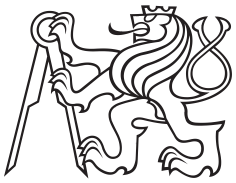
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Navigace uvnitř budov pomocí VLC

Jan Vomastek

Školitel: doc. Ing. Stanislav Vítek, Ph.D.
Květen 2024

Poděkování

Nejprve bych chtěl velmi poděkovat doc. Ing. Stanislavu Vítkovi, Ph.D. za jeho cenné rady a pomoc při tvorbě této práce. Dále bych chtěl poděkovat svým blízkým za jejich podporu a trpělivost během tvorby této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských kvalifikačních prací.

V Praze, 15. května 2024

Abstrakt

Cílem této bakalářské práce je provést rešerši, návrh a implementaci systému pro navigaci ve vnitřních prostorách pomocí viditelného světla. Systém má za cíl být schopen zachytit signály ze světelných vysílačů, zpracovat tyto signály, určit polohu uživatele a tuto informaci předat uživateli. Práce analyzuje problém navigace ve vnitřních prostorách a popisuje řešení, která lze pro tuto navigaci využít. Práce popisuje celkový průběh vývoje systému a dokumentuje veškeré kroky, které vedly k vytvoření funkčního prototypu.

Klíčová slova: Visible Light Communication (VLC), Visible Light Positioning (VLP),navigace, vnitřní navigace, viditelné světlo, K-Nearest Neighbors (KNN), predikční model

Školitel: doc. Ing. Stanislav Vítek, Ph.D.

Abstract

The aim of this bachelor thesis is to conduct research, design and implement a system for indoor navigation using visible light. The system aims to be able to capture signals from light transmitters, process these signals, determine the user's location and provide this information to the user. The thesis analyzes the problem of indoor navigation and describes solutions that can be used for this navigation. The thesis describes the overall development of the system and documents all the steps that led to the creation of a functional prototype.

Keywords: Visible Light Communication (VLC), Visible Light Positioning (VLP), navigation, indoor navigation, visible light, K-Nearest Neighbors (KNN), prediction model

Title translation: Indoor navigation based on VLP

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V dnešní době se můžeme spolehlivě s přesností na jednotky metrů celosvětově navigovat pomocí družicových navigačních systémů. Mezi tyto navigační systémy patří například GPS nebo GLONASS [4]. Tyto systémy jsou již nedílnou součástí našich životů, bez které si již neumíme představit běžný život. Avšak jedním z větších problémů těchto systémů je, že noposkytují dostatečnou přesnost polohy při pohybu v budovách. Tento jev nastává především z důvodu útlumu signálu. Tomuto problému se dosud univerzálně nedá vyhnout a vznikají tedy alternativy jak navigaci ve vnitřních prostorách budov alternativně uskutečnit. Na toto téma vzniklo již několik prací, které se snaží navigaci ve vnitřních prostorách řešit [5][6].
Jeden z těchto systémů, přesněji navigaci ve vnitřních prostorách pomocí viditelného světla, budu v této práci rozebírat a to především z důvodu, že se jedná o systém, který je velmi přesný a má potenciál využívat již existující infrastrukturu [7].

1.1 Motivace

Motivací pro tuto práci je především zájem o vytvoření uživatelsky přívětivého systému pro navigaci ve vnitřních prostorách. Tento systém je navrhnut tak, aby byl co nejvíce funkční a zároveň co nejméně náročný na infrastrukturu a tedy aby bylo reálné používání v praxi co nejvíce usnadněno.

1.2 Cíle práce

Cílem práce je provést rozsáhlou rešerši problému, vytvořit platformu pro lokalizaci pomocí viditelného světla s využitím 3 světel, které budou vysílat přesně definované sekvence. Tyto sekvence budou následně zachyceny pomocí mikrokontroleru, který bude tyto sekvence následně předávat do externího zařízení, aplikačního serveru, které bude tyto sekvence zpracovávat a na základě nich bude určovat polohu uživatele. Zjištěná poloha bude následně předána uživateli.

Kapitola 2
Rešerše

Tato kapitala se bude zabývat rešerší v oblasti určování polohy v budovách, jak je tento problém řešen v praxi a jaké technologie se dají využít pro tento účel. Dále se bude zabývat technologiemi, které se dají využít pro určování polohy v budovách a jaké jsou jejich výhody a nevýhody.

K určování polohy vě většině případů využíváme družicové polohové systémy, tyto systémy jsou však v budovách, v podzemí, v tunelech nebo v jinak zastavěných oblastech velmi nepřesné a někdy zcela nefunkční. Tento jev nastává jelikož signál je značně oslaben a dochází k vícecestnému šíření. V této kapitole se budu zabývat alternativami k těmto systémům, které mohou být použity pro určování polohy v budovách.

2.1 Úvod do problematiky

S nárůstem technologií se lidé stále více spoléhají na navigační systémy, které jim pomáhají určovat polohu. I proto se v posledních letech začalo více řešit určování polohy v budovách. Ať už se člověk nachází v rozsáhlém obchodním centru, nemocnici nebo na letišti, vždy se může ocitnout v situaci, kdy se potřebuje dostat z bodu A do bodu B bez předchozí znalosti prostoru. Tento problém se dá řešit několika způsoby. Jedním z nich je využití cedulových navigačních systémů, které jsou však většinou nepřesné a nejsou schopny uživatele efektivě a rychle dostat přesně do cíle. Toto je především problém u míst, kde se člověk potřebuje rychle dostat z místa na místo, například v nemocnicích nebo na letištích [8].

Dalším důležitým problémem je, že tento systém není uzpůsoben pro osoby, které mají omezené schopnosti orientace. Tomuto problému se v dnešní době zabývalo již několik projektů. Jedním z nich je projekt navigace pomocí akustických orientačních majáků. Tyto majáky však mají několik nevýhod. Jednou z nich je to, že se jedná o systém, který je závislý na infrastruktuře, která musí být v budově nainstalována. Další nevýhodou je, že se jedná o systém, který je závislý na zvuku, který je v budovách často rušený a může docházet k jeho zkreslení [9].

Obě tato řešení by se dnes dalo označit za zastaralé. Proto se nabízí další možnosti, které se dají využít pro určování polohy v budovách. Nejčastěji se jedná o systémy, které využívají Bluetooth, Wi-Fi nebo RFID.

Dalšími možnostmi jsou systémy, které využívají viditelné světlo. Všechny tyto systémy mají své výhody a nevýhody. V této práci se budu zabývat pouze systémem, který využívá viditelného světla z důvodu, že se jedná o systém, který je přesný a má potenciál využívat již existující infrastrukturu. [10]

2.2 Technologie pro určování polohy v budovách

K určování polohy v budovách se dají využít různé technologie. Mezi nejčastěji používané patří Bluetooth, Wi-Fi, RFID a viditelné světlo. Každá z těchto technologií se snaží co nejpřesněji a nejspolehlivěji splnit všechna kritéria pro určení polohy v budovách. Každá z těchto technologií k dané problematice přistupuje odlišnými přístupy.

2.2.1 Wi-Fi

Navigace pomocí technologie Wi-Fi je založena na zjištění síly přijatého signálu *(označováno také jako RSS)* díky metodě 'fingerprinting' [11]. Tato technologie se v poslední době stává stále více rozšířenou, především z důvodu vysokého pokrytí již vybudovanou infrastrukturou. Nejnovějším trendem se u Wi-Fi stala technologie Wi-Fi Round Trip Time, která je schopna zjistit vzdálenost připojeného zařízení od vysílače díky zjištění zpoždění od odeslání do přijetí signálu. Tato funkce je pro použití vnitřního systému pro zjištění polohy výhodnější než RSS, protože dokáže zajistit vyšší přesnost. Tato technologie se stále více objevuje u nově vyráběných Wi-Fi vysílačů, dá se tedy předpokládat, že se v příštích letech stane velmi rozšířenou a tedy i lukrativní možností pro vnitřní zjištění polohy [12].

2.2.2 BLE

Navigace pomocí technologie Bluetooth Low Energy (BLE) je založena na zjištění síly přijatého signálu *(označováno také jako RSS)* ze tří a více vysílačů. Tato technologie je výhodná především z důvodu nízké spotřeby energie a poměrně jednoduché implementace. Tato technologie je schopna při strategickém umístění vysílačů dosáhnout přesnosti přibližně 5 metrů, což může pro určité případy být dostatečné. S příchodem Bluetooth verze 5.1 však bude možno získat vyšší přesnosti a to především z důvodu nové funkce 'Angle of Arrival', která je schopna zjistit úhel, pod kterým je signál přijat. Mezi některé nevýhody technologie navigace pomocí BLE patří potřeba vybudování infrastruktury, která je schopna pokrýt celý prostor, a pro některé případy nedostatečná přesnost [13].

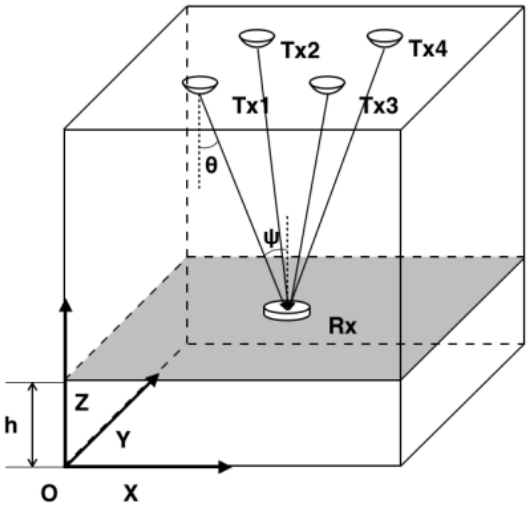
2.2.3 RFID

Technologie RFID je založena na odesílání rádiových vln mezi čtečkou a čipem. Tato technologie je využívána především v průmyslu pro sledování zboží a

předmětů. Tato technologie se dá, při dostatečně dobrém rozmmístění čipů, využít i pro určování polohy ve vnitřních prostorách. Mezi nevýhody této technologie patří náročnost na infrastrukturu, protože je potřeba vybudovat dostatečné pokrytí, které nemá žádné jiné využití. Další z nevýhod je velmi nízká vzdálenost, na kterou je možno zachytit signál (maximálně do 1 metru) [14].

2.2.4 Navigace pomocí viditelného světla

Systém pro určování polohy pomocí viditelného světla využívá LED světel, které během běžného provozu vysílají sekvence, které pouhým okem nejsou viditelné. Tyto sekvence musí být pro množinu světel unikátní. Tuto unikátnost lze dosáhnout pomocí různých technik, například pomocí vysílání světelných signálů na různých frekvencí nebo i změnou amplitudy či fáze. Tyto sekvence jsou následně zachyceny pomocí fotodiody, která je umístěna v navigačním zařízení. Systém jako takový netrpí většinou nedostatků, jako dříve zmíněné technologie. Na rozdíl od předchozích technologií má technologie viditelného světla tu výhodu, že může využívat všudypřítomnou infrastrukturu světelných zdrojů. Popisovaný systém je zobrazen na obrázku 2.1, kde 'Tx1' - 'Tx4' jsou vysílače světelného signálu a 'Rx' je přijímačem těchto signálů.



Obrázek 2.1: Návrh systému pro určování polohy pomocí viditelného světla z [1]

2.3 Termíny

Mezi pojmy, se kterými se často při práci s navigačními systémy, systémy pro určování polohy a viditelným světlem setkáme, patří:

- **Indoor navigation** - Jedná se o navigaci ve vnitřních prostorách. Tento

termín se používá pro označení navigace v budovách, kde je signál GPS nedostatečný.

- **Visible Light Positioning** - Jedná se o systém, který využívá viditelného světla pro určování polohy.
- **LED** - Light Emitting Diode, jedná se o diodu, která vydává světlo, když jí prochází elektrický proud.
- **Photodiode** - Jedná se o diodu, která při dopadu světla vytváří elektrický proud.
- **Visible Light Communication** - Jedná se o komunikaci pomocí viditelného světla.
- **RSS** - Received Signal Strength, jedná se o sílu přijatého signálu.
- **Fingerprinting** - Jedná se o metodu, která využívá sílu přijatého signálu pro určení polohy.
- **Wi-Fi Round Trip Time** - Jedná se o technologii, která využívá zpoždění signálu pro určení polohy.
- **RFID** - Radio-Frequency Identification, jedná se o technologii, která využívá rádiové vlny pro komunikaci mezi čtečkou a čipem.
- **Bluetooth** - Jedná se o bezdrátovou technologii, která využívá rádiové vlny pro komunikaci mezi zařízeními.

Kapitola 3
Analýza

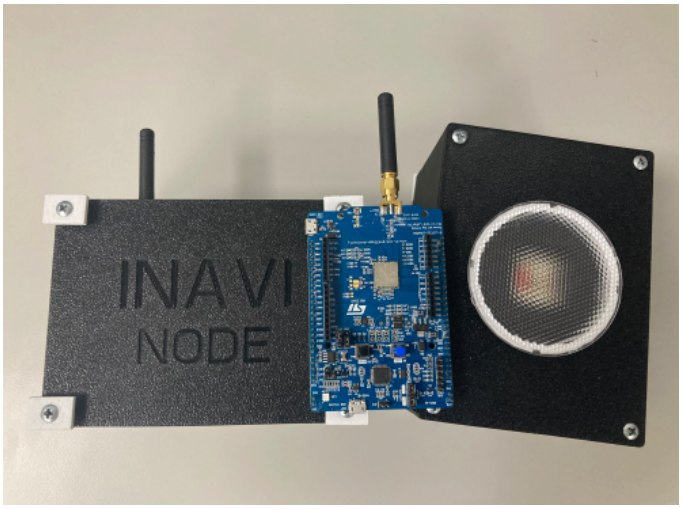
Tato kapitola se zabývá analýzou řešení z práce [3], ze které vycházím, určením funkčních a nefunkčních požadavků na výsledek projektu a technologií potřebných pro vývoj vlastního řešení.

3.1 Analýza stávajícího stavu

Stávající platforma pro řízení robota je pro použití člověkem příliš robustní a zároveň moc specifická pro obecné užití, například člověkem nebo jakýmkoliv jiným robotem či zařízením. Proto bude nutné v rámci této práce vytvořit novou platformu, která bude schopna komunikovat s přijímačem a zpracovávat data z něj i jinak než přes stávající platformu, která je velmi závislá na přítomnosti robota.

3.1.1 Světelné vysílače

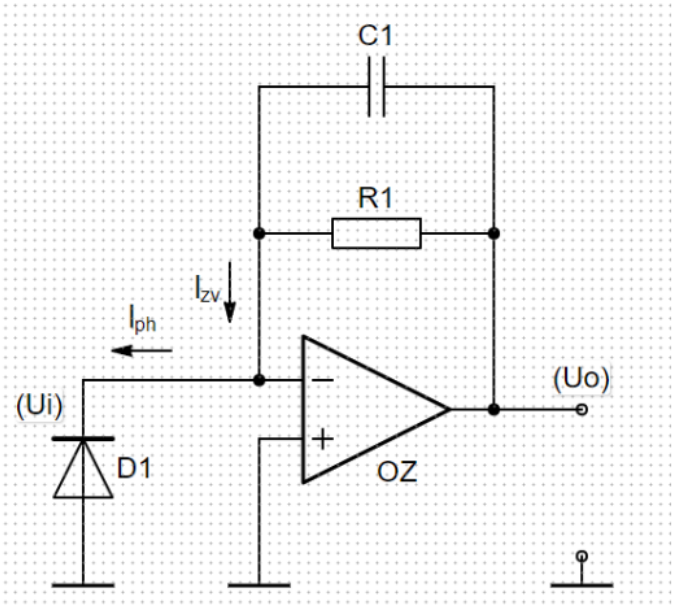
Na základě teorie z minulé kapitoly byla sestrojena světla, která slouží jako vysílače. Tyto vysílače pochází z prací [15] a [2]. Práce se zabývaly především konstrukcí hardware světel a také softwarovou částí. Světla jsou ovládána pomocí LoRaWAN přes webovou aplikaci. Tímto způsobem se dá na světlech řídit vysílání signálů. Světla vysílají signály pomocí PWM nebo pomocí sinusového průběhu. Dále lze u PWM nastavit frekvenci a duty cycle. U sinusového průběhu lze nastavit frekvenci, amplitudu a fázi. Maximální frekvence se pohybuje v řádu jednotek kHz 3.1. Světla jsou pro využití v této práci vhodně vytvořena, jak softwarově, tak sestavením fyzické části světel. Avšak s přihlédnutím k možnostem testování, které jsou během vypracovávání práce k dispozici, bylo nutno zajistit vytvoření adaptéru pro vysílače, díky kterým bude možné vysílače přidělat na standardizované držáky pro studiové osvětlení [16].



Obrázek 3.1: Světlo použité pro vysílání signálu z práce [2]

3.1.2 Přijímač světelného signálu

Přijímač, který vychází z práce [3], byl sestaven na desce s mikrokontrolerem ESP32. Tento mikrokontroler je vhodný pro přijímání signálů, protože má vestavěný Wi-Fi modul. Díky tomu je možné přijímané signály odesílat do externího zařízení. Přijímač byl doplněn o fotodiodu BPW34. Fotodioda, generuje elektrický proud, který je závislý na intenzitě dopadajícího světla. Pro zpracování AD převodníkem mikrokontroléru bylo nutné tento proud převést na napětí. K tomuto účelu byl použit obvod znázorněn na obrázku 3.2.



Obrázek 3.2: Schéma přijímače z práce [3]

Jedná se o transimpedanční zesilovač, který převádí proud na napětí. Na obvodu se nachází zmíněná fotodioda, která je připojena na invertující vstup operačního zesilovače. Dále se zde nachází rezistor s hodnotou 680kΩ. Nakonec se zde nachází kondenzátor s hodnotou 22pF. Dále tento obvod obsahuje operační zesilovač LM358, napájený napětím 3.3V. Tento obvod již nebudu více podrobněji popisovat, jelikož je již popsán v dříve zmíněné práci, ze které jsem čerpal [3]. Zároveň je ale tento obvodu v celém svém rozsahu použit jako část obvodu, který je používán v této práci pro přijímání signálů.

3.2 Požadavky

Požadavky popisují potřeby uživatele, které musí být splněny, aby byl výsledek projektu úspěšný. Požadavky se dělí na funkční a nefunkční. Funkční požadavky popisují, co systém dělat má, zatímco nefunkční požadavky popisují, jak by měl systém fungovat [17].

3.2.1 Funkční požadavky

Funkční požadavky popisují konkrétní funkce, které by měl systém splňovat. Na základě analýzy byly stanoveny následující funkční požadavky:

Nahrávání

- FR00: Zvolení nahrávané pozice - Uživatel bude moci zvolit pozici, pro kterou chce nahrávat data.
- FR01: Nahrávání dat - Uživatel bude moci nahrávat data z přijímače do souboru.

Navigace

- FR02: Získání polohy - Uživatel bude moci získat polohu přijímače.
- FR03: Zvuková navigace - Uživatel bude moci získat zvukovou navigaci.

Vytvoření predikčního modelu

- FR04: Vytvoření predikčního modelu - Uživatel bude moci vytvořit predikční model.

3.2.2 Nefunkční požadavky

Nefunkční požadavek je ohraničení sytému, která se týkají typů řešení, které splňují funkční požadavky. Například výkon, spolehlivost, dostupnost, bezpečnost, responzivita a jiné [17]. Na základě analýzy byly stanoveny následující nefunkční požadavky:

- QR00: Rychlá odezva - Rozhraní aplikace rychle reaguje na uživatelské vstupy.

- QR01: Intuitivní ovládání - Aplikace má intuitivní ovládání a nevyžaduje speciální školení ani studium manuálu.
- QR02: Spolehlivost spojení - Aplikace je schopna udržet spojení mezi přijímačem a serverem a jejich spojení je spolehlivé.
- QR03: Výkon - Aplikace je schopna zpracovat data v reálném čase a zobrazit je uživateli.
- QR04: Nízká spotřeba energie - Aplikace je uzpůsobena pro běh na zařízeních s nízkou kapacitou baterie a spotřebuje co nejméně energie.

3.3 Případy užití

Velmi důležitou součástí analýzy je popis případů užití a také popis posloupnoti událostí, ke kterým dochází při interakcích systému a uživatele za účelem dosažení určitého cíle [17]. Diagram všech případů užití a veškeré sekvenční diagramy lze nalézt v Příloze A.1 této práce.

3.4 Technologie řešení

Jak již bylo zmíněno v dřívějších kapitolách, pro řešení tohoto problému se používají různé technologie. Pro zvolenou technologii VLP je nutné zvolit vhodný mikrokontrolér, který bude schopen přijímat signály a zároveň je zpracovávat. Dále je nutné zvolit vhodný programovací jazyk, který bude schopen zpracovávat data z mikrokontroléru a to zároveň co nejlépe a nejrychleji. Dále je nutné zvolit vhodný způsob komunikace mezi mikrokontrolérem a externím zařízením, které bude zpracovávat data z mikrokontroléru. Hlavními aspekty, které je nutné zohlednit při výběru technologií, jsou:

- Rychlost zpracování dat
- Spolehlivost komunikace
- Dostupnost knihoven
- Jednoduchost použití
- Nízká spotřeba energie

Především je nutné zohlednit, že výsledné řešení bude používáno bez použití platformy robota a bude používáno pouze pro navigaci ve vnitřních prostorách a to především pro osobní použití.

3.4.1 Analýza technologií mikrokontrolérů

Pro vytvoření přijímače signálů je nutné zvolit vhodný mikrokontrolér, který bude schopen přijímat signály a zároveň je zpracovávat. Pro tyto účely jsou analyzovány mikrokontroléry ESP32 [18] a Raspberry Pi Pico W [19].

ESP32

ESP32 je mikrokontrolér, který je vhodný pro přijímání signálů, protože má vestavěný Wi-Fi modul. Díky tomu je možné přijímané signály odesílat do externího zařízení. ESP32 má také vestavěný AD převodník, který je vhodný pro převod signálů z fotodiod na digitální signály. ESP32 má také dostatečný výkon pro zpracování signálů v reálném čase. ESP32 má také dostatečnou paměť pro uložení dat. Zároveň má také dostatečně nízkou spotřebu energie pro běh na baterii a je také dostatečně malý pro použití v mobilním zařízení [18].

Raspberry Pi Pico W

Raspberry Pi Pico W je mikrokontrolér, který obsahuje velmi podobné vlastnosti jako ESP32. Ze společných vlastností lze zmínit vestavěný Wi-Fi modul, který je vhodný pro přijímání signálů, vestavěný AD převodník, který je vhodný pro převod signálů na digitální signály, dostatečný výkon pro zpracování signálů v reálném čase, dostatečnou paměť pro uložení dat, dostatečně nízkou spotřebu energie pro běh na baterii a dostatečně malý rozměr pro použití v mobilním zařízení [19]. Raspberry Pi Pico W má ale narozdíl od ESP32 několik výhod které jsou pro vývoj tohoto projektu klíčové. Jedná se o dostupnost knihoven pro tento mikrokontrolér, které jsou vhodné pro zpracování signálů z fotodiod a zároveň jsou vhodné pro zpracování signálů v reálném čase. Dále je Raspberry Pi Pico W velmi jednoduchý na použití a jeho dokumentace je velmi dobře zpracovaná.

3.4.2 Analýza programovacích jazyků

Pro zpracování dat z mikrokontroléru a zároveň pro zpracování dat v reálném čase je nutné zvolit vhodný programovací jazyk. Také je nutné zvolit programovací jazyk, který bude možno použít pro vytvoření programu pro externí zařízení, které bude zpracovávat data z mikrokontroléru.

Micropython

Micropython je programovací jazyk, který je vhodný pro zpracování dat z mikrokontroléru a zároveň je vhodný pro zpracování dat v reálném čase. Micropython je také vhodný pro vytvoření programu pro mikrokontrolér, který bude zpracovávat data z fotodiod a zároveň bude zpracovávat data v reálném čase. Tento programovací jazyk mu především výhody v jednoduchosti použití a v dostupnosti knihoven pro tento jazyk.

C++

C++ je velmi rozšířený programovací jazyk, který je vhodný pro vytváření programů od nejmenších mikrokontrolérů až po velmi složité a výkonné počítače. Tento programovací jazyk je také základem pro všechny programovací

jazyky, které vychází z jazyku Python, tedy i zmiňovaný Micropython. Veškeré funkce, které jsou popsány u sekce Micropython, jsou tedy taktéž dostupné i pro C++. Avšak jednou z nevýhod použití C++ oproti Micropythonu je hned několik. První z nich může být složitější syntaxe pro psaní programů, dále to také může být složitější a méně intuitivní použití knihoven a v neposlední řadě také nutnost kompilace programu před jeho spuštěním. Jednou z výhod použití C++ však může být vyšší rychlost při běhu programu a méně náročné požadavky na zdroje zařízení.

3.4.3 Analýza knihoven

Pro použití obou programovacích jazyků bude nutno použít knihovny, které budou schopny zpracovávat data z mikrokontroléru a zároveň budou schopny zpracovávat data v reálném čase. Pro oba programovací jazyky byly analyzovány knihovny, které jsou vhodné pro zpracování signálů z fotodiod a zároveň jsou vhodné pro celkovou funkčnost programu.

QTCPSocket

Pro komunikaci mezi zařízenímy je na straně externího zařízení možno použít knihovnu QTCPSocket. Tato knihovna je schopna zpracovávat data poslaná pomocí TCP protokolu a zároveň je schopna data odesílat pomocí TCP protokolu. Tato klíčová schopnost je nutná pro zajištění spolehlivé komunikace mezi mikrokontrolérem a externím zařízením. Knihovna QTCPSocket je dostupná pro oba zmíněné programovací jazyky [20].

Scikit-learn

Scikit-learn je knihovna, která je vhodná pro vytvoření predikčního modelu pomocí různých metod. Jednou z těchto metod je například metoda K-neaerest neighbors, která je vhodná pro vytvoření predikčního modelu pro zpracování signálů z fotodiod. Tato knihovna je schopna zpracovávat nasbíraná a ohodnocená data. Dáte tato knihovna také obsahuje funkce pro rozdělení dat na trénovací a testovací data, pro vytvoření predikčního modelu a pro vyhodnocení predikčního modelu. Tato knihovna je dostupná především pro programovací jazyk Python [21].

Ostatní knihovny

Mezi knihovny, které byly analyzovány jako vhodné pro tento projekt, patří také knihovny:

- NumPy - Knihovna pro zpracování dat, uspořádání dat a jejich množná editace a zpracování [22].
- Pandas - Knihovna pro čtení dat z různých formátů, především z formátu CSV [23].

- Pickle - Knihovna pro serializaci a deserializace objektů ze statndardní knihovny [24].

3.5 Shrnutí

V této kapitole byla provedena analýza stávajícího stavu, byly stanoveny funkční a nefunkční požadavky na výsledek projektu a byly analyzovány technologie potřebné pro vývoj vlastního řešení. Byly analyzovány mikrokontroléry ESP32 a Raspberry Pi Pico W, programovací jazyky Micropython a C++ a knihovny QTCPSocket, Scikit-learn a další potřebné knihovny. Na základě této analýzy byly zvoleny technologie pro vývoj tohoto projektu. Z analýzy vyplývá, že pro vývoj tohoto projektu bude nejlepší možností použít mikrokontrolér Raspberry Pi Pico W, programovací jazyk Micropython a knihovny QTCPSocket a Scikit-learn a to především kvůli jejich dostupnosti, jednoduchosti použití, vzájemné kompatibilitě a vysokému standardu jejich dokumentace a podpory.

Kapitola 4
Návrh řešení

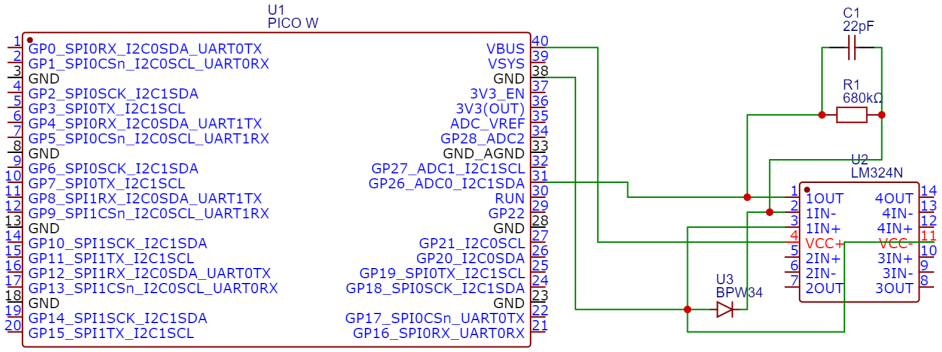
Tato kapitola obsahuje řešení navrženého systému, související diagramy, obvody a popis a ukázku prototypu uživatelského rozhraní.

4.1 Popis řešení

V minulých kapitolách byla provedena analýza nástrojů potřebných pro vývoj a byly zvoleny vhodné technologie pro vývoj systému pro navigaci ve vnitřních prostorách. Vzhledem k charakteru řešeného produktu bylo rozhodnuto pro vývoj vlastního modulu pro příjem signálu, jeho zpracování a odeslání na server, příjem zpracované informace a prezentace dat uživateli. Tento modul bude obsahovat mikrokontrolér, obvod pro příjem signálu [3], tři ovládací tlačítka a 16 znakový displej. Celková architektura systému je rozdělena na dvě části, a to na MCU modul a serverovou část. MCU modul bude zajišťovat příjem signálu, jeho zpracování, odeslání na server a prezentaci dat uživateli. Serverová část bude zajišťovat příjem dat, zpracování dat, predikce polohy a odeslání zjištěné polohy zpět na MCU modul. Jak již bylo zmíněno, serverová část bude realizována v jazyku Python a část mikrokontroleru bude realizována v jazyce MicroPython.

Návrh obvodu pro VLC

Návrh obvodu pro příjem signálu pomocí viditelného světla 4.1 se skládá z již dříve zmíněného obvodu 3.2, který je složen z fotodiody BPW34 a operačního zesilovače LM358. Dále tento obvod obsahuje rezistor R1 s hodnotou 680kΩ a kondenzátor C1 s hodnotou 22pF. Narozdíl od předchozího zapojení je tento obvod navíc připojen na mikrokontrolér Raspberry Pi Pico W, který zajišťuje přívod napětí (pomocí pinů VBUS a GND) a zároveň i zpracování signálu z obvodu (pomocí pinu GP26_ADC0). Tento obvod je díky napojení na AD převodník mikrokontroleru Raspberry Pi Pico W, který je schopný pracovat až rychlostí 500kS/s [25] , je schopen uspokojit požadavky na rychlost příjmu signálu. Vzhledem k parametrům vysílače, který je schopen vysílat signál až o frekvenci 10kHz, je potřeba, aby AD převodník byl schopen zpracovat signál alespoň o dvojnásobné frekvenci, tedy musí být schopen zpracovávat data při

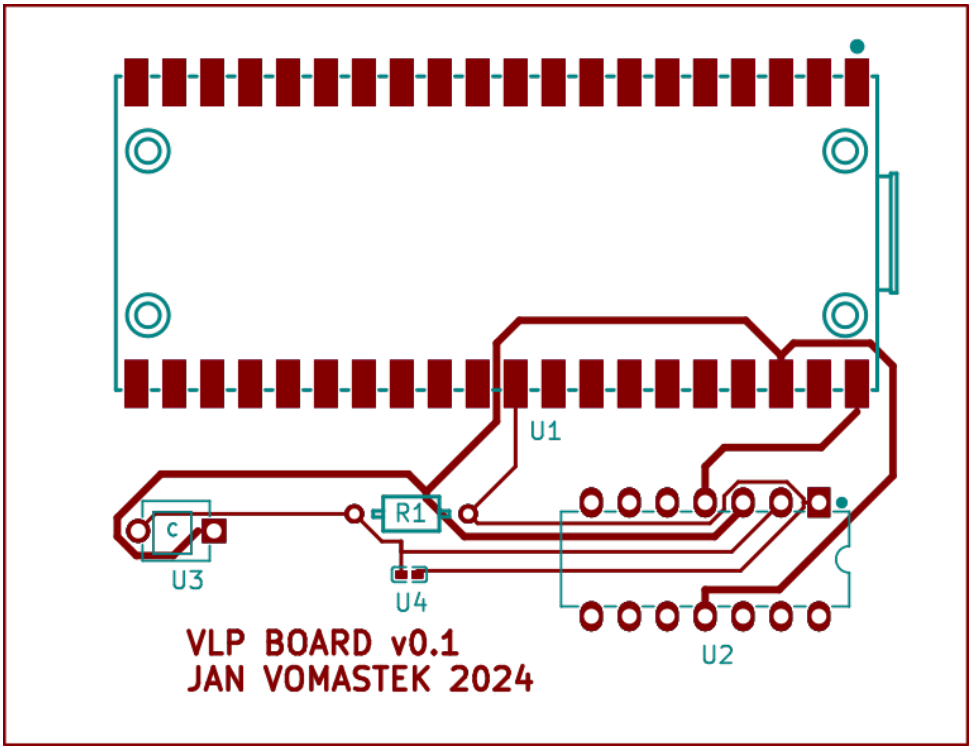


Obrázek 4.1: Návrh obvodu pro příjem signálu pomocí viditelného světla

frekvenci alespoň 20kHz [26]. AD převodník mikrokontroleru Raspberry Pi Pico W je tedy schopen tuto potřebu zcela uspokojit.

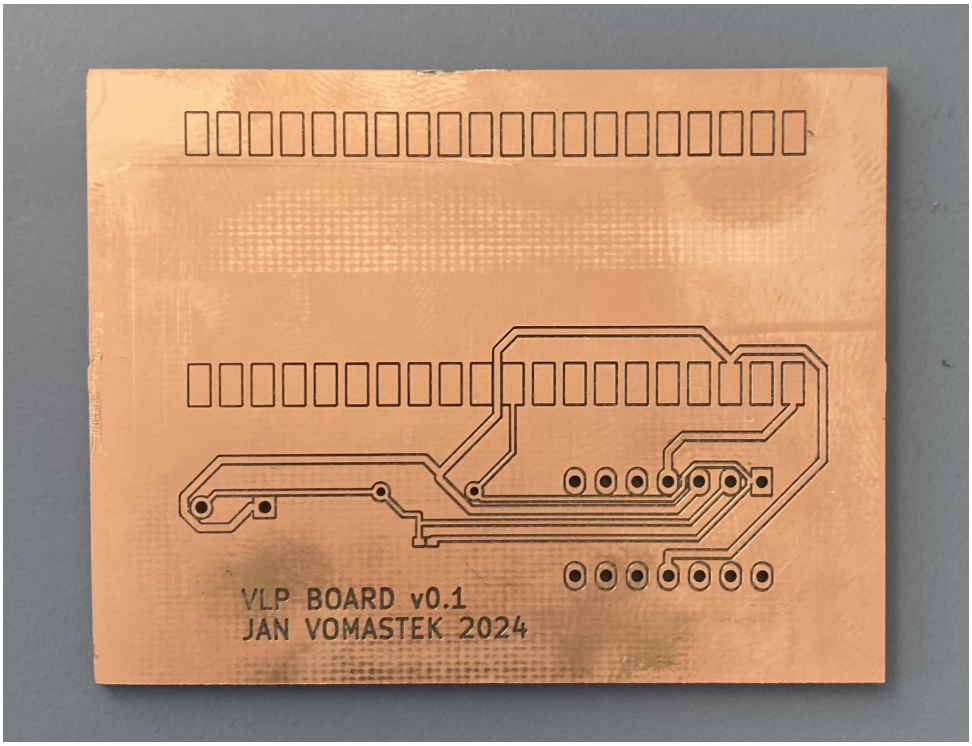
Návrh a osazení PCB obvodu pro VLC

První verze PCB byla navržena pro ověření správnosti návrhu obvodu pro příjem signálu pomocí viditelného světla a jeho funkčnosti. Zároveň bylo cílem ověřit, zda je možné tento obvod osadit a zda jsou šířky vodičů dostatečně dimenzovány pro možnost osazení a následného použití. Na obrázku 4.2 je zobrazen návrh a osazení první verze PCB. Tento návrh byl vytvořen v programu EasyEDA, ze kterého byl následně vyexportován pro použití v programu KiCad [27]. V programu KiCad byl následně tento návrh doupraven a připraven pro výrobu. Po vyexportování návrhu do formátu Gerber, který byl následně použit přímo pro výrobu.



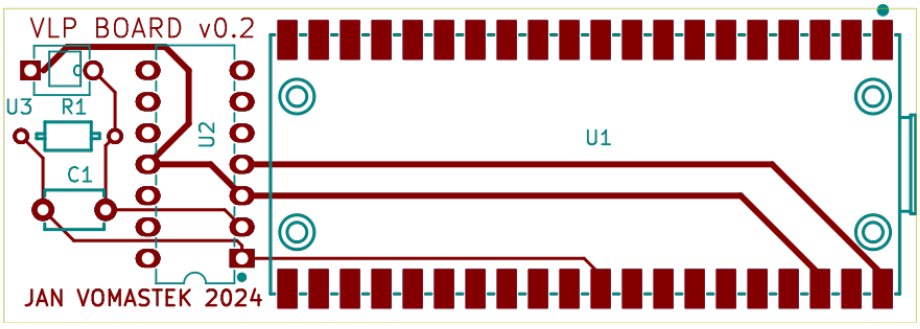
Obrázek 4.2: Návrh a osazení PCB obvodu pro příjem signálu pomocí viditelného světla

Po vyexportování návrhu do formátu Gerber byl tento návrh importován do programu LPKF CircuitPro, který je dodáván k frézovacímu stroji LPKF ProtoMat E44. V tomto programu lze návrh dále upravovat, přiřadit mu přislušné vrstvy, nastavit parametry frézování a nakonec přímo z programu spustit samotné frézování na frézovacím stroji [28]. Při prvním pokusu o frézování byl proces nastavený na frézování měděné vrstvy, která má být následně na určitých místech chemicky odstraněna. Výrobek, který po tomto procesu bez chemické úpravy vznikl, je zobrazen na obrázku 4.3.



Obrázek 4.3: První verze vyfrézovaného PCB pomocí frézovacího stroje LPKF ProtoMat E44

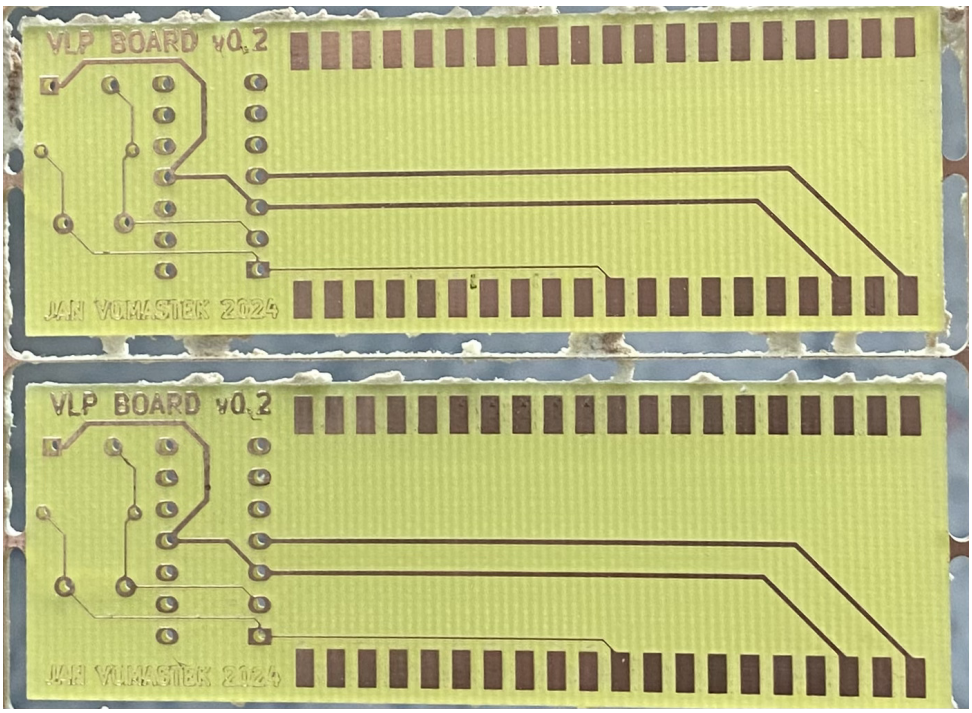
Po zjištění správnosti návrhu, funkčnosti obvodu a případných nedostatků, byl návrh upraven podle zjištěných nedostatků. Jedním z těchto nedostatků bylo například příliš dlouhé propojení mezi jednotlivými komponenty, které bylo způsobeno špatným návrhem umístění jednotlivých komponent. Dalším nedostatkem bylo špatně zvolené balení kondenzátoru C1, který byl zvolen jako SMD, což způsobilo, že námi zvolený typ kondenzátoru nebylo možné osadit na desku. Tyto problémy byly adresovány a návrh byl upraven. Výsledný, tedy druhý návrh, je zobrazen na obrázku 4.4.



Obrázek 4.4: Návrh a osazení PCB obvodu pro příjem signálu pomocí viditelného světla - druhá verze

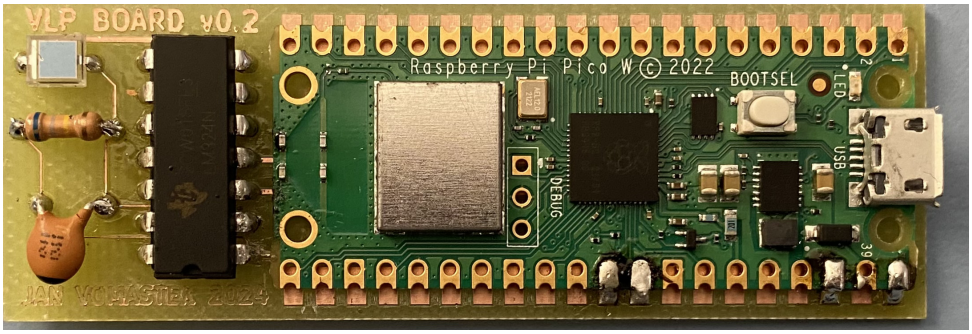
Tento návrh nejen adresuje všechny nedostatky první verze, ale zároveň je podstatně menší a kompaktnější, čímž je více vhodný pro osazení a použití.

Po vyexportování návrhu do formátu Gerber byl tento návrh opět importován do programu LPKF CircuitPro, kde byl připraven pro výrobu. Posledním nedostatkem předchozí verze bylo, že byl návrh vytvořen pro frézování obrysu měděné vrstvy pro následné chemické odstranění mědi. Tento postup je v praxi často používaný, ale vzhledem k možnostem pracoviště a dostupným materiálům bylo rozhodnuto o změně postupu. Postup byl změněn na frézování celé měděné vrstvy, kde nejsou vedeny žádné vodiče. Tento postup byl pro prototypování zvolen jako mnohem jednodušší a efektivnější. Výsledný výrobek, který vznikl po frézování druhé verze návrhu, je zobrazen na obrázku 4.5.



Obrázek 4.5: Druhá verze vyfrézovaného PCB pomocí frézovacího stroje LPKF ProtoMat E44

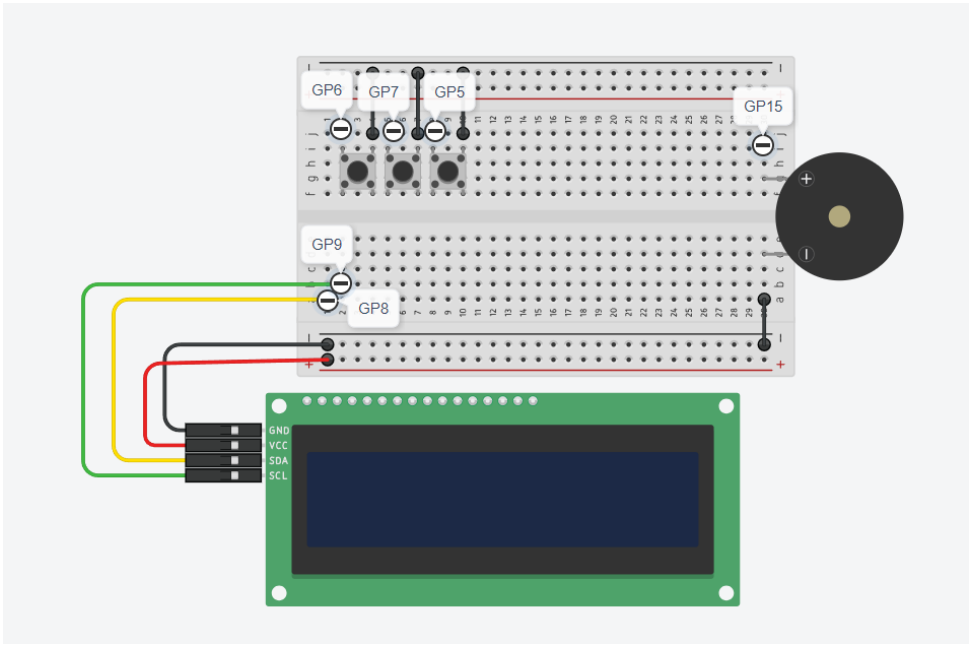
V neposlední řadě bylo provedeno osazení desky, kde byly osazeny všechny komponenty, které byly součástí návrhu. Osazení desky bylo provedeno ručně, kde byly jednotlivé komponenty osazeny na desku a následně byly připájeny. Výrobek byl následně otestován a bylo zjištěno, že návrh a osazení desky jsou korektní a nezávadné. Osazená deska je zobrazena na obrázku 4.6.



Obrázek 4.6: Druhá verze vyfrézovaného a osazeného PCB pomocí frézovacího stroje LPKF ProtoMat E44

Připojení periferií k mikrokontroléru

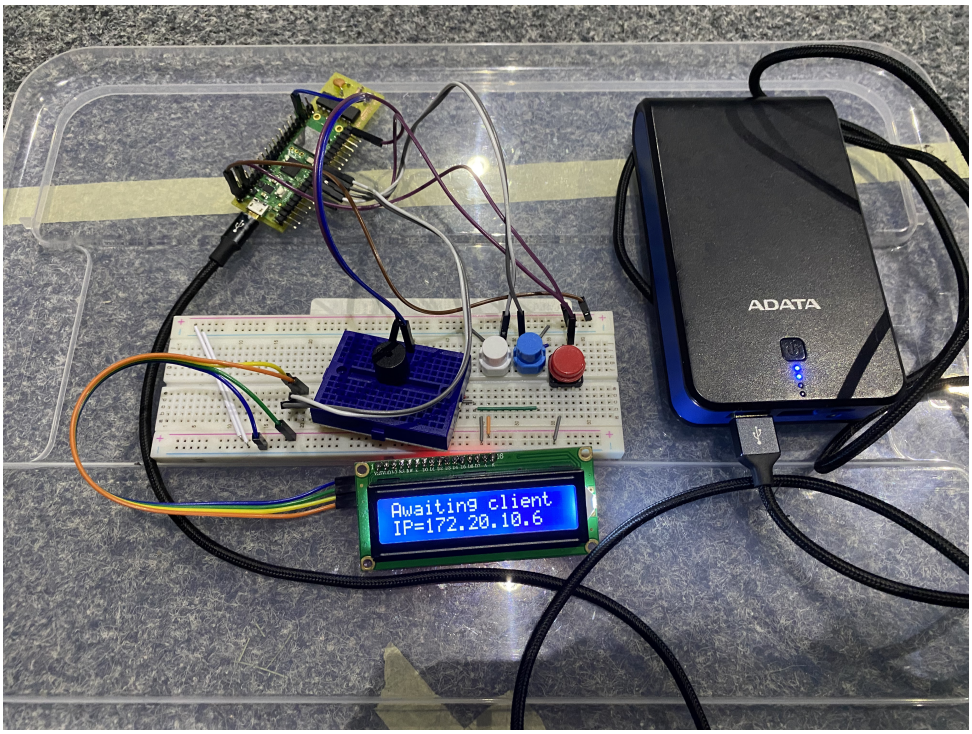
Pro zobrazení informací uživateli a ovládání mikrokontroléru byly připojeny tři ovládací tlačítka a 16 znakový displej. Další periferie, která byla připojena k mikrokontroléru, byl piezo reproduktor, který slouží k signalizaci uživateli. Připojení těchto periferií je pro vývoj a testování klíčové, avšak pro produkční verzi by bylo možné některé tyto periferie vynechat. Například pokud by se jednalo o produkt pro zrakově postižené, bylo by možné vynechat displej i tlačítka a ponechat pouze reproduktor. Ostatní periferie hrají klíčovou roli především při nahrávání pozic a následném zpracování dat, z čehož vyplývá, že pro použití uživatelem tyto periferie nejsou nutné. Návrh zapojení periferií k mikrokontroléru je zobrazen na obrázku 4.7.



Obrázek 4.7: Návrh zapojení periferií k mikrokontroléru

Toto zapojení bylo následně pro potřeby testování a dalšího vývoje pro-

pojeno s mikrokontrolérem Raspberry Pi Pico W. Zapojení by mohlo být pro další používání integrováno přímo do desky s mikrokontrolérem, avšak pro potřeby testování a vývoje bylo zvoleno externí zapojení, které umožňuje snadnou výměnu a případnou úpravu periferií. Výsledné zapojení periferií k mikrokontroléru je zobrazeno na obrázku 4.8.



Obrázek 4.8: Výsledné zapojení periferií k mikrokontroléru

Architektura aplikace

Vyvýjená aplikace je rozdělena na dvě části, a to na serverovou část a část mikrokontroléru. Serverová část je realizována v jazyce Python a část mikrokontroléru je realizována v jazyce MicroPython. Obě tyto části jsou následně propojeny pomocí TCP Socket komunikace, která je realizována jako Full Duplex komunikace. Tato komunikace tedy umožňuje obousměrný přenos dat mezi serverem a mikrokontrolérem. Serverová část představuje především roli zpracování dat predikce polohy, vytvoření predikčního modelu z nahraných dat a následné odeslání zjištěné polohy zpět na mikrokontrolér.

Vzhledem k charakteristice popsané aplikace tedy poněkud jasně vyplývá, že musí být zvolena architektura klient-server. Tato architektura je výhodná především z důvodu úspory zdrojů na straně klienta, tedy mikrokontroléru, a zároveň umožňuje snadnou škálovatelnost a rozšiřitelnost aplikace. Je však třeba brát v potaz, že tato architektura vyžaduje stálé připojení mezi klientem a serverem, což může být v některých případech nevýhodou [29].

Sekvenční diagram

V rámci návrhu aplikace byly připraveny sekvenční diagramy, poposující logiku zvoleného řešení.

Na diagramu A.2 je zobrazena logika nahrávání pozic. Tento proces je zahájen stisknutím tlačítka na periferii mikrokontroléru. Po stisknutí tlačítka je zahájeno nahrávání pozic, které je následně odesláno na server, kde jsou zpracovány a uloženy do dočasného souboru ve formátu CSV. Po uložení dat je na mikrokontroler odeslána zpráva o úspěšném uložení dat, která je předána uživateli pomocí některé z periferií mikrokontroléru.

Další diagram A.3 zobrazuje logiku navigace. Tento proces je zahájen interakcí uživatele s periferiemi mikrokontroléru. Po zahájení procesu jsou nejprve načtena data z AD převodníku, která jsou následně odeslána na server. Na serveru jsou tato data zpracována a na základě těchto dat je zjištěna poloha uživatele. Tato poloha je následně odeslána zpět na mikrokontrolér, kde je zpracována a předána uživateli pomocí periferií mikrokontroléru.

Poslední diagram A.4 zobrazuje logiku vytvoření predikčního modelu. Tento proces je zahájen odesláním požadavku na vytvoření predikčního modelu na server. Po obdržení tohoto požadavku je na serveru zahájeno vytvoření predikčního modelu, který je následně uložen do souboru. Po uložení predikčního modelu je odeslána zpráva o úspěšném uložení, která je předána uživateli pomocí periferií mikrokontroléru.

4.2 Odhad náročnosti implementace

Náročnost implementace a testování tohoto řešení se bude hodnotit ve člověkodnech, značeno Man-Day(MD), kde jeden MD je definován jako jeden den práce jednoho člověka [30].

Mikrokontroler-backend. Implementace a testování mikrokontroléru bude nejnáročnější částí celého řešení. Tato část bude vyžadovat nejvíce času a zdrojů. Bude složena z implementace čtení dat z AD převodníku, odeslání dat na server, příjem dat ze serveru a prezentaci dat uživateli (pomocí logů či sériovému monitoru). Celková náročnost této části je odhadnuta na **20 MD**.

Server-backend. Implementace serverové části přímo souvisí s implementací mikrokontroléru. Tato část bude vyžadovat méně času a zdrojů než implementace mikrokontroléru, avšak bude vyžadovat souběžnou práci s implementací mikrokontroléru. Tato část se bude skládat především z implementace příjmu dat z mikrokontroléru, zpracování dat, predikce polohy, uložení dat, vytvoření predikčního modelu a odeslání zjištěné polohy zpět na mikrokontrolér. Celková náročnost této části je odhadnuta na **15 MD**.

Mikrokontroler-periferie. Pro implementaci a testování periferií bude potřeba kompletnost obou předchozích částí. Tato část se skládá především z implementace ovládání tlačítek, zobrazení dat na displeji, signalizace pomocí

reproduktoru a následné napojení těchto periferií na backend část mikrokontroleru. Tato část se může na první pohled zdát jako nejjednodušší, avšak bude vyžadovat částečné zasahování do obou předchozích částí. Tímto se celková náročnost této části ztíží. Celkem je odhadnuta na **10 MD**.

Celková náročnost implementace a testování tohoto řešení je odhadnuta na **45 MD**. To znamená, že pro implementaci a testování tohoto řešení bude potřeba až **360 hodin** práce jednoho člověka.

4.3 Prototyp uživatelského rozhraní

Pro vývoj softwaru je velmi přínosné vytvořit prototyp uživatelského rozhraní, které umožní prozkoumat možnosti aplikace ještě před samotným vývojem. Vytvořením prototypu získáme lepší představu, zda rozhraní, které bylo navrženo, splňuje uživatelské potřeby. Dále také můžeme identifikovat případné nedostatky nebo mylné představy o navrhované aplikaci [17].

Prototyp uživatelského rozhraní byl vytvořen v programu Figma. Tento program umožňuje vytvářet prototypy uživatelského rozhraní, které jsou interaktivní a umožňují uživatelům prozkoumat možnosti aplikace ještě před samotným vývojem. Prototyp uživatelského rozhraní je vytvořen jako low-fidelity prototyp, což znamená, že je vytvořen pouze jako návrh rozhraní, který je následně možné upravit a vylepšit [31].

Prototyp začátku uživatelského rozhraní je vyobrazen v příloze B.1. Tento prototyp zobrazuje hlavní menu aplikace s vybraným polem 'Recordings'. Vybraná položka je označena znakem '*' na začátku vybraného řádku. Zvolením položky se zvoleným kurzorem a následným potvrzením tlačítkem se uživatel dostane do dalšího menu, které je zobrazeno na obrázku B.2. Toto menu zobrazuje možnosti pro nastavení frekvence nahrávání. Následným potvrzením tlačítkem se uživatel dostane do módu pro zvolení frekvence, který je zobrazen na obrázku B.3. Zde si uživatel může vybrat frekvenci nahrávání pomocí tlačítek. Po zvolení frekvence se uživatel dostane do módu pro spuštění nahrávání, který je zobrazen na obrázku B.4. Po stisknutí tlačítka se nahrávání spustí a uživatel je informován o úspěšném spuštění nahrávání.

Další částí prototypu je menu pro navigaci, které je zobrazeno na obrázku B.5. Toto menu zobrazuje možnosti pro navigaci. Zvolením položky se zvoleným kurzorem a následným potvrzením tlačítkem se uživatel dostane do dalšího menu, které je zobrazeno na obrázku B.6. Toto menu zobrazuje možnosti pro spuštění navigace. Potvrzením tlačítkem se spustí navigace a uživateli je v poli 'Frequency' zobrazena frekvence podle pozice, ve které se uživatel nachází.

Poslední částí prototypu je menu pro predikci polohy, které je zobrazeno na obrázku B.7. Toto menu zobrazuje možnosti pro predikci polohy. Zvolením

položky se zvoleným kurzorem a následným potvrzením tlačítkem se uživatel dostane na obrazovku, která zobrazuje informaci o vytváření predikčního modelu. Tato obrazovka je zobrazena na obrázku B.8.

Kapitola 5

Implementace

V této kapitole budou popsány jednotlivé kroky implementace navrženého systému. Vývoj aplikace byl rozdělen na tři části - vývoj backend aplikace pro mikrokontroler, vývoj backend aplikace pro server a vývoj části aplikace pro rozhraní mikrokontroleru.

5.1 Backend mikrokontroleru

V rámci implementace backend aplikace pro mikrokontroler byl použit jazyk MicroPython verze 1.21. K vývoji bylo použito několik knihoven, které byly do mikrokontroleru manuálně nahrány pomocí vývojového prostředí Thonny.

5.1.1 Získání dat z AD převodníku

Pro získání dat z AD převodníku byla použita část aplikace z projektu 'Pi Pico ADC input using DMA and MicroPython' [10]. Jak již z názvu vyplývá, byla použita implementace získání dat z AD převodníku pomocí DMA (angl. Direct Memory Access, tj. přímý přístup do paměti). Tato implementace zajišťuje velmi vysokou rychlost čtení dat z AD převodníku, což je jedním z klíčových požadavků na tento projekt. Ukázka kódu 5.1 zobrazuje základní způsob získání dat z AD převodníku.

```
1  import time, array, uctypes, rp_devices as devs
2
3  def read_adc_multiple(NSAMPLES=256, RATE=20000):
4      # Multiple ADC samples using DMA
5      DMA_CHAN = 0
6      dma_chan = devs.DMA_CHANS[DMA_CHAN]
7      dma = devs.DMA_DEVICE
8
9      adc = devs.ADC_DEVICE
10     adc.FCS.EN = adc.FCS.DREQ_EN = 1
11     adc_buff = array.array('H', (0 for _ in range(NSAMPLES)))
12     adc.DIV_REG = (48000000 // RATE - 1) << 8
13     adc.FCS.THRESH = adc.FCS.OVER = adc.FCS.UNDER = 1
14
15     dma_chan.READ_ADDR_REG = devs.ADC_FIFO_ADDR
16     dma_chan.WRITE_ADDR_REG = uctypes.addressof(adc_buff)
17     dma_chan.TRANS_COUNT_REG = NSAMPLES
```

```
18
19      dma_chan.CTRL_TRIG_REG = 0
20      dma_chan.CTRL_TRIG.CHAIN_TO = DMA_CHAN
21      dma_chan.CTRL_TRIG.INCR_WRITE = dma_chan.CTRL_TRIG.IRQ_QUIET
        = 1
22      dma_chan.CTRL_TRIG.TREQ_SEL = devs.DREQ_ADC
23      dma_chan.CTRL_TRIG.DATA_SIZE = 1
24      dma_chan.CTRL_TRIG.EN = 1
25
26      while adc.FCS.LEVEL:
27          x = adc.FIFO_REG
28
29      adc.CS.START_MANY = 1
30      while dma_chan.CTRL_TRIG.BUSY:
31          time.sleep_ms(10)
32      adc.CS.START_MANY = 0
33      dma_chan.CTRL_TRIG.EN = 0
34      return adc_buff
```

Listing 5.1: Získání dat z AD převodníku pomocí DMA

Původní implementace byla upravena tak, aby bylo možné získat více vzorků z AD převodníku, což je pro Zpracování a odesílání dat na server klíčové z důvodu ušetření prostředků a zvýšení průchodnosti dat po síti mezi jednotlivými komponentami systému. Změnou kódu bylo zajištěno, aby bylo možné získat z AD převodníku 256 vzorků s frekvencí 20kHz pomocí DMA.

5.1.2 Zpracování dat a odeslání dat na server

Pro odeslání dat na server je možno použít několika módů. Tyto módy se dají zvolit před spuštěním odesílání dat. Prvním módem je odesílání dat pro nahrání dat do predikčního modelu. Tento mód odesílá data na server, kde jsou data zpracována a uložena do souboru. Druhým módem je odesílání dat pro získání predikce polohy. Tento mód odesílá data na server, kde jsou data zpracována a na základě těchto dat je provedena predikce polohy pomocí predikčního modelu. Třetím módem je odeslání požadavku na vytvoření predikčního modelu z dříve nahraných dat. Tento mód je použit pouze v případě, že je potřeba vytvořit nový predikční model. Třetí mód zároveň jako jediný ze všech zmíněných neodesílá žádná další data na server.

O zpracování dat z převodníku se stará funkce *timed_data_sender*, která je zobrazena v ukázce kódu 5.2. V této funkci je zajištěno, že se data z AD převodníku získají jako 256 vzorků zapsaných do pole bajtů, které jsou následně zaslány na server (viz. řádek 7 v ukázce kódu 5.2).

```
1       def timed_data_sender(self,timer):
2           self.idx += 2
3           .
4           .
5           .
6           if self.idx % 512 == 0 and self.sending_data:
7               self.client_socket.write(adc_dma.read_adc_multiple()
    )
8               received_data = self.client_socket.read()
9               if received_data:
```

```
10                  converted_data = int.from_bytes(received_data,'
    little')
11                  if converted_data < 101 and self.position == 0:
12                      print(converted_data)
13          .
14          .
15          .
```

Listing 5.2: Zpracování dat z AD převodníku

5.1.3 Získání predikce polohy

Pro získání predikce polohy se můžeme opět odkázat na ukázku kódu 5.2. Předchozí sekce již popisuje, jak jsou data z AD převodníku zpracována a odeslána na server a jak zajistit, aby server věděl, který mód je zvolen. Před začátkem odesílání dat je tedy nejprve odeslána informace o zvoleném módu. Toto odeslání je zajištěno pomocí kódu z ukázky 5.4, především z řádků 2 a 3.

```
1     def send_data(self):
2         self.client_socket.write(b'\x00\x00')
3         self.client_socket.setblocking(False)
4
5         self.timer.init(freq=20000, mode=machine.Timer.PERIODIC,
     callback=self.timed_data_sender)
```

Listing 5.3: Odeslání módu na server - Navigace

Po odeslání módu je zahájeno odesílání dat na server, kde jsou data zpracována, příslušně upravena a na základě těchto dat je provedena predikce polohy. Výsledek predikce polohy je následně odeslán zpět na mikrokontroler, kde je přijat. Pokud je predikce polohy jiná než v předchozím přijetí polohy, je tato poloha vypsána do konzole (5.2, řádky 8.-12.).

5.1.4 Vytvoření predikčního modelu

Dalším módem, který je možno zvolit, je mód pro vytvoření predikčního modelu. Tento mód je zvolen v případě, že je potřeba vytvořit nový predikční model. Vytvoření predikčního modelu je zajištěno pomocí kódu z ukázky 5.4.

```
1     def create_knn(self):
2         self.client_socket.write(b'\xFF\xFE')
```

Listing 5.4: Odeslání módu na server - Vytváření predikčního modelu

Po odeslání tohoto módu je na serveru zahájen proces vytvoření predikčního modelu. Z mikrokontroleru se již další data neodesílají a pouze se čeká na dokončení procesu serverem.

5.2 Backend serveru

Pro implementaci backend aplikace serveru byl použit jazyk Python ve verzi 3.11. Pro zpracování dat, komunikaci a další operace bylo použito několik

externích knihoven. Vývoj s použitím těchto knihoven probíhal převážně v odděleném virtuálním prostředí, které bylo vytvořeno pomocí nástroje **virtualenv**. Tato možnost umožňuje velmi snadno a rychle vytvořit izolované prostředí pro vývoj aplikace, které neovlivní ostatní aplikace vytvářené na stejném systému.

Závislosti, které byly použity pro vývoj backend aplikace serveru, jsou uvedeny v souboru **requirements.txt**, ze kterého mohou být zpětně nainstalovány pomocí správce balíčků **pip**.

5.2.1 Propojení s mikrokontrolerem

K propojení s mikrokontrolerem byla použita komunikace pomocí komunikační knihovny QTcpSocket. Tato knihovna je dostupná v rámci knihovny PyQt6 a umožňuje velmi snadnou a intuitivní implementaci komunikace mezi serverem a mikrokontrolerem [3][32]. Toto propojení je zajištěno jako první krok při spuštění serveru. Po spuštění serveru je zahájeno hledání mikrokontroleru v síti. Pokud je mikrokontroler nalezen, je zahájena komunikace s mikrokontrolerem a je zahájen proces získání dat z mikrokontroleru. Server je od připojení schopen přijímat data z mikrokontroleru a zpracovávat je dle zvoleného módu, případně je schopen měnit mód, ve kterém je komunikace a v neposlední řadě je schopen odesílat data zpět na mikrokontroler.

5.2.2 Zpracování a úprava dat z mikrokontroleru

Po přijetí dat z mikrokontroleru je zahájen proces zpracování a úpravy dat. Data jsou nejprve převedena z pole bajtů na pole nepřevedených čísel přímo z AD převodníku. Tato data jsou následně přepočítána na hodnoty v rozsahu 0-3.3V, což je rozsah, ve kterém reálně pracuje AD převodník. Po této úpravě jsou data upravena tak, aby výsledná křivka byla co nejvíce hladká a neobsahovala žádné šumy. Tato úprava je zajištěna pomocí Savitzky-Golay filtru, který je implementován v knihovně SciPy. Tento filtr je schopen velmi efektivně odstranit šumy z dat a zároveň zachovat významné body křivky. Zároveň je možné nastavit parametry tohoto filtru tak, aby bylo dosaženo co nejlepších výsledků [33]. Toto zpracování je prováděno vždy při přijetí nových dat z mikrokontroleru v režimu navigace, ale zároveň i v režimu nahrání dat do predikčního modelu.

5.2.3 Ukládání dat do souboru

Při režimu nahrávání dat do predikčního modelu jsou data přijata z mikrokontroleru, příslušně zpracována a vyfiltrovaná a následně pomocí knihovny Pandas uložena do souboru [23]. Data jsou ukládána do souboru ve formátu CSV v pořadí, ve kterém byla získána z mikrokontroleru. Pro každý řádek je ale navíc také do posledního sloupce přidána informace o tom, jaká frekvence polohy byla pro nahraný vzorek zvolena.

5.2.4 Vytváření predikčního modelu

Pro vytvoření predikčního modelu byla použita knihovna Scikit-learn, která je jednou z nejpoužívanějších knihoven pro vytváření predikčních modelů v Pythonu [34]. Tato knihovna obsahuje mnoho různých algoritmů pro vytváření predikčních modelů, ale pro tento projekt byl zvolen algoritmus K-Nearest Neighbors (KNN). Tento algoritmus byl zvolen z důvodu jeho jednoduchosti a zároveň schopnosti dosáhnout velmi dobrých výsledků v oblasti predikce polohy [35]. Pro vytvoření predikčního modelu je nejprve nutné načíst data z CSV souboru, který musel být předem vytvořen. Po načtení dat je zahájen proces vytváření predikčního modelu. Tento proces je zajištěn pomocí knihovny Scikit-learn a je zajištěno, aby byl vytvořen predikční model, který bude schopen předpovídat polohu na základě nahraných dat. Použitá data jsou rozdělena na trénovací a testovací data, kde trénovací data jsou použita pro vytvoření modelu a testovací data jsou použita pro ověření správnosti modelu. Po vytvoření modelu je tento model serializován a uložen do souboru, aby bylo možné tento model použít i v budoucnu bez nutnosti znovu vytvářet model, například pro potřeby predikce polohy.

5.2.5 Získání predikce polohy

Po uložení potřebných dat a vytvoření predikčního modelu je možné zahájit proces získávání predikce polohy na základě dat v reálném čase. Pro získání predikce polohy jsou nejprve získána data z mikrokontroleru, která jsou následně zpracována a vyfiltrovaná. Po této úpravě jsou data předána jako pole vstupních dat predikčnímu modelu. Predikční model pak následně na základě těchto dat provede predikci polohy a zjištěnou polohu uloží do proměnné. Tato poloha je následně v případě získání jiné polohy než v předchozím iteraci vypsána do konzole a zároveň je odeslána zpět na mikrokontroler, kde je přijata a dále zpracovávána. Tento proces získávání polohy je prováděn při každém přijetí nových dat z mikrokontroleru v režimu navigace a je tedy zajištěno, že je poloha získána v reálném čase.

5.3 Rozhraní mikrokontroleru

Pro ovládání mikrokontroleru a zobrazení dat bylo vytvořeno rozhraní, které sestává ze 3 tlačítek, LCD displeje, který obshuje 2 řádky po 16 znacích a reproduktoru. Propojení těchto periferií s již vytvořenou backend částí mikrokontroleru bylo zajištěno pomocí vytvoření State Machine, která je schopna zpracovávat stavy a přechody mezi nimi. Dále bylo pro ovládání a zobrazení dat vytvořeno několik tříd, které zajišťují dostatečné oddělení jednotlivých částí aplikace a zajišťují tedy lepší přehlednost a udržitelnost kódu.

5.3.1 LCD displej

Pro ovládání LCD displeje byla vytvořena třída, která zajišťuje zobrazení dat na LCD displeji. Zobrazení dat na displeji je zajištěno pomocí komunikace s LCD displejem s použitím protokolu I2C. Tato komunikace je zajištěna za použití dvou knihoven, *pico_i2c_lcd* a *lcd_api*, které jsou schopny zpracovávat data a zobrazovat je na displeji [36]. Pro zobrazení dat na displeji je nejprve nutné vytvořit objekt pro komunikaci pomocí I2C. Následně je možné pomocí tohoto objektu prozkoumat, zda je displej připojen a případně zjistit, jaká je jeho adresa pro komunikaci. Po zjištění těchto informací je inicializován objekt pro komunikaci s displejem a je zahájeno zobrazení dat na displeji. Pro zobrazení dat na displeji je poté potřeba pouze zvolit pozici kurzoru a zapsat požadované bloky textu na displej. Ukázka kódu 5.5 zobrazuje základní způsob zobrazení dat na displeji.

```
from machine import I2C, Pin
from pico_i2c_lcd import I2cLcd
i2c = I2C(0, sda=Pin(8), scl=Pin(9), freq=400000)

I2C_ADDR = i2c.scan()[0]
lcd = I2cLcd(i2c, I2C_ADDR, 2, 16)
while True:
  print(I2C_ADDR, "| Hex:",hex(I2C_ADDR))
  print()
  lcd.move_to(0,0)
  lcd.putstr("I2CAddress:"+hex(I2C_ADDR)+"\n")
  lcd.move_to(0,1)
  lcd.putstr("CircuitSchools.")
```

Listing 5.5: Zobrazení dat na LCD [36]

5.3.2 Tlačítka

Pro ovládání tlačítek byla vytvořena třída, která zajišťuje zpracování stisknutí tlačítek a následné provedení akce. Tato třída je schopna zpracovávat stisknutí tlačítek a na základě toho provádět různé akce. Pro zpracování stisknutí tlačítek je nejprve nutné vytvořit objekt pro komunikaci s tlačítky. Tento objekt je následně pomocí signálů přerušení schopen v periodických intervalech zjišťovat, zda bylo nějaké tlačítko stisknuto. Pokud bylo nějaké tlačítko stisknuto, je zavolána příslušná metoda, která zpracuje stisknutí tlačítka a provede příslušnou akci. Pro projekt jsou používána tři tlačítka, přičemž dvě z nich slouží pro posun kurzoru na displeji a zároveň pro změnu frekvence při jejím nastavování. Třetí tlačítko slouží pro potvrzení akce, která byla zvolena na displeji nebo pro uložení zvolené frekvence.

5.3.3 Reproduktor

Po získání polohy z predikčního modelu je tato poloha zobrazena na displeji a zároveň zazní přerušovaný zvukový signál, jehož délka přerušení je závislá na zjištěné poloze. Čím vyšší je hodnota frekvence odpovídající zjištěné

poloze, tím kratší je doba přerušení zvukového signálu. Respektive čím nižší je hodnota frekvence, tím delší je doba přerušení zvukového signálu.

Implementace reproduktoru byla zajištěna pomocí vytvoření objektu PWM z knihovny machine, která je již přítomna v základních knihovnách jazyka MicroPython. Tento objekt je schopen generovat signál s nastavitelnou frekvencí a délkou. Pro získání délky doby přerušení zvukového signálu byla vytvořena metoda, která na základě zjištěné polohy v požadovaném intervalu spouští, nebo ukončuje generování zvukového signálu. Příklad generování zvukového signálu je zobrazen v ukázce kódu 5.6.

```
def play_buzzer(self):
    self.buzzer.freq(500)
    self.buzzer.duty_u16(1000)

def stop_buzzer(self):
    self.buzzer.duty_u16(0)
```

Listing 5.6: Generování zvukového signálu

5.4 Struktura aplikace

Struktura aplikace je zobrazena ve stromové struktuře níže. Tato struktura zobrazuje jednotlivé soubory a složky, které byly vytvořeny v rámci implementace backend aplikace mikrokontroleru.

5.4.1 Struktura backend aplikace mikrokontroleru

```
pico_sw
├── lib ..................................... Složka pro knihovny
│   ├── adc_sender.py ....................... Manipulaci s ADC daty
│   ├── display_handler.py .................. Ovládání LCD
│   ├── lcd_api.py .......................... Podpůrná knihovna pro LCD
│   ├── menu_handler.py ..................... Ovládání tlačítek v menu
│   ├── pico_i2c_lcd.py ..................... Podpůrná knihovna pro LCD
│   └── rp_devices.py ....................... Podpůrná knihovna pro ADC
├── adc_dma.py .............................. Čtení dat z ADC pomocí DMA
└── main.py ................................. Hlavní aplikace
```

5.5 Shrnutí

V této kapitole byly popsány jednotlivé kroky implementace navrženého systému. Byly popsány jednotlivé části backend aplikace mikrokontroleru, backend aplikace serveru a rozhraní mikrokontroleru. Pro každou část byly popsány techniky a knihovny, které byly použity pro implementaci. Výsledná implementace byla vytvořena za účelem testování a ověření funkčnosti navrženého systému. Po řádném otestování a ověření funkčnosti tedy bude možné

přistoupit k případné úpravě implementace a vylepšení systému s ohledem na adresované nedostatky.

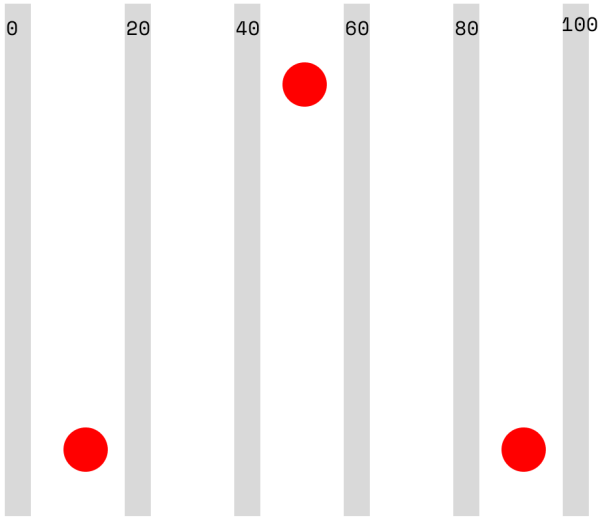
Kapitola 6
Testování

V rámci testování uživatelského rozhraní bylo prozkoumáno výsledné řešení aplikace s cílem ověřit, zda splňuje veškeré požadavky, které byly stanoveny v analýze. Testování bylo provedeno ve specializoavném testovacím prostředí, které bylo upraveno pro potřeby testování aplikace. Zároveň byly pro testování určeni uživatelé, kteří byli schopni ověřit funkčnost aplikace a zároveň poskytnout zpětnou vazbu na základě svých zkušeností s používání aplikace. V rámci testování byly vytvořeny testovací scénáře, které byly následně použity pro ověření funkčnosti aplikace. Tyto testovací scénáře byly uživateli prováděny bez jekékoliv asistence a bez jekéhokoliv předchozího seznámení s aplikací.

6.1 Testovací scénáře

Pro potřeby testování uživatelské aplikace byly vytvořeny tři testovací scénáře, které byly následně použity pro ověření funkčnosti aplikace. Testovací scénáře byly vytvořeny tak, aby pokrývaly veškeré funkce aplikace a zároveň byly dostatečně náročné na ověření funkčnosti aplikace. Testy byly provedeny na reálném zařízení s reálnými daty.

Pro účely testování bylo vytvořeno testovací prostředí, ve kterém byly všechny testovací scénáře provedeny. Testovací prostředí sestávalo ze tří zavěšených světelných zdrojů, které byly umístěny v místnosti do tvaru rovnostranného trojúhelníku. Každý světelný zdroj byl umístěn do jednoho vrcholu trojúhelníku ve výšce 2 metry nad zemí. Všechny světelné zdroje byly umístěny tak, aby jim nebylo žádným externím předmětem stíněno. Zároveň bylo zajištěno, aby světelné zdroje nebyly rušeny žádnými jinými světelnými zdroji. Rozmístění světelných zdrojů je zobrazeno na obrázku 6.1.



Obrázek 6.1: Rozmístění světelných zdrojů

6.1.1 Testovací scénář 1

Název: Nahrání dat z mikrokontroleru na server
Popis: Testovací scénář 1 ověřuje, zda je možné nahrát data z mikrokontroleru na server. Data jsou nahrána na server, kde jsou zpracována a korektně uložena do souboru ve formátu CSV.
Předpoklady: Spojení mezi mikrokontrolerem a serverem je navázáno a je možné odesílat data.
Na serveru je spuštěn serverový backend. Na serveru je dostupný soubor 'metoda1/train_data.csv'. Tento soubor neobsahuje žádná data.

Krok 1: V hlavní nabídce se pomocí tlačítek posuvu dostaňte na položku 'Recordings' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 1: Zobrazí se nabídka pro nahrávání. Kurzor je na položce 'Go back'. Dále je viditelná položka 'Frequency: xx ', kde xx je hodnota frekvence vzorkování.

Krok 2: Pomocí tlačítek posuvu se dostaňte na položku 'Frequency: xx ' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 2: Položka 'Frequency: xx ' se změní do režimu pro editaci hodnoty frekvence vzorkování. (Frequency:<XX>)

Krok 3: Pomocí tlačítek posuvu zvyšte hodnotu frekvence vzorkování na '60' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 3: Hodnota frekvence vzorkování se změní na '60'. Položka 'Frequency: 60' není v režimu pro editaci.

Krok 4: Pomocí tlačítek posuvu se dostaňte na položku 'Start' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 4: Zobrazí se animace nahrávání dat. Data jsou nahrána na server.

Krok 5: Na serveru zkontrolujte obsah souboru 'metoda1/train_data.csv'.
Výsledek kroku 5: Soubor 'metoda1/train_data.csv' obsahuje data z mikrokontroleru. Data jsou korektně uložena ve formátu CSV a poslední položka obsahuje námi zadanou hodnotu frekvence vzorkování '60'.

6.1.2 Testovací scénář 2

Název: Vytvoření predikčního modelu
Popis: Testovací scénář 2 ověřuje, zda je možné vytvořit predikční model na základě dat nahrávaných z mikrokontroleru.
Předpoklady: Spojení mezi mikrokontrolerem a serverem je navázáno a je možné odesílat data.
Na serveru je spuštěn serverový backend.
Na serveru je dostupný soubor 'metoda1/train_data.csv'. Tento soubor obsahuje data o nahrávkách z mikrokontroleru.

Krok 1: V hlavní nabídce se pomocí tlačítek posuvu dostaňte na položku 'Create KNN' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 1: Zobrazí se zpráva 'Creating KNN model Please wait...'.

Krok 2: Čekejte na dokončení vytváření predikčního modelu.
Výsledek kroku 2: Po dokončení vytváření predikčního modelu se zobrazí zpráva 'Model Created. Click left btn!'.

Krok 3: Na serveru zkontrolujte časové razítko souboru 'ML_models/KNN_lin_6neigh'.
Výsledek kroku 3: Soubor 'ML_models/KNN_lin_6neigh' je vytvořený a jeho časové razítko odpovídá času vytvoření modelu.

6.1.3 Testovací scénář 3

Název: Získání predikce polohy
Popis: Testovací scénář 3 ověřuje, zda je možné získat predikci polohy na základě dat nahrávaných z mikrokontroleru.
Předpoklady: Spojení mezi mikrokontrolerem a serverem je navázáno a je možné odesílat data.
Na serveru je spuštěn serverový backend.
Na serveru je dostupný soubor 'ML_models/KNN_lin_6neigh'. Tento soubor obsahuje predikční model pro dříve stanovený testovací prostor.

Krok 1: V hlavní nabídce se pomocí tlačítek posuvu dostaňte na položku 'Navigation' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 1: Zobrazí se nabídka pro navigaci. Kurzor je na položce 'Go back'.

Krok 2: Pomocí tlačítek posuvu se dostaňte na položku 'Start Nav' a stiskněte tlačítko pro potvrzení výběru.
Výsledek kroku 2: Po krátké době se začne zobrazovat 'Frequency: XX', kde XX je hodnota nalezené frekvence polohy, kde se uživatel nachází. Zařízení vydává zvukový signál ve frekvenci XX.

Krok 3: Přesuňte se do bodu jiného bodu testovacího prostoru a opět zkontrolujte hodnotu frekvence polohy.
Výsledek kroku 3: Hodnota frekvence polohy se změní na hodnotu odpovídající nové poloze uživatele. Zařízení vydává zvukový signál ve frekvenci nové hodnoty frekvence polohy.

6.2 Výsledky testování

Výsledky testování ukázaly, že aplikace splňuje veškeré požadavky, které byly stanoveny v analýze. Všechny testovací scénáře byly úspěšně provedeny a všechny kroky byly splněny. Aplikace byla schopna nahrát data z mikrokontroleru na server, vytvořit predikční model a získat predikci polohy na základě dat nahrávaných z mikrokontroleru. Uživatelé byli schopni se pohybovat v testovacím prostoru a získávat predikce polohy na základě dat z mikrokontroleru a zároveň byli schopni vyčíst informace o aktuální frekvenci polohy. Dále bylo ověřeno, že aplikace je schopna zobrazit informace o frekvenci vzorkování a zároveň je schopna tuto frekvenci přehrát uživateli ve formě zvukového signálu.

Mezi některé nedostatky, které byly uživateli nahlášeny, patří následující:

- Nemožnost přehrát zvukový signál ve fázi nahrávání dat z mikrokontroleru na server.
- Hlasitost zvukového výstupu byla některými uživateli považována za příliš nízkou.
- V sekci Create KNN bylo některými uživateli považováno za matoucí zobrazení zprávy 'Click left btn!', namísto možnosti 'Go Back', tak jako tomu je v ostatních sekcích.

Žádný z těchto nedostatků nebyl považován jako selhání aplikace, ale spíše jako nedostatek, který nebyl v rámci analýzy požadavků dostatečně specifikován.

6.3 Shrnutí

V rámci testování bylo ověřeno, že aplikace splňuje veškeré požadavky, které byly stanoveny v analýze. Všechny testovací scénáře byly úspěšně provedeny a všechny kroky byly splněny. Adresované nedostatky jsou ke zvážení pro další vývoj aplikace, který by mohl být zaměřen na zlepšení uživatelského zážitku a zvýšení uživatelské přívětivosti aplikace. Pro vypracování této práce však bylo dosaženo všech cílů a bylo dosaženo všech požadavků, které byly stanoveny v analýze, není tedy nutné provádět žádné další úpravy aplikace.

Kapitola 7

Závěr

Hlavním cílem této práce bylo provést důkladnou rešerši problému a s použitím dostupné testovací platformy vytvořit systém pro vnitřní navigaci pomocí viditelného světla podle požadavků zadavatele. Cíl práce byl splněn v plném rozsahu a byl vytvořen prototyp systému, který byl řádně otestován.

Během realizace této práce došlo k následujícím krokům:

- Byla provedena rešerše problematiky vnitřní navigace.
- Byla provedena rešerše dostupných technologií pro vnitřní navigaci.
- Byla provedena rešerše dostupných technologií pro vytvoření systému pro vnitřní navigaci pomocí viditelného světla.
- Byl navržen systém pro vnitřní navigaci pomocí viditelného světla.
- Byla provedena implementace systému pro vnitřní navigaci pomocí viditelného světla.
- Bylo provedeno testování systému pro vnitřní navigaci pomocí viditelného světla.

Celkový průběh práce probíhal bez větších problémů a bylo dosaženo všech cílů, které byly stanoveny v zadání práce. Během testování bylo zjištěno několik nedostatků, které by bylo vhodné odstranit v dalším vývoji systému, avšak pro splnění cílů této práce byly tyto nedostatky zanedbatelné.

Literatura

[1] Y. Zhuang, L. Hua, L. Qi, J. Yang, P. Cao, Y. Cao, Y. Wu, J. Thompson, and H. Haas, "A survey of positioning systems using visible led lights," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1963–1988, 2018.

[2] S. Martin, "Infrastructure of indoor vlp system," Diplomová práce, České vysoké učení technické v Praze, Praha, 2022.

[3] Šimon Ondřej, "Algoritmy určování polohy pomocí komunikace ve viditelném světle," Bakalářská práce, České vysoké učení technické v Praze, Praha, 2023.

[4] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle, *GNSS — Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*, 1st ed. Vienna: Springer Verlag, Wien, 2007.

[5] D. I. Vilaseca and J. I. Giribet, "Indoor navigation using wifi signals," in *2013 Fourth Argentine Symposium and Conference on Embedded Systems (SASE/CASE)*, 2013, pp. 1–6.

[6] Y. Li, P. Zhang, H. Lan, Y. Zhuang, X. Niu, and N. El-Sheimy, "A modularized real-time indoor navigation algorithm on smartphones," in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2015, pp. 1–7.

[7] Y. See, N. M. Noor, and C. T. Y.M, "Investigation of indoor positioning system using visible light communication," in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 186–189.

[8] M. Brøgger. (2023) Five benefits of indoor mapping for your facility. [Online]. Available: https://www.spiceworks.com/tech/iot/guest-article/benefits-of-indoor-mapping-for-facility/

[9] F. Brašna. (2022) Akustické majáčky a jejich umisťování. [Online]. Available: https://www.sons.cz/Akusticke-majacky-a-jejich-umistovani-P4013016.html

[10] J. P. Bentham, "Pi pico adc input using dma and micropython," 2021. [Online]. Available: https://iosoft.blog/2021/10/26/pico-adc-dma/

[11] Y. Chen and H. Kobayashi, "Signal strength based indoor geolocation," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, vol. 1, 2002, pp. 436–439 vol.1.

[12] "Wi-fi location: ranging with rtt," 2024. [Online]. Available: https://developer.android.com/guide/topics/connectivity/wifi-rtt

[13] Inpixon, "Bluetooth rtls: Ble location tracking and positioning | inpixon." [Online]. Available: https://www.inpixon.com/technology/standards/bluetooth-low-energy

[14] A. Diallo, Z. Lu, and X. Zhao, "Wireless indoor localization using passive rfid tags," *Procedia Computer Science*, vol. 155, pp. 210–217, 2019, the 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919309457

[15] B. Štěpán, "Mobile-robot and platform for vlc indoor navigation," Diplomová práce, České vysoké učení technické v Praze, Praha, 2022.

[16] Adromnic, "Manfrotto 014ms rapidadapter," May 2020. [Online]. Available: https://www.thomann.de/cz/manfrotto_014ms_rapidadapter.htm

[17] A. Aurum and C. Wohlin, *Engineering and Managing Software Requirements*. Springer Berlin, Heidelberg, 2005.

[18] "Espressif systems," 2024. [Online]. Available: https://www.espressif.com

[19] "Raspberry pi pico and pico w - raspberry pi documentation," 2024. [Online]. Available: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

[20] "Qtcpsocket class," 2024. [Online]. Available: https://doc.qt.io/qt-6/qtcpsocket.html

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett,

A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[23] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[24] G. Van Rossum, *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[25] C. Staff, "How to use adc on raspberry pi pico in detail with micropython example," Dec. 2021. [Online]. Available: https://www.circuitschools.com/how-to-use-adc-on-raspberry-pi-pico-in-detail-with-micropython-example/

[26] R. Berka, F. Rund, L. Husník, A. J. Sporka, and České vysoké učení technické v Praze. Elektrotechnická fakulta, *Multimédia I*, 1st ed. V Praze: České vysoké učení technické, 2016.

[27] "Easyeda 2 kicad online | wokwi," 2024. [Online]. Available: https://wokwi.com/tools/easyeda2kicad

[28] Lpkf, "Lpkf protomat e44." [Online]. Available: https://www.lpkf.com/en/industries-technologies/research-in-house-pcb-prototyping/products/lpkf-protomat-e44

[29] O. Al-Debagy and P. Martinek, "A comparative review of microservices and monolithic architectures," in *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2018, pp. 000 149–000 154.

[30] M. Webster, "Man-day." May 2024. [Online]. Available: https://www.merriam-webster.com/dictionary/man-day

[31] B. Y. O. Laptop, "What is low fidelity wireframes vs high fidelity wireframes in figma," Apr. 2022. [Online]. Available: https://www.youtube.com/watch?v=UU_eyUGWIEI

[32] D. Tutoriales, "Tutorial pyqt5 - qtcpserver and qtcpsocket," May 2020. [Online]. Available: https://youtu.be/Q42mDMWtb7E

[33] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Online]. Available: https://doi.org/10.1021/ac60214a047

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and

Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

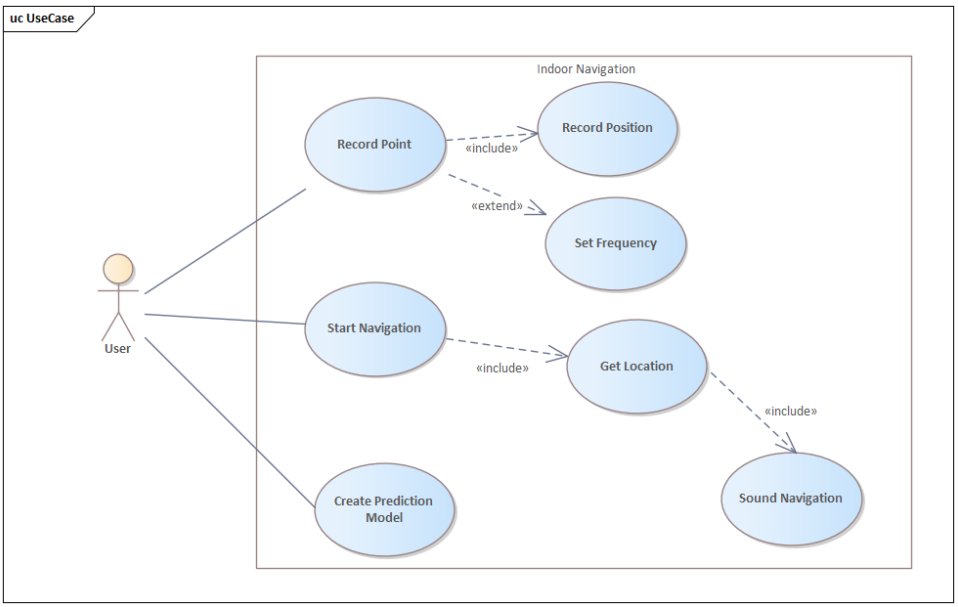
[35] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879

[36] C. Staff, "Interfacing 16x2 lcd module with raspberry pi pico with and without i2c," Jan. 2023. [Online]. Available: https://www.circuitschools.com/interfacing-16x2-lcd-module-with-raspberry-pi-pico-with-and-without-i2c/#Method2_Interfacing_16X2_LCD_display_module_with_Raspberry_Pi_Pico_with_I2C_adapter

Příloha A

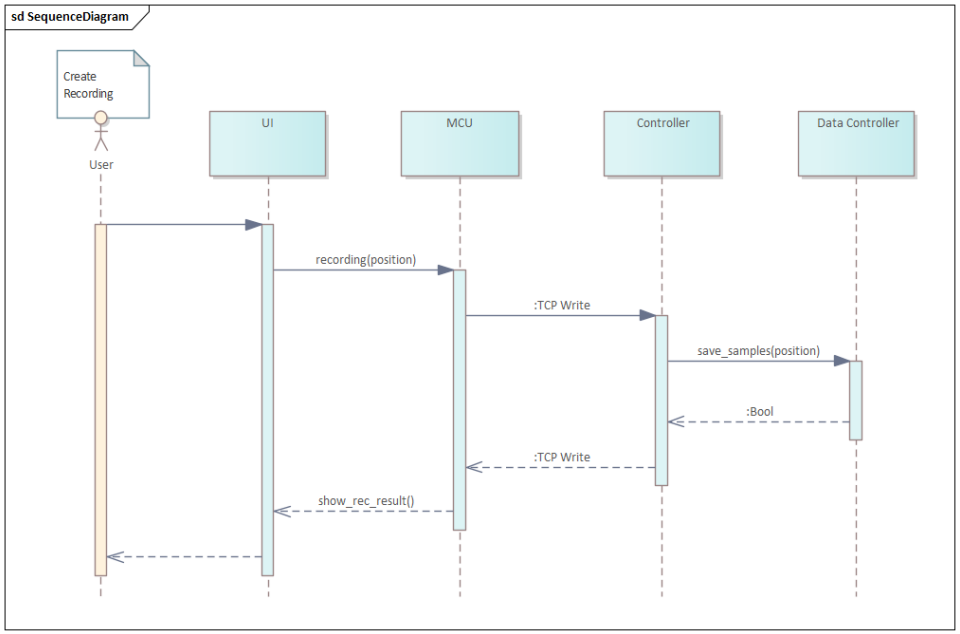
Diagramy

A.1 Diagram případů užití



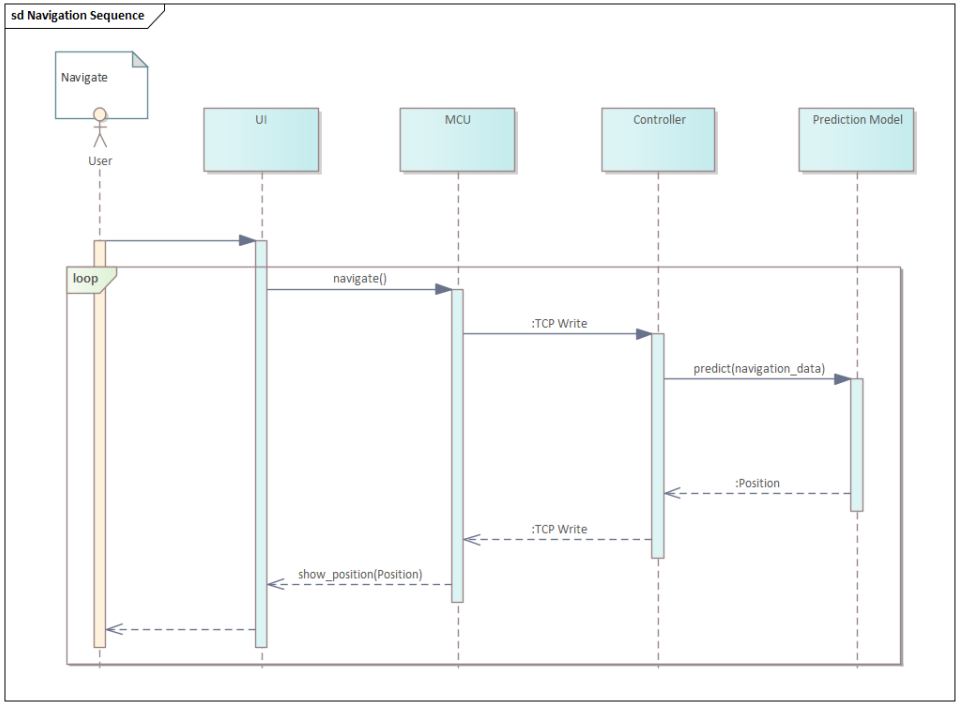
Obrázek A.1: Diagram případů užití

A.2 Sekvenční diagram: Nahrání pozičních dat



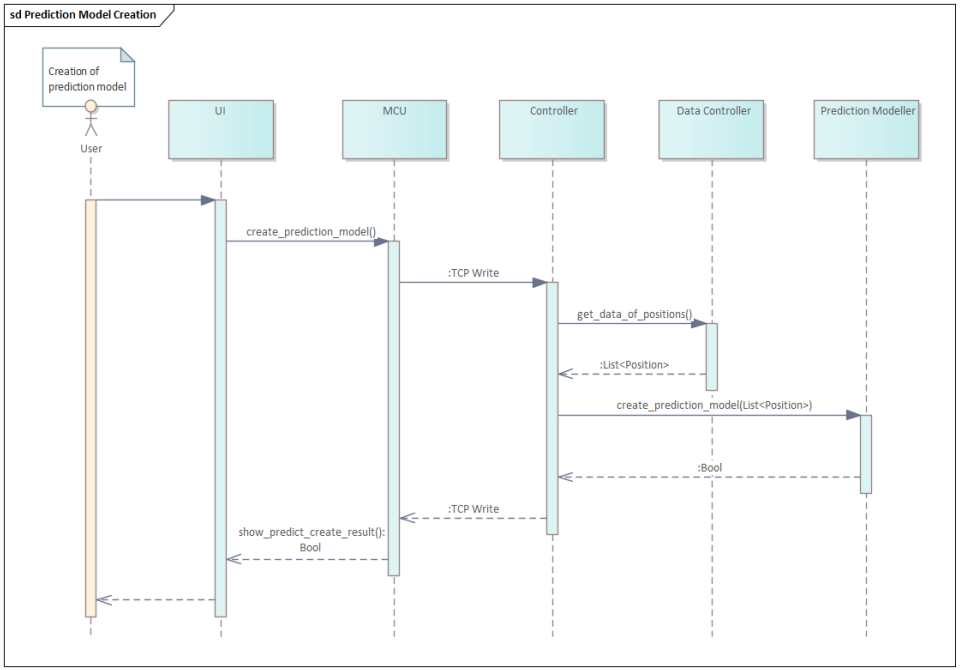
Obrázek A.2: Sekvenční diagram: Nahrání pozičních dat

A.3 Sekvenční diagram: Navigace



Obrázek A.3: Sekvenční diagram: Navigace

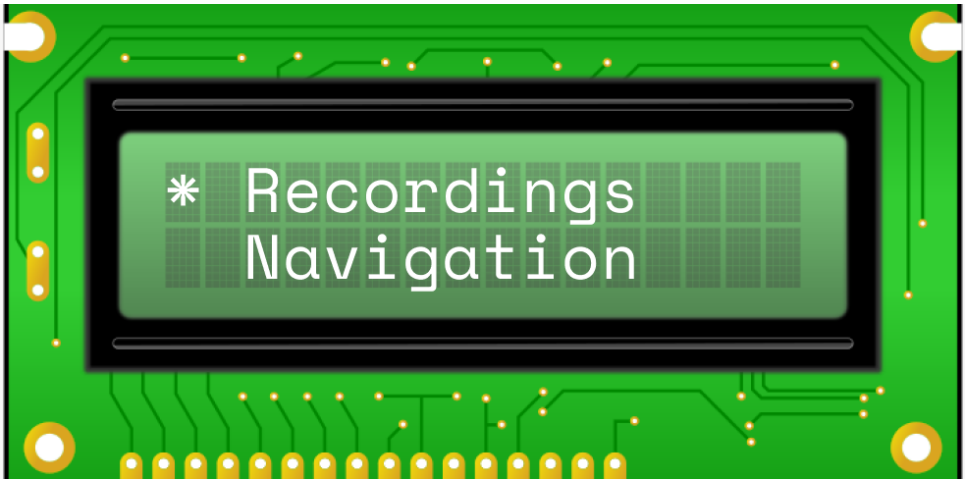
A.4 Sekvenční diagram: Vytvoření predikčního modelu



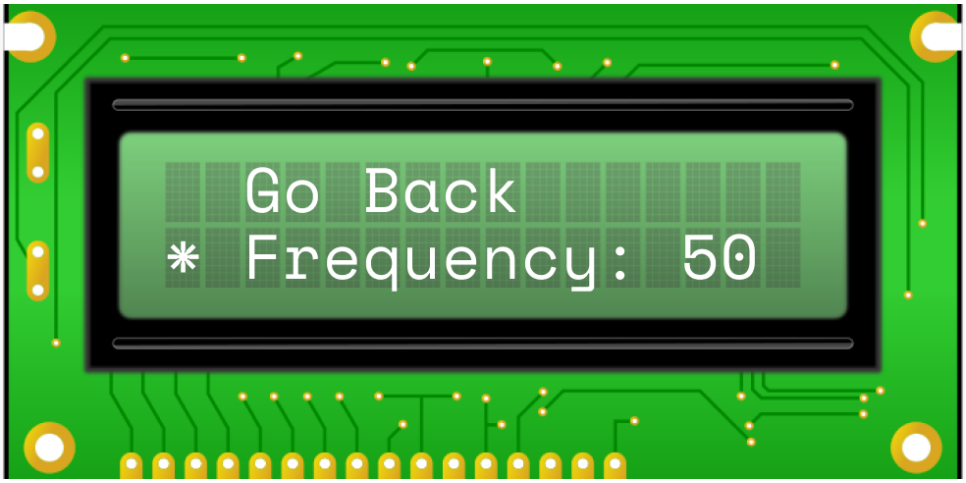
Obrázek A.4: Sekvenční diagram: Vytvoření predikčního modelu

Příloha B

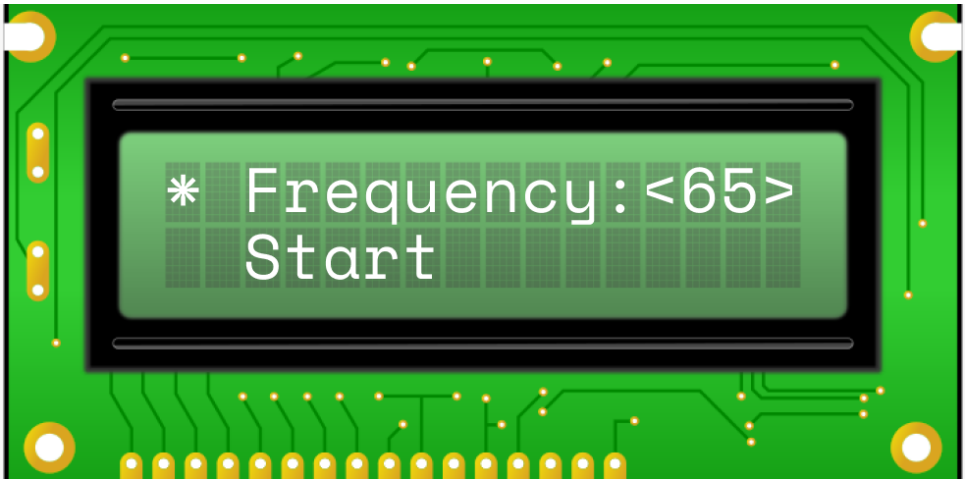
Ukázky uživatelského rozhraní



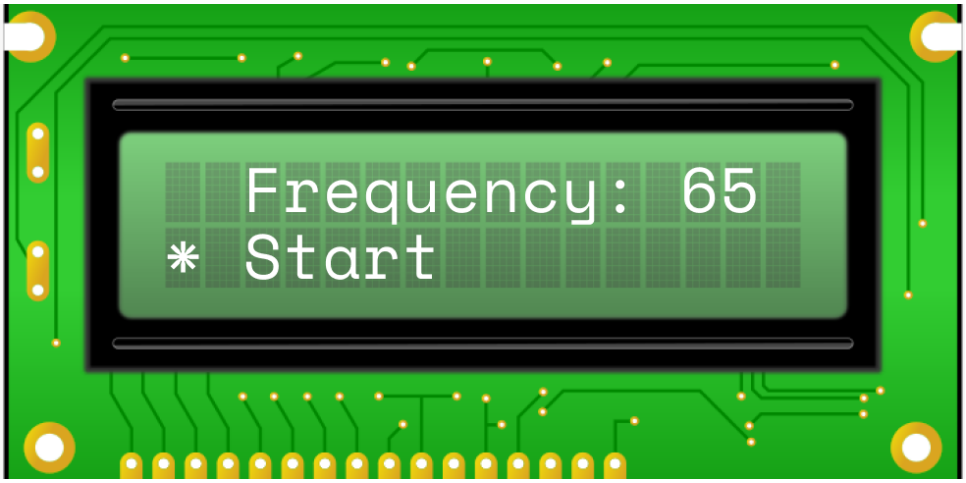
Obrázek B.1: Menu - Recordings



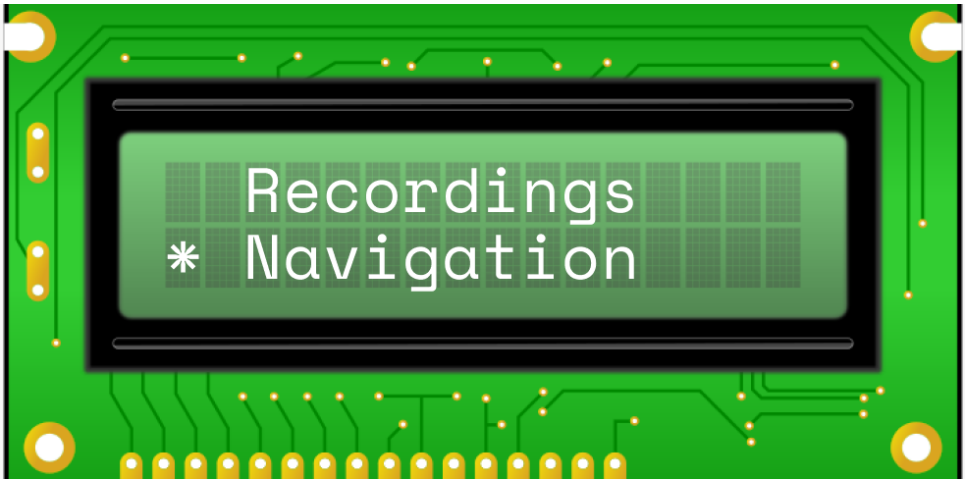
Obrázek B.2: Nahrávání - Frekvence



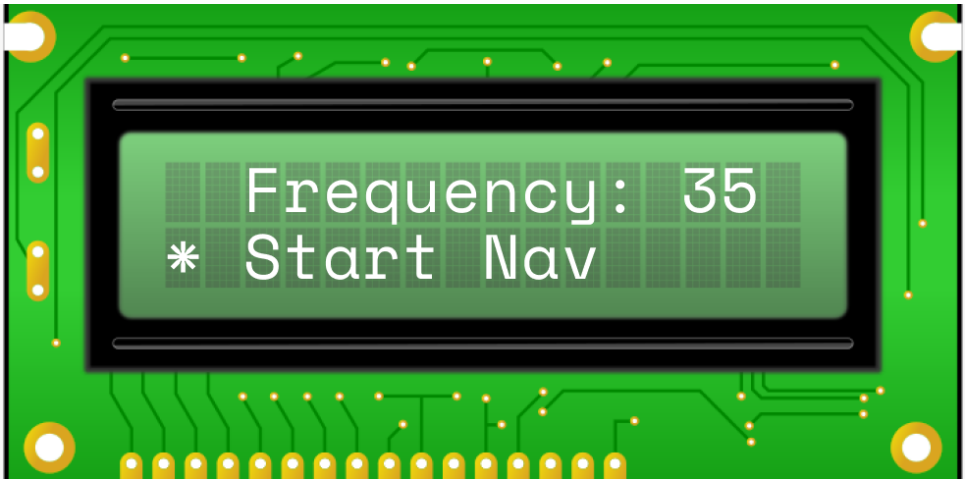
Obrázek B.3: Nahrávání - Zvolení frekvence



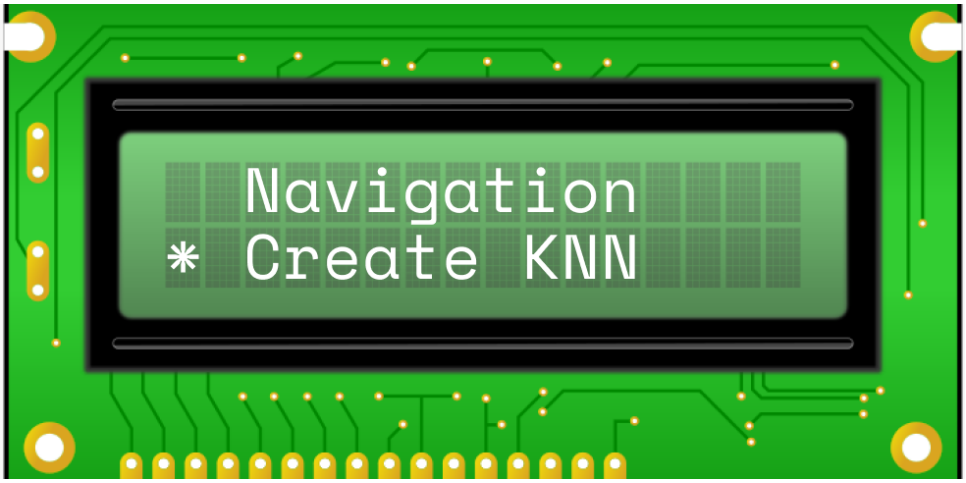
Obrázek B.4: Nahrávání - Spuštění nahrávání



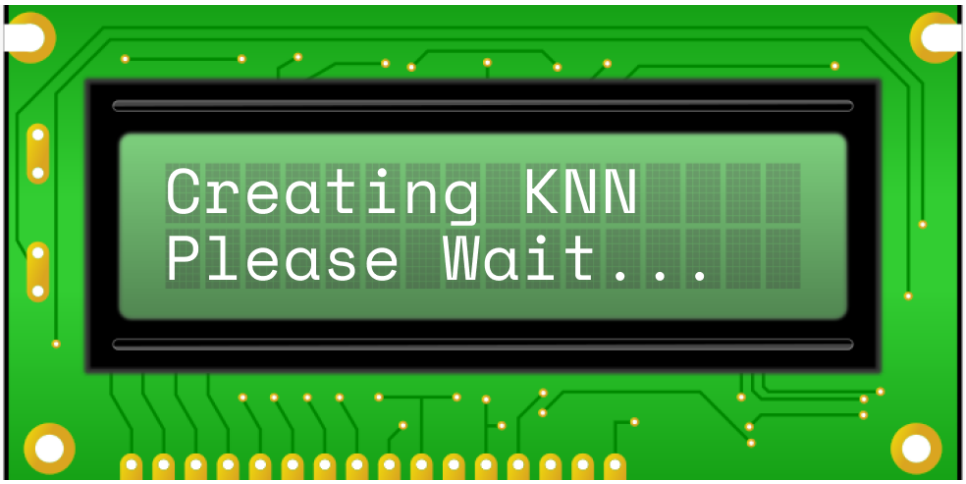
Obrázek B.5: Menu - Navigation



Obrázek B.6: Navigace - Start Nav



Obrázek B.7: Menu - Create KNN Model



Obrázek B.8: Create KNN - Wait for model creation

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vomastek** Jméno: **Jan** Osobní číslo: **510851**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Technologie internetu věcí**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vnitřní navigace pomocí VLC

Název bakalářské práce anglicky:

Indoor navigation using VLC

Pokyny pro vypracování:

Seznamte se s metodami navigace uvnitř budov pomocí VLC a s testovací platformou dostupnou na pracovišti zadavatele. Navrhněte a implementaci zařízení pro určení polohy v rámci testovací platformy. Návrh proveďte s ohledem na nízkou spotřebu a přenositelnost zařízení. Implementované zařízení otestujte a diskutujte případné nedostatky.

Seznam doporučené literatury:

[1] Y. Zhuang, L. Hua, L. Qi, J. Yang, P. Cao, Y. Cao, Y. Wu, J. Thompson, and H. Haas, “A survey of positioning systems using visible led lights,”
IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 1963–1988, 2018.
[2] S. Martin, “Infrastructure of indoor vlp system,” Diplomová práce, České vysoké učení technické v Praze, Praha, 2022.
[3] B. Štěpán, “Mobile-robot and platform for vlc indoor navigation,” Diplomová práce, České vysoké učení technické v Praze, Praha, 2022.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Stanislav Vítek, Ph.D. katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **15.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

doc. Ing. Stanislav Vítek, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta