenie, keďže služba Firebase aj framework Flutter sú priamo od Googlu a ich prepojenie je tak jednoduché.

Pre správne fungovanie služieb Firebase je nevyhnutné v nastavení projektu pridať kľúč, ktorým sa podpisuje mobilná aplikácia. Tak je zaistené to, že sa k databázy pripojí jedine táto aplikácia, a žiadna iná.

Pre databázu sa ešte nastavili oprávnenia tak, že každý užívateľ môže čítať a zapisovať len tam, kde sa v ceste dotazu zhoduje jeho ID užívateľa s ID záznamu.

Kapitola 7

Testovanie a vyhodnotenie systému

Finálne testovanie prebiehalo priamo na elektromere. Zhotovené zariadenie spolu s vytlačenou 3D krabičkou bolo upevnené na jeho povrchu. Najskôr sa pomocou obojstrannej lepiacej pásky osadila spodná časť krabičky tak, aby bol otvor v krabičke zarovnaný na LED dióde. Následne sa do krabičky vložilo vstavané zariadenie, do ktorého sa zasunula batéria. Nakoniec sa krabička uzatvorila a vstavané zariadenie sa nakonfigurovalo pomocou mobilnej aplikácie. V tomto momente bola poznačená aj hodnota spotreby zobrazená na elektromere. Údaje zo zariadenia sa následne synchronizovali 1 až 2krát do dňa a vždy sa porovnávala nameraná spotreba zobrazená v mobilnej aplikácií so spotrebou na elektromere. Takto sa zariadenie testovalo v reálnej prevádzke niekoľko dní. Nameraná spotreba sa zhodovala so skutočnou spotrebou. Fotky z osadenia vstavaného zariadenia je možné nájsť v prílohe A.

7.1 Časová agregácia pulzov a zaplnenie pamäte

Na zaznamenávanie počtu pulzov som zvolil časový interval 5 minút. Každých 5 minút sa do zariadenia ukladajú údaje o veľkosti 2B, pričom pre potreby ukladania dát je vyhradená pamäť o veľkosti 307200B. Z týchto čísel vyplýva, že zariadenie môže bežať nepretržite viac ako 533 dní pred tým, než sa jeho pamäť zaplní. Táto vyhradená pamäť však počíta s vývojovou verziou kódu, ktorá zaberá približne 188kB pamäte. Verzia kódu určená na distribúciu je však zhruba o 50kB menšia a je tak možno vyhradiť viac pamäte na zápis hodnôt. Hodnotu intervalu je tak možné znížiť napríklad na 1 minútu a dostať tak jemnejšiu granularitu nameraných údajov. Pri zachovaní rovnakého časového intervalu je možné využiť o radu nižší mikrokontrolér nrf52820, ktorý síce disponuje iba pamäťou o veľkosti 256kB, no v porovnaní s nrf52832 dosahuje nižšiu spotrebu. Pri využití tejto možnosti sa optimalizáciou oblastí pamäte na distribučnú verziu firmwaru by bolo možné využiť na ukladanie údajov pamäť o veľkosti zhruba 120kB, čo by vystačilo na 208 dní ukladania záznamov. V prípade zaplnenia pamäte zariadenie ďalej zaznamenáva spotrebu, pričom využíva pamäť ako kruhový buffer. Znamená to, že najnovší záznam vždy prepíše ten najstarší. V takomto prípade sú na zariadení uložené vždy merania za posledných 533 dní.

7.2 Odhadovaná výdrž zariadenia

Na odhad spotreby som využil nástroj Power Profiler[1]. Do tohto nástroja som nastavil parametre BLE komunikácie podľa implementácie vstavaného zariadenia. Zapol som využitie

[1] https://devzone.nordicsemi.com/power/w/opp/2/online-power-profiler-for-bluetooth-le

DCDC regulátora, ktorý sa nachádza v module BT832 a je aj povolený v konfigurácií. Ako zdroj hodín som vybral externý kryštál a výkon antény som nechal na 0dBm, keďže ten som v implementácií neupravoval.

Rolu Bluetooth som najskôr nastavil na Advertising (RX/TX), pretože zariadenie bude väčšinu času oznamovať prítomnosť na sieti a zároveň sa k nemu bude možné pripojiť. Interval vysielania je nastavený na dve sekundy a veľkosť odosielaných údajov na 11 bajtov, čo je dĺžka názvu zariadenia. Priemerná spotreba zariadenia v takomto prípade dosahuje odhadovanú hodnotu $7,4\mu A$.

Za predpokladu, že sa mobilná aplikácia bude k vstavanému zariadeniu pripájať raz denne, je možné efekt tohto spojenia a prenosu údajov na celkovú spotrebu zanedbať, keďže nebude dostatočne veľký.

Medzi Bluetooth udalosťami bude zariadenie väčšinu neaktívne v režime System ON. Zariadenie sa prebúdza len na prerušenia od GPIO, kedy iba zvýši hodnotu počítadla a je ďalej neaktívne, alebo na prerušenie od časovača, počas ktorého dochádza k uloženiu hodnoty do pamäte zariadenia. Ide tak o veľmi krátky moment, pri ktorom je spotreba prebudeného procesoru zanedbateľná. Pomocou konfigurácie zariadenia bolo možné nastaviť a sledovať vyťaženie procesoru každé dve sekundy, no vyťaženie procesoru mimo komunikácie Bluetooth je tak malé, že ho ani nebolo možné zaznamenať. Spotreba by sa tak mala pohybovať na úrovni $1,9\mu A$, čo je hodnota z dokumentácie pre nečinný mikrokontrolér v stave System ON. S touto spotrebou počíta aj nástroj na odhad spotreby, ktorý udáva spotrebu v nečinnosti zariadenia ako $2\mu A$.

Do celkovej spotreby ešte prispieva hardwerový obvod na detekciu optického impulzu. Na zistenie jeho spotreby je možné využiť simulačný program Tina TI. Do obvodu som zapojil ampérmeter a generoval som pulzy, medzi ktorými som nastavil medzeru 20ms. Nad týmto obvodom som následne spravil tranzientnú analýzu, z ktorej som získal priemernú hodnotu prúdu. Výsledok je približne 200nA.

Ak odhadovanej spotrebne zariadenia pripočítame spotrebu hardvérového obvodu na snímanie optického výstupu elektromera a pridáme rezervu, tak sa dostaneme na hodnotu $8\mu A$. Ak rátame s kapacitou batérie 200mAh, tak potom pri zanedbaní samovoľného vybíjania batérie vydrží zariadenie 25 000 hodín, čo je viac ako 2 a pol roku. Z uvedeného vyplýva, že ak by sme chceli skrátiť interval oznamovania, aby sme uľahčí vyhľadávanie a pripájanie k zariadeniu, zvýšili by sme tým spotreba, no zariadenie bude stále schopné vydržať z batérie aj niekoľko mesiacov. Pri aktuálnej konfigurácií je možné synchronizovať údaje zo zariadenia v priebehu 4 až 8 sekúnd.

7.3 Výsledná cena vstavaného zariadenia

V tabuľke 7.1 je možné počet kusov a celkovú cenu komponentov, ktoré sú potrebné na zhotovenie jedného vstavaného zariadenia. komponenty sú zoradené od najvyššej ceny.

Celková cena súčiastok na zhotovenie jedného zariadenia je 344,70 Kč. Z tabuľky vyplýva, že najväčším podielom na celkovej cene sa podieľa práve modul BT832. Ak by šlo o zníženie celkovej ceny, tak by bolo možné tento modul nahradiť priamo za mikrokontrolér nrf5283, ktorý je možné zaobstarať za 80 Kč. K tejto cene je však ale ďalej nutné pripočítať náklady na komponenty, ktoré by bolo potrebné osadiť k mikrokontroleru pre jeho správne fungovanie, a ktoré už sú priamo zahrnuté v module BT832. Ide o komponenty ako 32MHz kryštál a k nemu potrebné komponenty, cievky potrebné pre DCDC regulátor a ďalšie kondenzátory a komponenty potrebné pre chod zariadenia a vyladenie obvodu pre fungovanie antény. Zostavenie vstavaného zariadenia s využitím mikrokontroléru namiesto

Komponent	Počet kusov	Celková cena v Kč
modul BT832	1	221,03
komparátor TLV3691	1	28,82
fotodióda	1	17,49
32kHz kryštál	1	15,61
kolíková lišta	1	12,35
držiak na batériu	1	10,25
LED	1	8,85
tlačidlo	1	6,52
kondenzátor 10pF	2	4,66
kondenzátor 0,1µF	2	4,66
odpor 470kΩ	2	4,66
kondenzátor 1µF	1	2,81
kondenzátor 10nF	1	2,33
odpor 1MΩ	1	2,33
odpor 220Ω	1	2,33

Tabuľka 7.1: Cena jednotlivých súčiastok zoradená od najvyššej ceny

modulu BT832 by vyšlo lacnejšie, a to aj po započítaní ďalších potrebných súčiastok, ktoré sa však pohybujú v ráde jednotiek korún. Nevýhodou by však bola nutnosť návrhu Bluetooth antény a jej vyladenie.

Kapitola 8

Záver

Cieľom tejto bolo navrhnúť a implementovať systém na registráciu spotrebovanej elektrickej energie. Tento systém sa má skladať zo vstavaného zariadenia, ktoré má snímať optický výstup elektromera a namerané hodnoty ďalej predávať cez Bluetooth Low Energy do mobilného zariadenia. Úlohou mobilného zariadenia je tieto údaje získavať, konfigurovať vstavané zariadenie a umožňovať zobrazovanie nameraných hodnôt.

Výstupom tejto práce je zariadenie využívajúce mikrokontrolér nrf52832 od firmy Nordic, ktoré je možné upevniť na elektromer a toto zariadenie bude ďalej zaznamenávať spotrebu elektrickej energie. Zariadenie bolo navrhnuté tak, aby bolo napájané z gombíkovej batérie CR2032 a to tak, aby pri jeho činnosti bola optimalizovaná spotreba elektrickej energie. Zariadenie by podľa teoretických výpočtov malo byť schopné vydržať napájané z jednej batérie viac ako 2 roky. Toto zariadenie je ďalej schopné komunikácie s mobilnou aplikáciou pomocou technológie Bluetooth Low Energy.

Ďalším výsledkom tejto práce je mobilná aplikácia pre systém Android naprogramovaná vo frameworku Flutter. Aplikácia slúži na nastavenie a správu vstavaného zariadenia a na získavanie a zobrazovanie nameraných údajov. Aplikácia úzko spolupracuje a s realtime databázou Firebase, vďaka ktorej je možné údaje prehliadať na viacero zariadeniach súčasne a namerané údaje mať uložené na jednom mieste v Cloude. Samozrejmosťou je aj využitie lokálneho úložiska, vďaka čomu je aplikácia rovnako využiteľná aj v režime offline, pričom údaje sa synchronizujú hneď po tom, čo má aplikácia opäť prístup k internetu.

V rámci tejto práce som mal možnosť po prvýkrát navrhnúť a zhotoviť PCB pre vstavané zariadenie, a rovnako tak aj navrhnúť a vytlačiť 3D model krabičky pre toto zariadenie. Som rád, že som sa pri návrhu mohol obrátiť na skúsených a ochotných odborníkov, ktorý mi predali mnoho znalostí a vedomostí, ktoré určite ďalej využijem pri realizácií svojich následujúcich projektov.

Systém je možné v blízkej budúcnosti ďalej rozšíriť o zaujímave funkcionality. Mohlo by ísť napríklad o prepojenie s domácimi asistentmi a smart home riešeniami. Zaujímavou myšlienkou, s ktorou som sa počas vývoja zariadenia pohrával, je možnosť aktualizácie firmwaru zariadenia pomocou mobilnej aplikácie. Do zariadenia by tak bolo možné pridávať ďalšie funkcionality jednoducho pomocou bluetooth bez nutnosti odpojenia zariadenia a opätovnej inštalácie. Jednou z ďalších možností využitia tohto systému, ktorá nie je vôbec časovo náročná je schopnosť zaznamenávať aj vstupy z iných meracích zariadení, ktoré rovnako ako elektromer disponujú optickým výstupom. V neposlednom rade je vďaka vývoju aplikácie v multiplatformovom frameworku Flutter možnosť ďalej rozšíriť podporu mobilnej aplikácie aj na iné systémy, než len Android a to najmä na konkurenčný systém iOS.

Zhotovený systém chcem sám ďalej využívať na zaznamenávanie spotreby. Počas práce na tomto projekte som sa často stretával s pozitívnymi reakciami a záujmom o výsledný produkt od rodinných príslušníkov a kamarátov, preto si myslím, že by stálo za to na tom ďalej pokračovať a predstaviť produkt, ktorý by bolo možné ďalej predávať. Ako dobrá cesta ďalej sa preto javí crowndfundingová kampaň, na ktorej by som vedel zistiť záujem ľudí a vyzbierať finančné prostriedky na ďalší vývoj produktu.

Literatúra

[1] *Bluetooth Low Energy* online. Dostupné z: https://www.nordicsemi.com/Products/Bluetooth-Low-Energy. [cit. 2023-1-4].

[2] *Espressif modules* online. Dostupné z: https://www.espressif.com/en/products/modules. [cit. 2023-1-3].

[3] *Flutter architectural overview* online. Dostupné z: https://docs.flutter.dev/resources/architectural-overview. [cit. 2023-1-24].

[4] *GL5537 datasheet* online. Dostupné z: https://datasheetspdf.com/pdf-file/756912/ETC/GL5537/1. [cit. 2023-1-14].

[5] *Going global at Google Pay with Flutter* online. Dostupné z: https://flutter.dev/showcase/google-pay. [cit. 2022-12-10].

[6] *Meet Android Studio* online. Dostupné z: https://developer.android.com/studio/intro.

[7] *Memory footprint optimization* online. Dostupné z: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/test_and_optimize/optimizing/memory.html#app-memory.

[8] *NRF5x Power Management Tutorial*. Dostupné z: https://embeddedcentric.com/lesson-14-nrf5x-power-management-tutorial/.

[9] *Optimizing for Footprint* online. Dostupné z: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/zephyr/develop/optimizations/footprint.html.

[10] *Power optimization recommendations* online. Dostupné z: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/test_and_optimize/optimizing/power_general.html.

[11] *React Native* online. Dostupné z: https://reactnative.dev/. [cit. 2023-1-4].

[12] *Swift: The powerful programming language that is also easy to learn.* online. Dostupné z: https://developer.apple.com/swift/. [cit. 2023-1-4].

[13] *Who is using React Native?* online. Dostupné z: https://reactnative.dev/showcase.

[14] *Why a New Architecture* online. Dostupné z: https://reactnative.dev/docs/the-new-architecture/why. [cit. 2023-1-24].

[15] *Vybavení pro měření elektrické energie (AC) - Všeobecné požadavky, zkoušky a zkušební podmínky - Část 11: Elektroměry*. ČSN EN IEC 62052-11 ED.2. Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, august 2021.

[16] *Wi-Fi 6E expands Wi-Fi into 6 Ghz* online. 2021. Dostupné z: https://www.wi-fi.org/download.php?file=/sites/default/files/private/Wi-Fi_6E_Highlights_202112.pdf. [cit. 2022-12-04].

[17] *Jaká je průměrná spotřeba elektřiny u rodinného domu?* online. 2022. Dostupné z: https://www.cez.cz/cs/clanky/elektrina/jaka-je-prumerna-spotreba-elektriny-u-rodinneho-domu-174046.

[18] Energetický regulační úřad. *Metodika zavádění smart meteringu v ČR* online. 2022. Dostupné z: https://eru.gov.cz/sites/default/files/obsah/prilohy/metodika-zavadeni-sm-v-cr.pdf.

[19] Fanstel. *Bluetooth 5.3 Modules BT832, BT832A, BT832F, BT832AF Ver 2.17* online. November 2022. Dostupné z: https://www.fanstel.com/s/BT832-p-Product-Specifications.pdf. [cit. 2024-5-2].

[20] Flaherty, N. Espressif moves exclusively to RISC-V. online, 2022. Dostupné z: https://www.eenewseurope.com/en/espressif-moves-exclusively-to-risc-v/. [cit. 2023-1-23].

[21] Gans, W.; Alberini, A. a Longo, A. Smart meter devices and the effect of feedback on residential electricity consumption: Evidence from a natural experiment in Northern Ireland. *Energy Economics*, 2013, zv. 36, s. 729–743. ISSN 0140-9883.

[22] Gomez, C.; Oller Bosch, J. a Paradells, J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors (Basel, Switzerland)*, December 2012, zv. 12, s. 11734–53.

[23] Google. *Android's Kotlin-first approach*. Dostupné z: https://developer.android.com/kotlin/first.

[24] Nordic Semiconductor. *Explore our product portfolio* online. Dostupné z: https://www.nordicsemi.com/Products.

[25] Nordic Semiconductor. *NRF5340 SoC Product Brief Version 3.0* online. Dostupné z: https://nsscprodmedia.blob.core.windows.net/prod/software-and-other-downloads/product-briefs/nrf52840-soc-v3.0.pdf. [cit. 2023-1-4].

[26] Nordic Semiconductor. *NRF52832 Product Specification v1.9* online. December 2023. Dostupné z: https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.9.pdf. [cit. 2024-5-2].

[27] Shaw, S. *React Native Architecture — Old Vs New* online. 2022. Dostupné z: https://javascript.plainenglish.io/react-native-new-architecture-old-vs-new-d0130f42bc79. [cit. 2023-1-24].

[28] Statista. *Market share of mobile operating systems worldwide from 2009 to 2024, by quarter* online. 2024. Dostupné z: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.

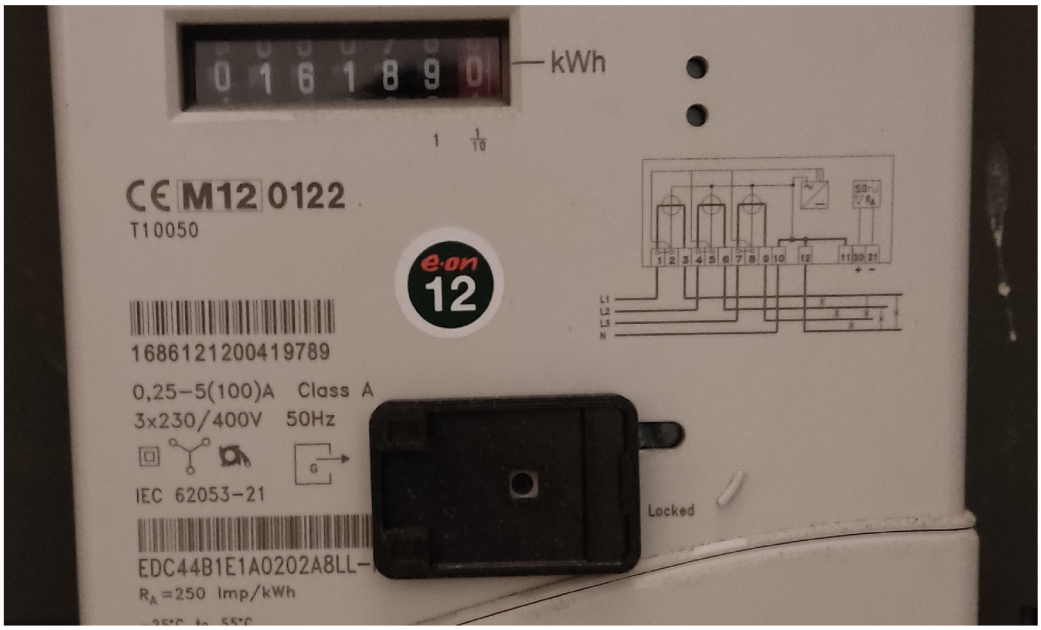
[29] Texas Instruments Incorporated. *Low Power Comparator for Signal Processing and Wake-Up Circuit in Smart Meters* online. 2017. Dostupné z: https://www.ti.com/lit/an/snva808/snva808.pdf. [cit. 2023-1-17].

[30] The Apache Software Foundation. *Apache Cordova: Overview* online. Dostupné z: https://cordova.apache.org/docs/en/11.x/guide/overview/index.html. [cit. 2023-1-11].

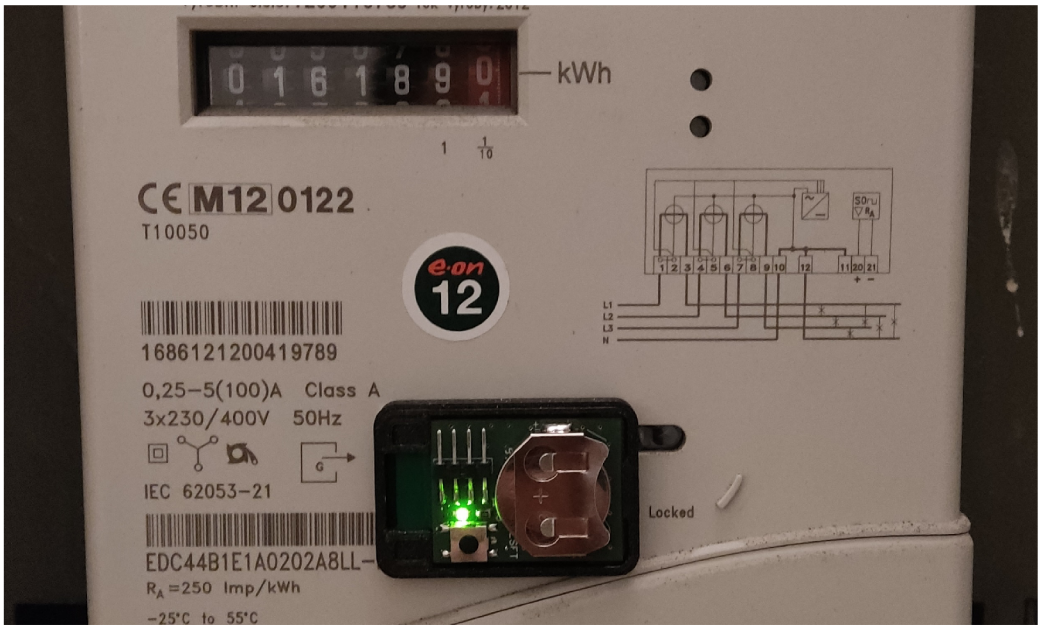
[31] Vašíček, Z. *Analýza technického provedení a uplatnitelnosti technologie Angle of Arrival*. 2022. 41 s. Dostupné z: https://www.fit.vut.cz/research/publication/12722.

Príloha A

Nasadenie vstaveného zariadenia na elektromer



Obr. A.1: Na elektromere osadená spodná časť krabičky. Cez otvor krabičky je možné vidieť optický výstup elektromera.

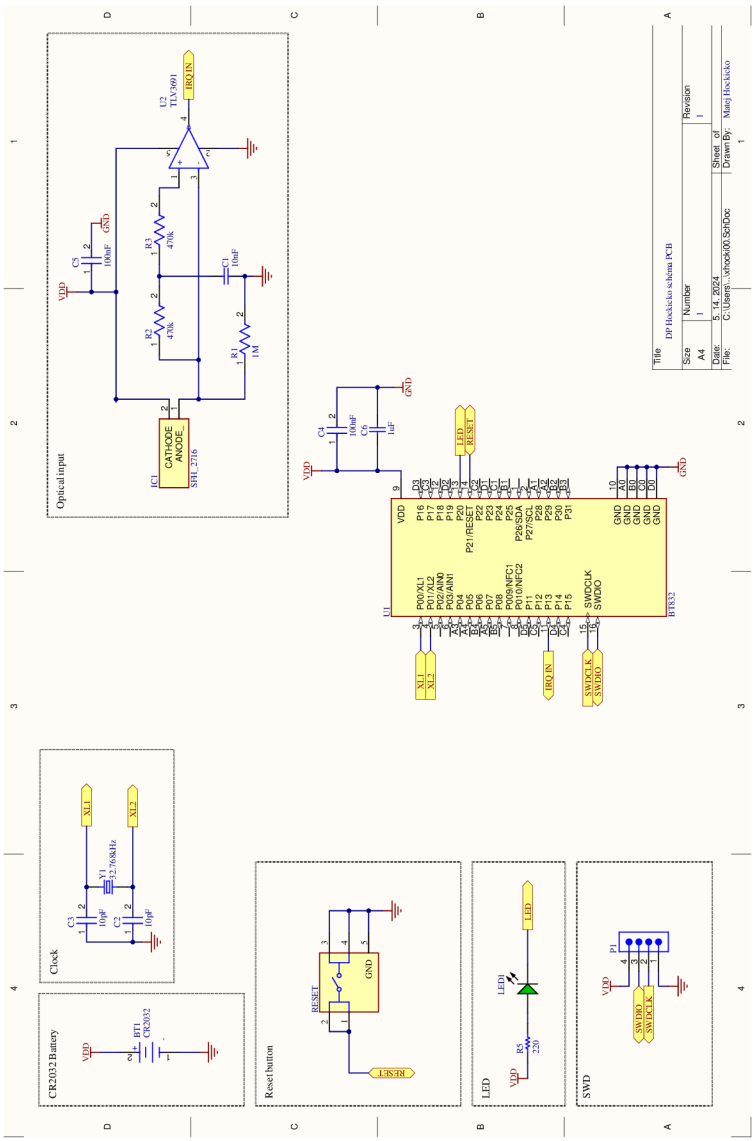


Obr. A.2: Vstavané zariadenie umiestnené do spodnej časti krabičky. Zapnutá LED signalizuje pripravenosť zariadenia na konfiguráciu.

Obr. A.3: Elektromer s osadeným vstavaným zariadením v krabičke.

Príloha B

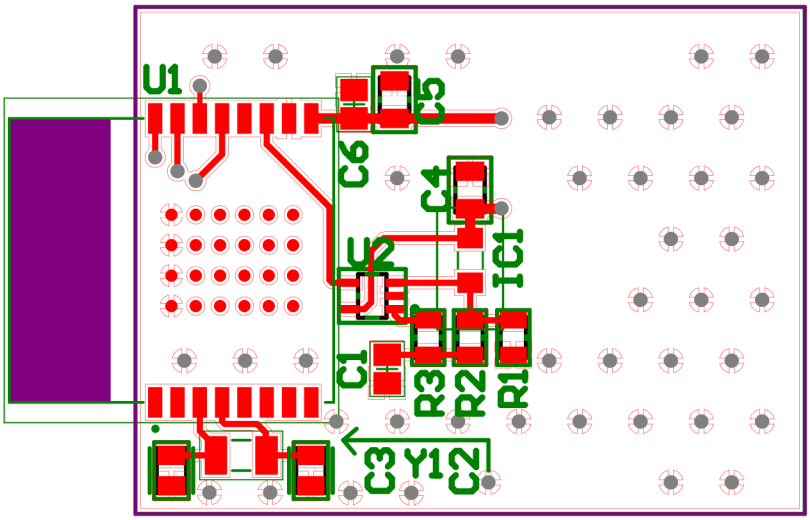
PCB schéma



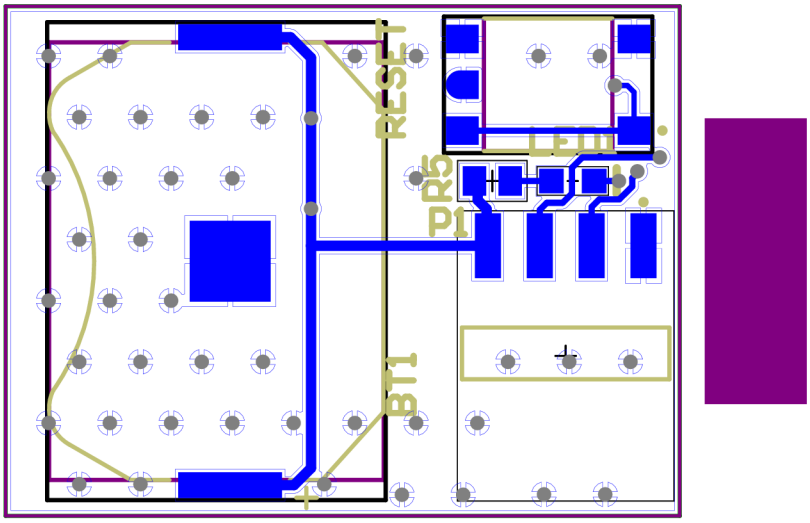
Obr. B.1: Schéma zapojenia vstavaného zariadenia

Príloha C

PCB layout



Obr. C.1: Vrchná časť PCB



Obr. C.2: Spodná časť PCB

Príloha D

Obsah SD karty

- **3D** - adresár so zdrojovými súbormi na vyhotovenie krabičky pre vstavané zariadenie
- **Embedded** - adresár so zdrojovým programom firmweru vstavaného zariadenia
- **Latex** - adresár so zdrojovými súbormi textu práce
- **Mobile** - adresár so zdrojovými kódmi na zostavenie mobilnej aplikácie
- **PCB** - adresár zdrojových súborov pre PCB
- **app.apk** - mobilná aplikácia
- **firmware.hex** - firmware vstavaného zariadenia
- **xhocki00.pdf** - text práce
- **xhocki00_print.pdf** - text práce určený na tlač