Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Inteligentní navigace v objektech

Bakalářská práce

Ondřej Šach

Školitel: Ing. Václav Novák, CSc.

České Budějovice 2020

Bibliografické údaje:

ŠACH, O., 2020: Inteligentní navigace v objektech. [Intelligent navigation in objects, Bc. Thesis, in Czech] – 39 p, Department of Physics, Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt:

Bakalářská práce se zabývá zařízením, pomocí kterého bude možné získat data pro navigaci nevidomých lidí a dále je zprostředkovat nadřazeným systémům. Hlavní součástí celého systému je mapovací modul „SLAMTEC Mapper M1M1“. Naměřená data z modulu jsou zpracována vlastní aplikací naprogramovanou v jazyce C++. Pomocí formátu CSV jsou následně zpracovaná data předávána nadřazeným systémům. Práce obsahuje návrh zařízení, popis jeho realizace včetně programování aplikace a v neposlední řadě také shrnutí závěrečného testování.

Abstract:

This bachelor thesis deals with a device able to obtain data for the navigation of visually impaired people and pass this data to higher-level systems. The main part of the whole system is the mapping module „SLAMTEC Mapper M1M1“. The measured data from the module are processed by custom application programmed in C++. The processed data are passed to higher-level systems using the CSV format. The thesis contains the design of the device, a description of implementation of the device and the programming of the custom application. The thesis contains a summary of final testing too.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích,

dne ..... Podpis studenta .....

Poděkování:

Touto cestou bych velmi rád poděkoval vedoucímu práce Ing. Václavu Novákovi CSc. za cenné rady, čas a trpělivost, kterou mi věnoval. Dále bych chtěl poděkovat rodině a blízkým za podporu a trpělivost během zpracovávání této práce.

Obsah

1 Úvod

Zrak – jeden z pěti základních smyslů člověka, je považován za ten nejdůležitější. Zrakem vnímá člověk drtivou většinu informací z okolního světa [1]. Někteří lidé se bohužel musí ve svém životě obejít bez něj. Ať už je důvodem vrozená vada, nebo nešťastná náhoda v průběhu života jedince, rozhodně je to značná komplikace pro život. Světová zdravotnická organizace WHO dělí zrakové postižení do pěti skupin [1, příloha č.1]. Lidé, kteří zcela přišli o zrakový smysl, nebo jsou pouze částečně omezeni, mají mnohem lépe vyvinuty smysly ostatní. Pokud se budeme bavit o pohybu nevidomého jedince, největší podíl na něm má paměť. Dostane-li se takto postižený člověk do neznámého prostředí, přicházejí na řadu jiné pomůcky. Výsledkem této práce by měla být jedna z nich, systém, který jim orientaci v objektech usnadní.

1.1 Stanovení cílů práce

Cílem této práce je navržení systému orientace v objektech pomocí modulu „SLAMTEC Mapper M1M1“. Pomocí výše zmíněného modulu získáme data, která následně zpracuje vlastní aplikace naprogramovaná v jazyce C++. Výstupem z aplikace bude seznam překážek v okolí nevidomého, který bude vhodný k dalšímu zpracování např. řečovými procesory nebo robotickým zařízením. Toto zpracování již nebude součástí této práce. Nedílnou součástí práce bude následné otestování výstupních dat a stanovení mezí jejich použitelnosti.

1.2 Navigace nevidomých

Nejjednodušší možností navigace postiženého člověka je doprovod. Pokud není tato možnost dostupná, je nutné k jeho navigaci použít kompenzační pomůcky.

„*Kompenzační pomůckou pro těžce zrakově postižené se rozumí nástroj, přístroj nebo zařízení, speciálně vyrobené nebo speciálně upravené tak, aby svými vlastnostmi a možnostmi použití alespoň částečně kompenzovalo nedostatečnost způsobenou těžkým zrakovým postižením.“* [1, s. 9]

Dle zdrojů [1] a [2] patří mezi běžné pomůcky pro navigaci postiženého slepecká hůl, vodící linie, vodící pes, reliéfní značky a akustické majáčky. Díky prakticky nezastavitelnému vývoji v informačních a komunikačních technologiích můžeme pozorovat ohromný pokrok také v oboru novodobých kompenzačních pomůcek. Tyto pomůcky si blíže představíme v teoretické části práce.

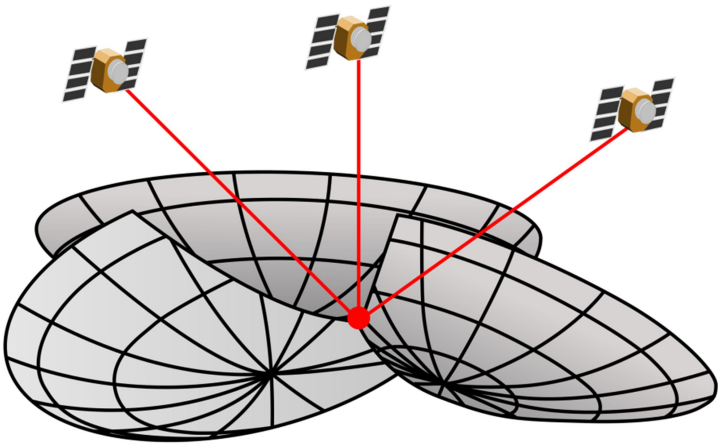
2 Teoretická část

V této části práce si představíme novodobé kompenzační pomůcky pro navigaci nevidomých. Je nutno říci, že ač jsou to moderní a často velmi dobře propracované produkty, stále disponují určitými nedostatky. Jedním z hlavních nedostatků moderních navigačních systémů je rozsah použití. Hned na samém začátku bude nutné rozdělit systémy do dvou skupin. V první skupině stručně rozebereme princip fungování navigace v exteriéru a představíme si existující navigační systémy. Na druhou skupinu navigačních systémů a to sice navigací v interiéru se podíváme podrobněji, protože jeden z těchto systémů je hlavním tématem této práce.

2.1 Navigace vně objektů

Při zaměření na navigaci v exteriéru je nutné zmínit hlavní součást - tou je nepochybně GPS (Global Positioning System), na kterém jsou současné venkovní navigační systémy založeny. Začátky GPS se mapují v roce 1973 za účelem navigačního systému pro americkou armádu. O 10 let později byl uvolněn přístup k technologii pro civilní účely. Roku 1994 byl systém rozšířen po celém světě a byl plně funkční. Nyní existují další alternativy systému, GLONASS (Rusko) a Galileo (EU). Dále se pak o vývoj snaží také Čína, Japonsko a Indie, kde jsou zatím systémy pouze v rámci regionu [3].

Funkčnost je založena na měření vzdálenosti mezi přijímačem GPS a alespoň třemi družicemi. Tím se dostáváme k hlavnímu problému tohoto druhu navigace. Čím lepší výhled na oblohu máme, tím je větší šance, že bude naše pozice zjistitelná a hlavně určení pozice proběhne s vyšší přesností. Proto je tento typ v interiéru prakticky nepoužitelný [3].



Obrázek 1: Princip funkčnosti GPS

(Zdroj: https://www.pinclipart.com/maxpin/bxxibw/)

Nyní se již pojďme věnovat samotným navigačním systémům této kategorie.

Klasická navigace GPS. První z možností je klasická navigace, dnes již dostupná například v mobilním telefonu. Hlavní předností jsou samozřejmě snadno dostupné mapové podklady, na kterých nám aplikace ukazuje cestu. Díky hlasovým pokynům může ale fungovat dobře i pro zrakově postižené jedince.

PST (Pospíšil Smart Talker). Jedná se o mluvící aplikaci pro mobilní telefony, jenž disponuje i funkcí navigace. Využívá databázi 35 000 zaměřených bodů po celé České Republice. Po zaměření osoby určí aplikace azimut k nejbližšímu bodu z databáze. Je možné si vytvořit vlastní databázi bodů [4].

Zahraniční aplikace Loadstone GPS. Je velmi podobná již zmíněné aplikaci PST. Mezi hlavní výhody patří možnost využití sdílení zájmových bodů s ostatními uživateli.

Alternativou je NaviTerier. Aplikace se snaží navigovat nevidomého pomocí předem připravených popisů exteriérů a budov. Z důvodu přesnosti a srozumitelnosti spolupracují na těchto popisech i lidé z navigačního centra SONS (Sjednocená organizace nevidomých a slabozrakých). Pokud by došlo ke krizové situaci, vše vyřeší operátor na lince SONS. Bohužel zatím není aplikace příliš rozšířená a funguje pouze v rámci hlavního města Prahy.

Navedení přímo pomocí centra navigace SONS. Pracovník asistenční linky lokalizuje postiženého pomocí přijímače GPS a navede jej do cílové destinace. Nejde pouze o navigaci, organizace nabízí také další služby. Můžeme si nechat např. vypracovat s předstihem itinerář plánované trasy, vyhledat si dopravní spojení a nově je také nabízena vzdálená podpora přes kameru chytrého zařízení prostřednictvím aplikace Skype. Služby jsou ale placené [5].

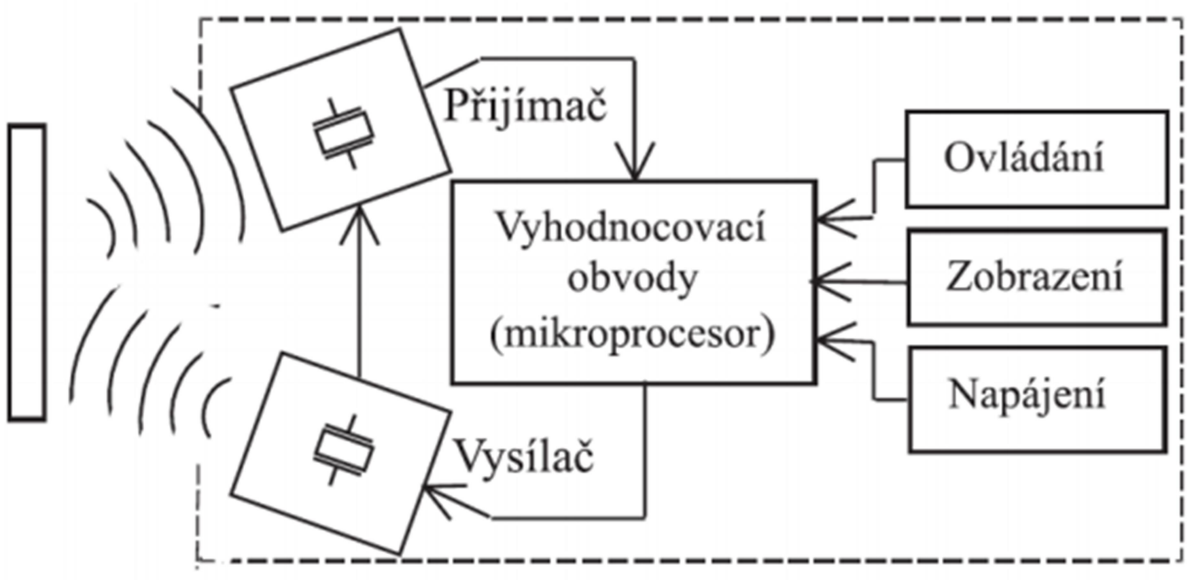
Slepecká hůl propojená s chytrým telefonem. Zatím nejnovější systém byl představen vědci ČVUT v listopadu 2015. Takto upravená hůl je schopná pomocí GPS přenést polohu nevidomého a zároveň pomocí mobilních dat a kamery také aktuální obraz, kde se nachází. Na základě těchto informací může být nevidomý navigován pracovníkem navigačního centra, ale také například rodinným příslušníkem [12].

2.2 Navigace uvnitř objektů

Jak je zřejmé již z nadpisu, druhá skupina navigačních systému se věnuje pohybu člověka uvnitř objektu. Člověk, který má zrakový smysl v pořádku a vstoupí do místnosti, si musí uvědomit, co se nachází kolem něj. Následně zvolí vhodnou trasu, kterou se bude pohybovat a za předpokladu, že je zrak a ostatní smysly v pořádku, by neměl nastat žádný problém při jeho pohybu po místnosti. Nyní si ale představme, co se odehrává, pokud do neznámé místnosti vstoupí nevidomý. Nevidí vůbec nic, nemá tušení, jak místnost vypadá a kde se nacházejí překážky. V tomto okamžiku musejí přijít na scénu navigace v objektech. Ty se dají rozdělit do několika skupin podle principu funkčnosti resp. podle použitých senzorů.

2.2.1 Navigace pomocí ultrazvuku

Hlavním faktorem tohoto druhu navigace je ultrazvukový detektor překážek. Detekce překážek je založena na vyhodnocovacích obvodech. Ty mají za úkol změřit dobu trvající od vyslání ultrazvukového pulzu, který se odráží od překážky, až po následnou detekci odraženého pulzu přijímačem. Pokud je odražený ultrazvukový pulz vyhodnocen jako možný odraz vyslaného signálu, je z délky intervalu a rychlosti šíření zvuku v konkrétním prostředí vypočtena vzdálenost od překážky. U tohoto typu snímače dochází ke zkreslení v případě vlivu prostředí na rychlost šíření zvuku (vlhkost, teplota). Zároveň mohou mít tyto detektory horší vlastnosti například v nemocnici, kde se nacházejí další přístroje využívající ultrazvuk. Pro lepší představu slouží blokové schéma ultrazvukového snímače polohy [6][8].



Obrázek 2: Blokové schéma ultrazvukového snímače polohy

(Zdroj: převzato z [6])

Ultrazvukový snímač nám překážku detekoval, nezbývá tedy, než zprostředkovat její polohu nevidomému. Interakce s nevidomým nejčastěji probíhá pomocí vibrací nebo akustického signálu. Čím je překážka blíže, tím jsou vibrace silnější resp. akustický signál je hlasitější, nebo má větší frekvenci opakování. Na stejném principu jsou založeny parkovací senzory v automobilech. Základní princip fungování známe, pojďme si tedy představit běžně dostupné ultrazvukové detektory překážek.

Vyhledávač překážek RAY. Tento detektor je doporučeno používat v kombinaci se slepeckou holí. Pracuje v rozmezí 15-250cm. Při nekonečné vzdálenosti překážky jsou vibrace cyklovány cca po jedné vteřině. Maximální možnou frekvenci mají při vzdálenosti 15cm od překážky. Tento přístroj dokonce zvládá detekovat i světlo, vše záleží na jeho nastavení, které lze provést pomocí manuálu v českém jazyce. Dle zdrojů je schopný odhalit i drobné větve při procházce zahradou [7].

Obrázek 3: RAY

(Zdroj: převzato z [7])

Ultrazvukové brýle. Na první pohled běžné brýle skrývají těsně pod horní obroučkou dva ultrazvukové senzory. Po stranách jsou ovládací tlačítka na zapnutí a regulaci vibrací. Brýle identifikují všechny překážky, které se nacházejí blíže než 3m. Pokud se nachází překážka blíže než 70cm, vibrace jsou souvislé. Jelikož bylo z důvodu velikosti třeba využít jiného senzoru než u detektoru RAY, mají brýle menší citlivost. Naopak výhodou jsou volné ruce při používání kompenzační pomůcky [7].

Obrázek 4: Ultrazvukové brýle

(Zdroj: http://pomucky.blindfriendly.cz/pomucky-pro-usnadneni-mobility.html)

Chytrý náramek Sunu Band. Náramek vypadající jako dnes již běžně využívané chytré hodinky se hodinkám blíží také funkcemi. Obsahuje navíc echolokaci pro orientaci v prostoru. K detekci překážek je využit pouze jeden ultrazvukový senzor, takže pro monitorování prostoru kolem sebe je nutné otáčet rukou. Výrobcem je přímo doporučeno nepoužívat pomůcku samostatně, naopak nejlepší výsledky se dostaví v kombinaci se slepeckou holí. Pokud náramek spárujeme s chytrým telefonem, můžeme měnit mimo jiné i krátký nebo dlouhý dosah sonaru. Krátký režim je primárně určen pro orientaci uvnitř a detekuje překážky od vzdálenosti 1,5m, rozpoznává pouze větší objekty. Pokud nastavíme dlouhý dosah, sonar je schopen detekovat překážky až do vzdálenosti 5m, zároveň je nastaven na citlivější režim, aby upozornil nevidomého i na keře, dopravní značky a podobné menší překážky. Chytrý náramek lze používat i bez spárování s telefonem, ale uživatel zbytečně přichází o mnoho funkcí a zároveň je interakce s náramkem omezena pouze na vibrace, nikoli na zvukový výstup. Mezi výhody patří opět volné ruce, jako při použití brýlí, tento komfort nám bohužel detektor RAY neumožňuje. Náramek není vhodný k detekci pohybujících se předmětů, proto by se měl nevidomý v městském provozu primárně spoléhat na svůj sluch jako doposud [8].

Obrázek 5: Náramek Sunu Band

(Zdroj: https://www.czechcrunch.cz/2019/10/karel-giebisch-nevidomi-jako-netopyri-aneb-jak-chytry-naramek-s-ultrazvukem-zrakove-postizenym-dodava-sesty-smysl/)

2.2.2 Navigace pomocí RFID čipů

Jak už název napovídá, hlavním faktorem navigace jsou RFID (Radio Frequency Identification) čipy, další nedílnou součástí je také speciálně upravená slepecká hůl.

RFID čip je malá součástka, která využívá energii vysílanou od čtečky k nabití svého napájecího kondenzátoru a následně odešle odpověď. Tyto čipy jsou rozmístěny po interiéru, většinou je zde možnost nalepení, nebo se již při výstavbě objektu zabudují pod omítku.

Pro čtení informací z čipu poslouží v konkrétním případě již zmíněná speciální slepecká hůl vybavená příslušnou elektronikou. Tato elektronika jednak vysílá signály pro napájení čipu, následně pak přijatou informaci z čipu odešle pomocí bezdrátové technologie do chytrého telefonu, který nevidomému předá předem naprogramovanou informaci pomocí hlasového výstupu [9][10].

Obrázek 6: Detail hole a telefonu s aplikací RF Guide

(Zdroj: převzato z [4])

2.2.3 Navigace prostřednictvím magnetického pole

Další systém, který si představíme, pracuje s velice zajímavou myšlenkou a tou je diskrétnost. Ne nadarmo je nazýván jako „Diskrétní informační systém“. Nevidomý totiž dostává potřebné informace do sluchátka, pro ostatní zcela neslyšitelnou formou.

Nyní již ale k hlavnímu principu systému. Ten je velice podobný principu předchozího systému. Magnety, které jsou umístěny např. do země v místě, kde je třeba předávat informace, vyzařují magnetické pole podle toho, jak jsou natočeny. Další součástí je opět speciálně upravená slepecká hůl s vyhodnocovací elektronikou. Pokud nad magnety projdeme s holí, elektronika vyhodnotí příslušné magnetické pole a pomocí vibrací pošle zpětnou vazbu nevidomému. Ten díky vibracím zjistí, že je poblíž informační systém. V ten moment přichází na řadu další dvojice komponentů. Sluchátko a elektronika obsahující informace o místě, kde se nevidomý nachází. Součástí této elektroniky je kontaktní pole, které po interakci s postiženým přenese informaci přes jeho kůži do sluchátka [11].

Obrázek 7: Bezdrátové sluchátko, klika se zabudovaným zdrojem informací a detail hole se zabudovaným snímačem magnetů

(Zdroj: převzato z [11])

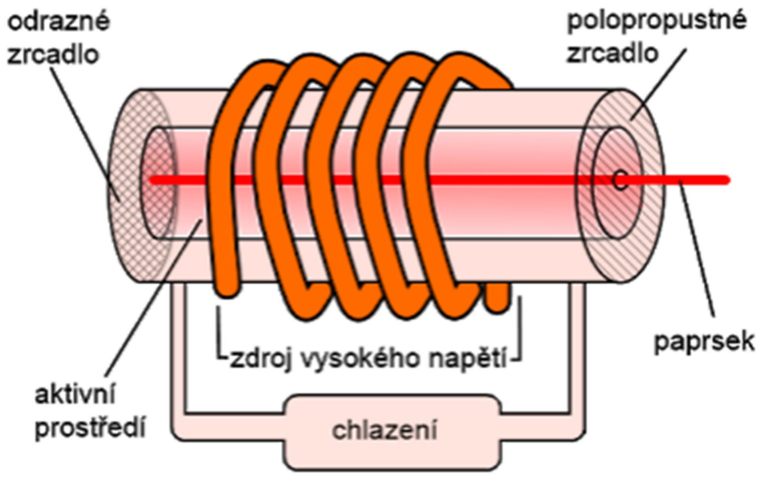
Tento systém již funguje například v kroměřížské Květné zahradě viz. https://ct24.ceskatelevize.cz/regiony/jihomoravsky-kraj/1333959-krasu-komerizske-kvetne-zahrady-oceni-i-nevidomi

2.2.4 Navigace pomocí LIDARu

Jelikož je tato metoda navigace hlavním tématem práce, rád bych ji probral podrobněji. LIDAR (Light Detection and Ranging) je bezkontaktní měření vzdálenosti pomocí LASERového paprsku. Bylo by tedy vhodné si LASER podrobněji popsat.

2.2.4.1 LASER

Zkratka LASER (Light Amplification by Stimulated Emission of Radiation) v překladu znamená zesilování světla stimulovanou emisí záření. Základem pro vznik LASERového paprsku je interakce fotonu a elektronu. Existuje-li elektron, do kterého vletí foton, je tento elektron excitován na vyšší energetickou hladinu a foton zanikne. Elektron ale na vyšší hladině dlouho nevydrží a dochází k přechodu zpět na základní energetickou hladinu. Při tomto přechodu vyletí foton. Pokud chceme, aby se vytvořil paprsek, je nutné mít tzv. aktivní prostředí. Tím se rozumí prostředí, které obsahuje prvky s metastabilní hladinou. Na metastabilní hladině vydrží elektron čekat až 100 000x déle na další foton, který ho stimuluje. Jelikož nám zdroj dodává neustále energii, dostaneme většinu elektronů na metastabilní hladinu. Tyto elektrony pak čekají na stimulující foton, který je donutí sestoupit na základní hladinu a vypustit foton. Vypuštěné fotony se spojí se stimulujícím fotonem v jednu vlnu, která má mnohem větší amplitudu. Směr této vlny je stejný, kterým letěl stimulující foton. Bohužel může být ale odlišný od směru, který požadujeme, proto je třeba rezonátor. Ten se skládá ze dvou zrcadel. Jedno je polopropustné a druhé odrazivé, obě jsou ale rovnoběžné vůči sobě. Fotony letící mimo osu rezonátoru se několikrát odrazí mezi zrcadly, nebo rovnou vyletí mimo. Existují ale určitě i fotony letící rovnoběžně s osou laseru. Počet fotonů odrážejících se mezi zrcadly bude exponenciálně přibývat až do doby, kdy budou schopny projít polopropustným zrcadlem, čímž vzniká laserový paprsek [14].



Obrázek 8: Princip funkce LASERu

(Zdroj: http://www.fyzika007.cz/fyzika-mikrosveta/laser)

Přesto, že existuje mnoho typů laserů, všechny pracují na stejném principu. Dnes nejrozšířenější je laser polovodičový, kde je zdrojem záření tzv. laserová dioda. V následující tabulce je k dispozici přehled nejpoužívanějších aktivních prostředí a k nim přiřazená vlnová délka paprsku.

Aktivní prostř.	Vlnová délka (nm)	Poznámka
argon - fluor	193	UV, excimer
krypton - fluor	248	UV, excimer
xenon - chlor	308	UV, excimer
helium - kadmium	325, 442	UV, viditelné
argon	488, 514	viditelné
rubín	694	viditelné
Nd:YAG	532, 1064	viditelné, IČ
helium - neon	543, 594, 612, 633, 1150, 3390	viditelné, IČ
polovodičové diody	630 - 1600	viditelné, IČ
erbium	1540	IČ
oxid uhličitý	9600, 10600	IČ

Tabulka 1: Přehled typů laserů

(Zdroj: https://www.cez.cz/edee/content/microsites/laser/f4.htm)

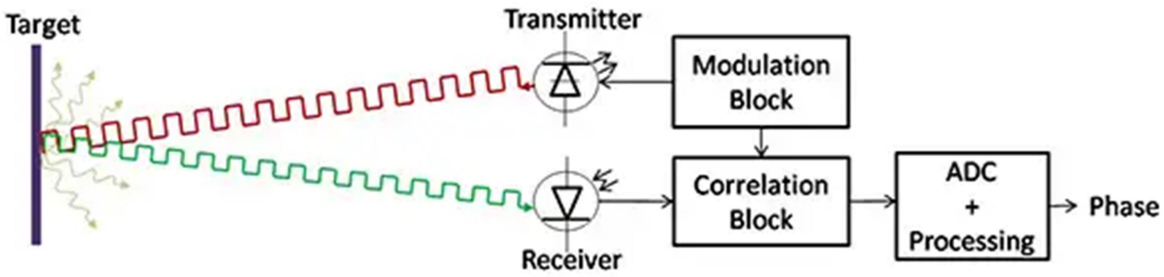
2.2.4.2 LIDAR

Princip základní součásti LIDARu již známe, nyní si představíme LIDAR samotný. První zmínky o tomto systému jsou z roku 1961. V počátcích se využíval v leteckém průmyslu na mapování terénu. Roku 1995 se eviduje první aplikace v automobilovém průmyslu, kdy byl tento systém použit pro adaptivní tempomat. V současné době je LIDAR téměř samozřejmostí vědy a je využíván v mnohých oblastech. Existuje mnoho druhů těchto senzorů, které se liší konstrukcí, metodou zpracování signálu a dalšími parametry. Všechny tyto parametry mají vliv na výslednou aplikaci senzoru.

Nyní si přiblížíme základní princip, na kterém LIDAR funguje. Základ tvoří již zmíněný laserový zdroj, který vysílá světelné pulzy. Měření vzdálenosti překážek probíhá v našem případě pomocí metody ToF (Time of flight). Tato metoda počítá čas, který uplyne od vyslání rovnoběžného světelného svazku z vysílače do doby, než je svazek odražený od překážky detekován přijímačem světelného signálu. Tento přijímač je tvořen fotocitlivou součástkou, která převede světelný signál na elektrický. Dále je na řadě zpracování elektrického signálu, to má na starosti buď A/D převodník, nebo komparátor.

Vzdálenost l od překážky získáme z rozdílu času Δt vyslání a příjmu světelného svazku a rychlosti světla c podle rovnice:

$$l = c \cdot \frac{\Delta t}{2} \tag{1}$$



Obrázek 9: Blokový diagram ToF principu

(Zdroj: https://www.digikey.com/en/articles/techzone/2017/jan/simplifying-time-of-flight-distance-measurements)

Protože potřebujeme o překážce zjistit nejen vzdálenost, ale i oblast kde se nachází. Musí mít LIDAR informaci o tom, v jaké pozici byl konkrétní paprsek vyslán. Tuto informaci získáme z úhlu konkrétního natočení modulu, který se udává v *rad*. Tento úhel zjišťujeme z dekodéru polohy DC motoru (stejnosměrný motor), který má za úkol otáčet zdrojem laserového paprsku.

U typu LIDARu, který pracuje na principu TOF, se také můžeme setkat s mnoha nepříjemnostmi, které ovlivní měření. Jednou z nich je počasí. Například déšť ovlivní senzor rovnou dvěma způsoby. Prvním je vlhkost, která určitě neprospívá použité elektronice. Druhým způsobem jsou samotné kapky, které ovlivňují odraz světelného svazku a mohou ho vychýlit špatným směrem. Dalším problémem jsou například sluneční paprsky směřující do detektoru. Proto je tento senzor určen primárně pro použití v interiéru. Zde se ale bohužel také naleznou faktory ovlivňující měření. Jednou z hlavních příčin chybného měření jsou použité okolní materiály. Je-li detekovaná překážka vyrobena z lesklého materiálu, dostáváme se do problému s odrazem paprsku. Vyslaný paprsek se od překážky odrazí jinak, než očekává detektor a dochází k chybnému měření [13].

3 Praktická část

Teorii týkající se současných možností navigace a jejich funkčnosti máme za sebou. V praktické části práce bych rád popsal jednotlivé fáze samotné realizace vlastního systému pro navigaci v objektech pomocí LIDARu.

3.1 Použité komponenty

Zde si detailně představíme hlavní komponenty celého systému, které se dají rozčlenit do dvou skupin. Výběr komponentů byl značně usnadněn zadáním této práce, kde bylo jasně specifikováno, že hlavním komponentem práce musí být SLAMTEC Mapper M1M1.

3.1.1 Hardware

Obrázek 10: Modul SLAMTEC Mapper M1M1

(Zdroj: vlastní tvorba)

SLAMTEC Mapper M1M1. Základ celého systému tvoří již zmiňovaný modul. Skládá se ze dvou desek. Popis začneme na spodní desce. Na čelní straně desky vidíme napájecí konektor pro DC (stejnosměrné) napětí. Pro připojení napájení slouží kabel, který je součástí balení. Je dlouhý 1,5m a je zakončen koncovkou USB. Napájení můžeme tedy realizovat přímo z počítače, na kterém data zpracováváme, nebo například pomocí powerbanky (sada akumulátorů s minimálně jedním USB výstupem). Vedle napájecího konektoru najdeme USB konektor, o kterém se v dokumentaci nepíše ani čárka, ovšem podobných komponent je na modulu více, proto se k tomuto problému ještě vrátíme. Další na řadě je rozhraní Ethernet pro komunikaci s počítačem. Pokud se budeme dívat dále po obvodu spodní desky, narážíme na již zmíněný problém. Můžeme zde vidět hromadu konektorů, ale využitý je pouze jeden z nich. Tento konektor slouží pro komunikaci s LIDARovým senzorem umístěným v horní

části modulu. Ohledně nevyužitých konektorů jsem vznesl dotaz skrze podporu na oficiálním webu výrobce. K mému překvapení byla obdržená odpověď velice stručná: „Tyto konektory nejsou pro Vás jako uživatele zajímavé, pro zpracování dat Vám stačí rozhraní Ethernet nebo WiFi.“ Ano WiFi modul je dalším komponentem, který můžeme odhalit na spodní straně výrobku, konkrétně pod LIDARovým senzorem. Poslední součástkou, která se nachází na spodní straně modulu, je zapínací tlačítko, vedle něj je již pouze zelená SMD LED dioda indikující zapnutý stav.

Horní deska modulu je z matného plastu a neslouží pro osazení elektronických součástek za pomoci pájení. Pomocí plastových šroubů je na ni připevněn LIDARový senzor se svým vlastním pohonem, který tvoří DC motor. Motor je propojen pomocí drátů s plastovými konektory přímo s LIDARovým senzorem za účelem komunikace a napájení.

LIDARový senzor má na své spodní straně desku s řídící elektronikou. V polovině těla senzoru můžeme vidět drobnou drážku sloužící pro vedení gumového řemenu, který senzorem otáčí. Pohon otáčení zajišťuje výše zmíněný stejnosměrný motor. Pokud se podíváme do horní části, můžeme vidět otočnou hlavu, která disponuje dvěma otvory. Jeden slouží pro vysílač paprsku a druhý je samozřejmě pro přijímač. Pokud omezíme prostupnost těchto otvorů, měření nebude k dispozici, nebo bude nepřesné. LASERový paprsek použitého modulu patří do bezpečnostní třídy 1, takže je bezpečný pro oko člověka i pro jeho domácí mazlíčky. Více informací o LASERu nad rámec oficiální dokumentace není možné sehnat, protože jsou podle informací z podpory obchodním a technologickým tajemstvím. Dostupné vlastnosti LASERu převzaté z dokumentace k modulu uvádím v tabulce č.4. Měření vzdálenosti probíhá s využitím již zmiňované metody ToF (Time of flight) s maximálním dosahem 20 m.

V přiložených tabulkách se podíváme na parametry modulu. První tabulka obsahuje rozměry modulu M1M1.

Length	104mm	Délka
Width	60mm	Šířka
Height	66mm	Výška
Weight	230g	Váha

Tabulka 2: Rozměry modulu M1M1

(Zdroj: vlastní tabulka, hodnoty převzaty z [16])

Další tabulka obsahuje parametry týkající se měření.

Working temperature range	-5°C – 45°C	Rozmezí provozních teplot
Distance range	20m	Maximální vzdálenost měřená radarem
Sample rate	7000 Hz	Počet změřených bodů za vteřinu
Maximum moving speed	1 m/s	Maximální lineární rychlost při které není zkresleno měření
Accuracy	<0,02m	Přesnost měření
Frequency of data refresh	8 Hz	Frekvence obnovení dat o měření

Tabulka 3: Parametry měření modulu M1M1

(Zdroj: vlastní tabulka, hodnoty převzaty z [16])

Následující tabulka obsahuje parametry LASERu použitého pro měření.

Laser wavelength	905nm	Vlnová délka paprsku
Laser power	28W	Výkon laseru
Pulse length	10ns	Délka pulsu
Laser safety class	IEC-60825 Class 1	Bezpečnostní třída

Tabulka 4: Parametry LASERu použitého pro měření

(Zdroj: vlastní tabulka, hodnoty převzaty z [16])

Notebook HP 4740s. Dalším stěžejním prvkem je pro orientační systém zařízení, na kterém se data zpracovávají a dále předávají uživateli. V mém případě je to notebook od společnosti Hewlett-Packard, konkrétně model 4740s.

3.1.2 Software

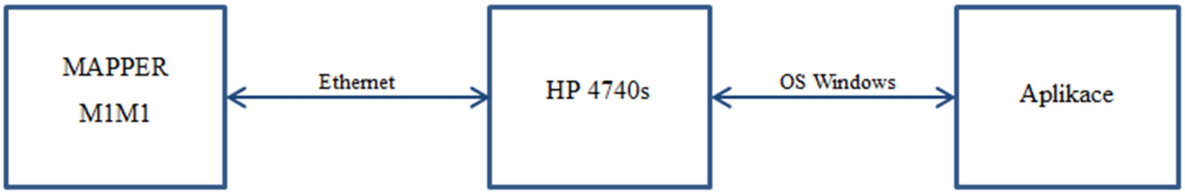
Pokud se budeme zabývat softwarovou stránkou týkající se samotného modulu, bylo by dobré zmínit zabudovanou funkci SLAM (Simultaneous localization and mapping). To v překladu znamená současnou lokalizaci např. robota a zároveň mapování jeho okolí. V dnešní době je tato metoda velice opěvovaná vzhledem k předpokladu, že by mohla v budoucnu dopomoci k vytvoření skutečně autonomních robotů. Jelikož bude v konkrétním případě systémem pohybovat uživatel, zůstává funkce mapování a lokalizace nevyužita [15].

Slamware C++ SDK. Pro detekci překážek budeme používat naměřená data přímo z LIDARu. Ty lze získat za pomoci SDK (Software development kit). SDK je sada vývojových nástrojů, která je volně k dispozici na oficiálním webu výrobce (*http://www.slamtec.com/en/Support*). Pro námi použitý modul nabízí výrobce několik typů SDK na platformy Windows, iOS, Android, Linux a také pro ROS (Robot Operating System), což je operační systém sloužící pro vývoj robotů. Na webu si vybereme používanou platformu a stáhneme příslušné SDK. Pomocí nadřazené aplikace, která využije tyto nástroje, lze poté přistupovat přímo k potřebným datům. [16].

Microsoft Visual Studio 2010. Pro vývoj nadřazené aplikace, která bude zpracovávat data z LIDARu, použijeme vývojové prostředí tzv. IDE (Integrated Development Environment) Microsoft Visual Studio 2010. Verzi 2010 jsem volil z důvodu, že ostatní verze (2017, 2019) jsou zatím pouze v testovacím tzv. beta provozu. Editor kódu vývojového prostředí podporuje v základním nastavení minimálně 4 programovací jazyky, další je možné nainstalovat rozšířením. Prostředí obsahuje taktéž designer formulářů pro tvoru aplikací s grafickým rozhraním. Pro naši aplikaci zvolíme programovací jazyk C++. Aplikace bude konzolového typu. Konkrétní výběr je nutný z důvodu kompatibility, jelikož zmiňované SDK nepodporuje aplikace typu Windows Form.

3.2 Návrh systému

Po představení použitých komponent se dostáváme k dalšímu kroku. Tím nemůže být nic jiného než vytvoření funkčního systému pro orientaci v objektech prostřednictvím zvolených zařízení. Vytvoření funkčního systému znamená spojení zvolených zařízení v celek, který bude schopen odhalit překážky nacházející se v okolí postiženého a následně vytvoří jejich seznam, který předá přídavným systémům. Pro lepší představu je přiložen návrh blokového schéma systému.



Obrázek 11: Blokové schéma systému

(Zdroj: vlastní tvorba)

Nyní si shrneme požadovanou funkčnost jednotlivých bloků. První je na blokovém schématu modul M1M1, který cca 20s po zapnutí trvale skenuje okolí v reálném čase. Výsledky měření by se následně měly dostat pomocí Ethernetu na notebook. Na něm běží OS Windows, kde bude spuštěna vlastní konzolová aplikace. Zde dojde ke zpracování dat a výstupem bude seznam překážek předávaný přídavným systémům. Pro lepší představu funkčnosti aplikace je zde přidána funkce výpisu překážek v opakujícím se časovém intervalu.

3.3 Implementace

V této části práce se detailně podíváme na realizaci jednotlivých částí a jejich následné propojení.

Na samotném modulu není třeba provádět žádné úpravy. Kabel pro napájení dodávaný s modulem jsem připojil k notebooku a pomocí tlačítka zapnul modul.

3.3.1 Komunikace

Po zapnutí mapovacího modulu dochází automaticky k vytvoření AP (přístupového bodu) pro bezdrátové připojení. U mapovacího modulu M1M1 lze pro komunikaci využít WiFi nebo Ethernet v následujících standardech:

Rozhranní	Standard	Rychlost
Ethernet	802.3/802.3u	10/100M
WiFi	802.11a/b/g/n/ac	-

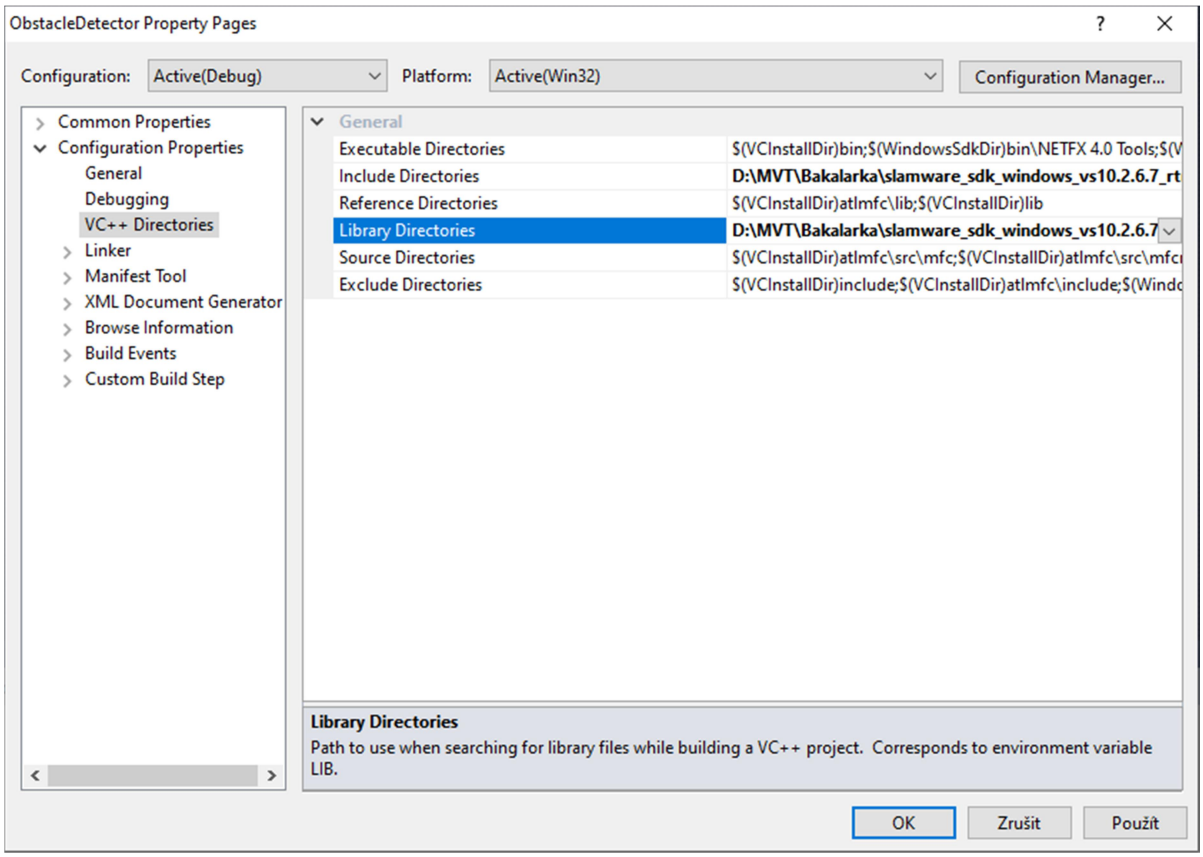
Tabulka 5: Možnosti komunikace s modulem M1M1

(Zdroj: vlastní tabulka, hodnoty převzaty z [16])

Při pokusu o první připojení jsem narazil na problém. Použitý notebook nenalezl v dostupných sítích přístupový bod pro bezdrátové připojení vytvořené modulem. Pro ověření funkčnosti bezdrátového připojení modulu jsem použil mobilní telefon, který síť nalezl a bez problémů se na ni připojil. Ani po několikahodinovém bádání jsem nebyl schopen uvést do provozu bezdrátové připojení na notebooku a musel jsem se tedy smířit s komunikací přes ethernetový kabel. Ačkoli nebude komunikace skrze kabel příliš komfortní, slibuje lepší stabilitu. Bezchybnost komunikace pomocí ethernetu jsem ověřil pomocí aplikace RoboStudio, kterou nabízí výrobce zdarma ke stažení na oficiálním webu (*http://www.slamtec.com/en/Support*).

3.3.2 Aplikace pro zpracování dat

Nyní se dostáváme k samotnému programování aplikace pro zpracování naměřených dat. Prvním krokem v IDE je založení nového projektu konzolové aplikace v jazyce C++ a import knihoven Slamware SDK. Návod na import knihoven do projektu je k dispozici v oficiální dokumentaci k SDK. Na přiloženém obrázku lze vidět okno vlastností projektu, kde import nastavuji.



Obrázek 12: Import knihoven do projektu

(Zdroj: vlastní tvorba)

Dalším krokem je vytvoření platformy pro spojení s modulem. Tato část programu musí být v bloku Try/Catch z důvodu zachycení případných výjimek. Pro vytvoření spojení byla využita již existující metoda *SlamwareCorePlatform::connect("host", port, timeout)* která nese následující parametry:

- "host" (string) IP adresa modulu
- port (int) Port modulu
- timeout (int) Maximální čas pro pokus o spojení v milisekundách

Pro ověření spojení jsem na vytvořené platformě využil metodu *getSDPVersion()*, která pomocí řetězce *string* vrací aktuální verzi SLAMWARE systému. Zpracování těchto kroků a konkrétní parametry metody *connect* vidíme na přiloženém obrázku č.13. Spodní část obrázku znázorňuje výstup zpracovaného kódu.

```
#include <rpos/robot_platforms/slamware_core_platform.h>
#include <iostream>
#include <Windows.h>

using namespace std;
using namespace rpos::robot_platforms;

int main(int argc, char * argv[]){

    SlamwareCorePlatform platform;
    try{
        cout << "Connecting to the module..." << endl;
        platform = SlamwareCorePlatform::connect("192.168.11.1", 1445, 10000);
    }catch(rpos::robot_platforms::ConnectionTimeOutException exc){
        cout << "CAN'T CONNECT TO THE MODULE. CHECK IP ADRESS AND CABLE!" << endl;
        Sleep(5000);
        return 1;
    }

    cout << "Base version: " << platform.getSDPVersion() << endl;
    Sleep(5000);
    return 0;
}
```

```
c:\users\lumik\documents\visual studio 2010\Projects\ObstacleDetector\Debug\ObstacleDetector.exe
Connecting to the module...
Base version: 2.0.0_dev-sdp-20161008
```

Obrázek 13: Kód a výstup pro první spojení s modulem

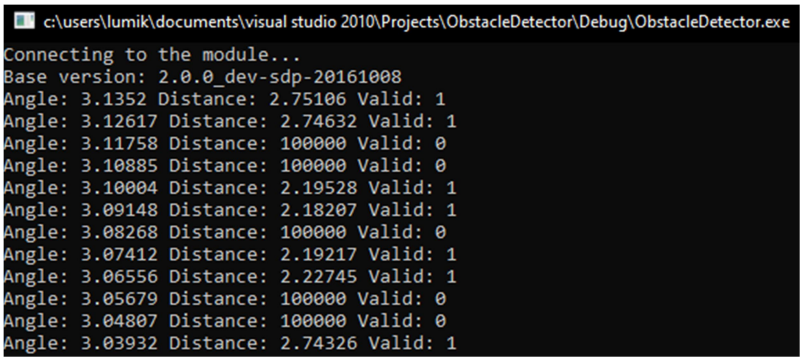
(Zdroj: vlastní tvorba)

Po spuštění aplikace dostávám v konzolovém okně informaci o verzi, znamená to, že spojení funguje.

Dalším krokem je získání naměřených dat. Začínám vytvořením instance třídy *LaserScan,* která uchovává sérii objektů *LaserPoint*, ty představují jednotlivé paprsky z měření LIDARu. Pomocí metody *getLaserPoints()* dostávám sérii paprsků, která reprezentuje měření z jedné otáčky LIDARu. Následně jednotlivé paprsky série ukládám do vektoru pomocí cyklu *for*. Vektor je dle [17] sekvenční kontejner zapouzdřující pole s proměnným počtem prvků. Zabírá více paměti než obyčejné pole, protože je alokována paměť navíc pro budoucí růst vektoru. Tímto stylem je bohužel kladena větší náročnost na výpočetní výkon

stroje, při zdejším použití je ale tento nárůst složitosti zanedbatelný. Pro použití vektoru jsem se inspiroval od vývojářů firmy Slamtec, kteří ho v příkladech práce s SDK využívali také.

Pomocí cyklu *for* vypisuji vektor a dostávám první měřená data. Příklad těchto dat, která zatím nejsou nijak zpracována můžeme vidět na přiloženém obrázku č.14.



Obrázek 14: První naměřená data

(Zdroj: vlastní tvorba)

Jak můžeme vidět na výpisu, objekt *LaserPoint* má 3 parametry:

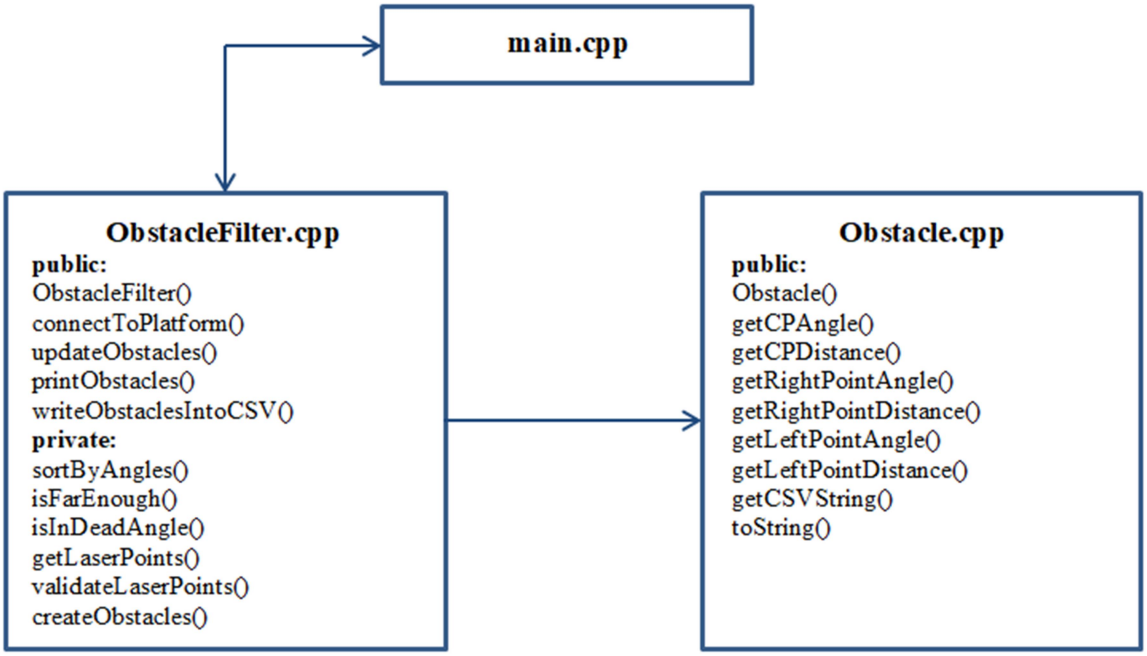
- angle (float) úhel jednotlivého paprsku
- distance (float) naměřená vzdálenost jednotlivého paprsku
- valid (bool) platnost měření

Úhel je naměřen v radiánech, lépe si ale pozici překážky představíme ve stupních, proto bude nutné založit proměnnou pro přepočet. Vzdálenost je udávaná v metrech, což je v pořádku. Platnost měření je vyhodnocována přímo samotným modulem. Pokud je měření platné, dostávám od modulu hodnotu parametru „valid“ = 1.

Do této chvíle jsem celý kód psal pouze v rámci souboru *main.cpp*. V momentě kdy pouze vypisuji naměřená data z jedné otočky LIDARu je to adekvátní řešení. Z důvodu předpokládaného nárůstu funkcí pro zpracování a případných změn v programu bude dobré zavést objektovou strukturu programu.

Myšlenkou objektového návrhu je rozdělení programu do jednotlivých tříd, kdy každá třída má na starosti konkrétní funkce. Interakce mezi jednotlivými třídami pak znamená samotný chod aplikace. Výhodou tohoto systému je snazší údržba aplikace a řešení případných změn a rozšíření. Nevýhodou je rozsáhlejší kód a prvotní vývoj aplikace, který je složitější.

Struktura vlastní aplikace je znázorněna zde:



Obrázek 15: Znázornění struktury aplikace

(Zdroj: vlastní tvorba)

Soubor *main.cpp*, kde bude hlavní část programu je již vytvořený. Nyní je třeba vytvořit třídu Obstacle a ObstacleFilter. K jednotlivým třídám vytvářím také příslušné hlavičkové soubory s příponou *.h*. Tyto soubory obsahují deklarace metod a proměnných jednotlivých tříd.

Obstacle.cpp. Třída obsahuje veškeré informace o překážce. Nese následující parametry:

- cPdistance (float) vzdálenost nejbližšího bodu překážky
- cPangle (float) úhel nejbližšího bodu překážky
- rightPointAngle (float) úhel pravého bodu překážky
- rightPointDistance (float) vzdálenost pravého bodu překážky
- leftPointAngle (float) úhel levého bodu překážky
- leftPointDistance (float) vzdálenost levého bodu překážky

Zmíněné parametry jsou privátní. Proto třída obsahuje tzv. GETové metody znázorněné ve struktuře aplikace, které při zavolání vracejí hodnotu parametru. Dále jsou zde dvě metody sloužící pro výpis překážek. Metoda *toString()* vypisuje překážky přímo v rámci běžící

aplikace. Metoda *getCSVString()* zajišťuje formát vhodný pro výpis aktuálních překážek do souboru s příponou CSV. Jednotlivé hodnoty překážek jsou v následujícím formátu:

leftPointAngle;leftPointDistance;cPangle;cPdistance;rightPointAngle;rightPointDistance

ObstacleFilter.cpp. Třída sloužící pro veškeré zpracování měřených dat. Data postupně zpracovávají jednotlivé metody, které si nyní představíme.

Zpracování dat začíná jejich načtením, to zastává metoda *getLaserPoints()*. Výstupem z metody je vektor obsahující jednotlivé paprsky seřazené podle úhlu pomocí privátní metody *sortByAngles()*. Pokud nechceme pouze data z jedné otočky LIDARu, lze metodě zadat počet měření pomocí proměnné *numberOfMeasures* , která zajistí více naměřených dat.

Dalším krokem při zpracování je tzv. validace naměřených dat. Každý paprsek nese parametr „valid“, jak již bylo zmíněno výše. Pokud byl paprsek systémem vyhodnocen jako vadný, nebude do vektoru uložen. Současně filtruji paprsky podle vzdálenosti. Pokud je vzdálenost změřeného bodu delší než 1m, paprsek je vyřazen. Tuto filtraci obstarává metoda *validateLaserPoints().*

Vytváření jednotlivých překážek obstarává metoda *createObstacles().* Prochází vektor validovaných paprsků a na základě daných kritérií vytváří jednotlivé překážky. Pevně zvolená hodnota rozestupu mezi jednotlivými překážkami je zvolena na 3°. Při použití modifikovaného vzorce na obvod kruhu lze vypočítat rozestup jednotlivých překážek pro vzdálenost 1m.

$$l = \left(\frac{\pi \cdot 2}{360}\right) \cdot 3 \qquad (2)$$

Pokud je rozestup mezi paprsky při vzdálenosti 1m větší než l = 0,052m budou vytvořeny dvě samostatné překážky. Vzdálenost l se samozřejmě s přiblížením překážky k modulu úměrně zmenšuje, protože se zmenšuje poloměr. Další parametry překážky se získávají porovnáváním aktuálního paprsku s předchozím při průchodu vektorem validovaných paprsků.

Všechny dosud zmíněné metody jsou privátní. Proto byla vytvořena veřejná metoda *updateObstacles()*, která postupně zavolá výše zmíněné metody a vytvoří seznam překážek uložený do vektoru. Jako parametr mohu metodě předat požadovaný počet sérií paprsků,

které má metoda zpracovat. Zde se jedná o předání hodnoty počtu měření pro metodu *getLaserPoints()* zmiňovanou výše. Při používání tohoto parametru je potřeba myslet na maximální frekvenci obnovení dat o měření, která je 8Hz.

Další veřejná metoda *printObstacles()* slouží pro výpis překážek. Výpis překážek probíhá cyklem *for*. Cyklus prochází vektor vytvořených překážek a postupně jednu po druhé vypisuje. Výpis překážek není primárním požadavkem na aplikaci, slouží pouze pro lepší představu, jak aplikace funguje a jaká data budou předávány přídavným systémům.

Předání dat přídavným systémům je řešeno pomocí souboru „output.csv“. Tento soubor je vytvořen metodou *writeObstaclesIntoCSV()*. Postupně se projde celý vektor překážek a u každé se používá již zmíněná metoda *getCSVString()* z třídy *Obstacle*. Každý řádek souboru tedy prezentuje jednu překážku s jednotlivými parametry, které jsou odděleny středníkem. Nutno zmínit, že na rozdíl od klasického výpisu překážek není ve výstupním souboru úhel převeden na stupně, ale zůstává uveden v radiánech. Tato skutečnost by měla zajistit pohodlnější zpracování pro přídavné systémy.

main.cpp. Zde se odehrává samotný běh aplikace. Při spuštění aplikace se vytvoří nová platforma a proběhne připojení k modulu.

Následně je uživatel dotázán na mód, v kterém má být aplikace spuštěna. Pokud bude chtít uživatel tlačit modul pravou rukou, zadává požadavek na pravoruký režim vyhodnocení překážek pomocí písmena „R“. Naopak pro režim levé ruky slouží písmeno „L“.

Po zjištění režimu dochází k vytvoření filtru, který zpracovává jednotlivé paprsky. Pokud je zvolen režim pravý, paprsky od úhlu 100° do 165° nejsou zpracovávány, protože se s největší pravděpodobností jedná o nohy samotného uživatele. Pro levý režim jsou vynechány paprsky od úhlu -165° do -100°.

Nyní se dostáváme k samotnému cyklu měření a vyhodnocení paprsků včetně výpisu překážek. Tuto část aplikace tvoří nekonečný cyklus *while(1)*. V rámci tohoto cyklu se provádí metoda *updateObstacles()*, která zajistí aktuální data z modulu a následně se provede metoda *printObstacles()*, která aktuální překážky zprostředkuje uživateli. Každých 5 vteřin dostává uživatel nový výpis překážek s aktuálními daty. Cyklus lze přerušit pomocí stisku písmena „Q“. Jakmile je stisknuto písmeno „Q“ v momentě, kdy by mělo dojít k novému měření, je cyklus ukončen.

Než dojde k ukončení aplikace, jsou nejaktuálnější data o překážkách automaticky zapsány do souboru „output.csv“ pomocí funkce *writeObstaclesIntoCSV()*.

3.4 Testování modulu

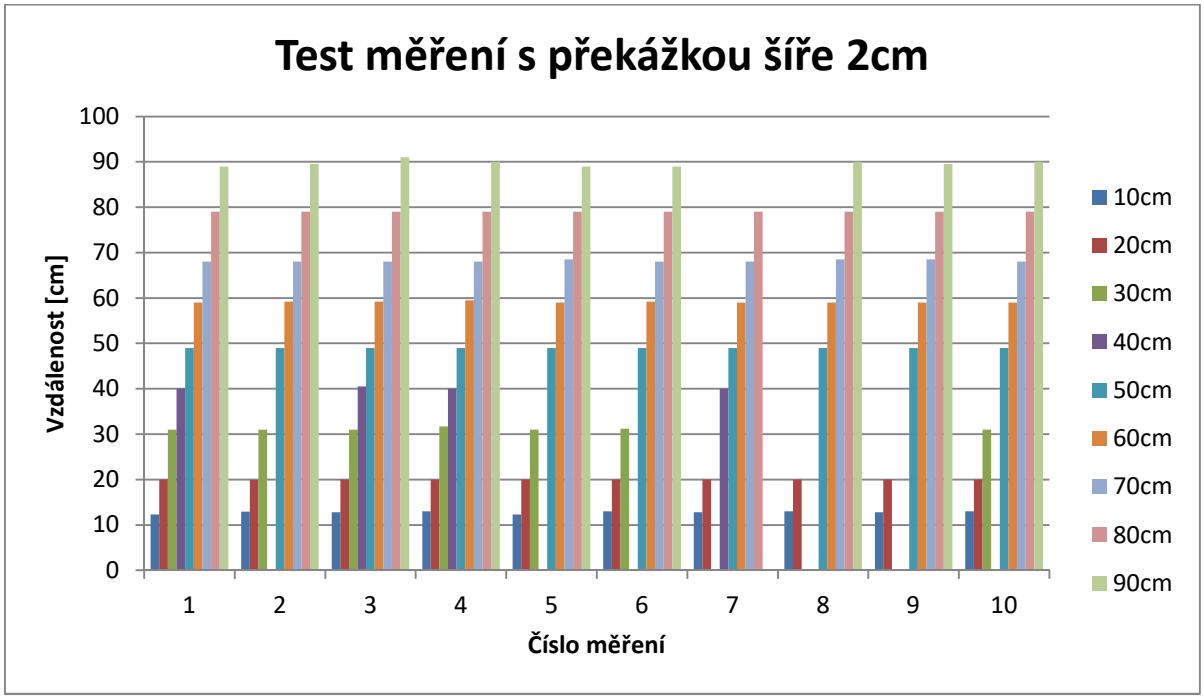
Po dokončení aplikace přichází na řadu otestování modulu. Je nutné otestovat stabilitu aplikace, ale v prvé řadě přesnost měření.

Testování přesnosti jsem provedl za pomoci papírové krabice bílé barvy o šířkách 2, 4, 6, 8 a 10cm. Vzdálenost překážky od modulu jsem vždy nastavil pomocí laserového dálkoměru.

Zmiňovaná hodnota „Přesnost [%]“ při vyhodnocení výsledků měření je počítána podle vzorce (3). Je to hodnota průměrné relativní odchylky všech měření dané vzdálenosti a konkrétní šíře překážky, odečtená od hodnoty 100. Při výpočtu přesnosti měření nebyly zahrnuty hodnoty, kdy nebyla překážka detekována.

$$100 - \frac{\sum_{i=1}^{n} \left| \frac{MH_i - SH_i}{SH_i} \right| \cdot 100}{n} \tag{3}$$

Výsledky testování modulu jsou znázorněny v následujících grafech.

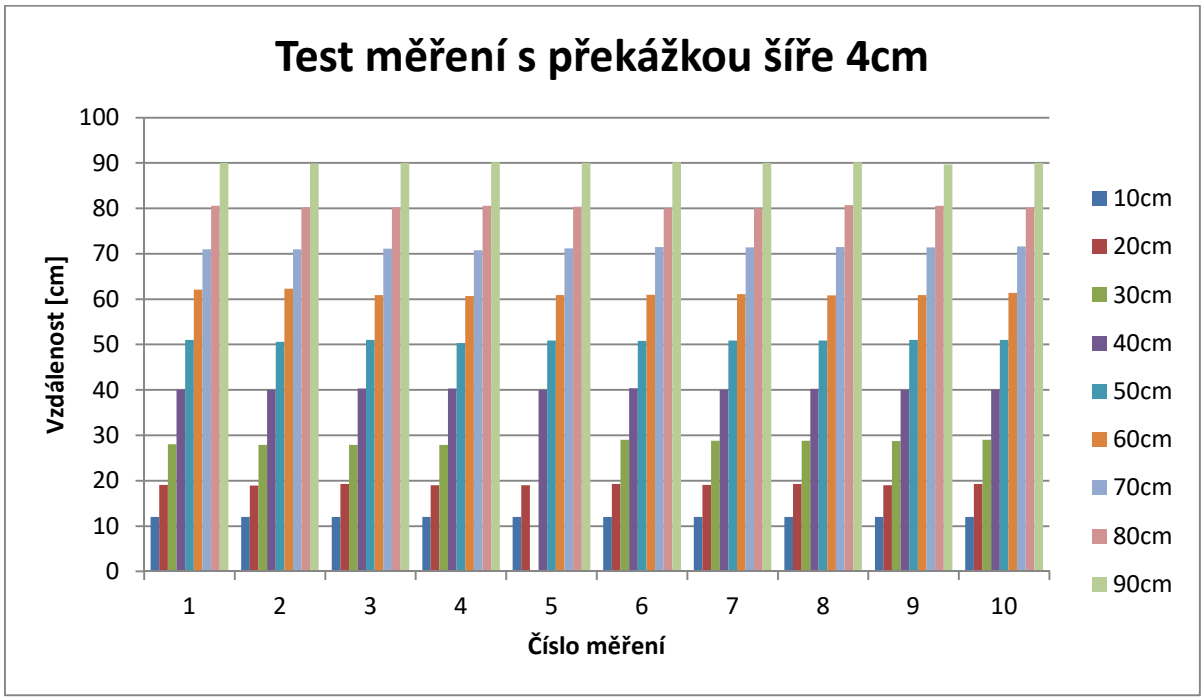


Obrázek 16: Graf č.1 – Přesnost měření s překážkou šíře 2cm

(Zdroj: vlastní tvorba)

Z grafu č.1 vidíme, že nejméně přesný je modul při měření překážky ve vzdálenosti 10cm. I za předpokladu, že by byla zohledněna tolerance měření přesnosti 2cm zmiňovaná v parametrech modulu, se přesto všechny hodnoty liší od hodnoty nastavené kontrolním měřidlem. Vzhledem k předpokládanému používání pro detekci překážek v rámci jednoho metru se nepředpokládá, že by se náhle objevila překážka 10cm od nevidomého. Vždy by mělo dojít k postupnému přibližování a nepřesnost měření při této vzdálenosti může být tedy zanedbána. Důležitá je detekce, která byla úspěšná. Při pohledu na výsledky z měření vzdálenosti 20cm vidíme, že toto měření bylo velmi přesné. Pouze v jediném případě se nám hodnota liší a to pouze o 0,1cm. Problém nastává při vzdálenostech 30 a 40cm. Zde modul v devíti případech z dvaceti překážku vůbec nedetekoval. Průměrná úspěšnost detekce pro vzdálenosti 30 a 40cm byla tedy pouze 55%. Příčina špatné detekce a možná řešení problému budou zmíněna v kapitole týkající se optimalizace, nyní se pojďme věnovat dalším měřením. Testování na vzdálenostech 50-80cm lze považovat za úspěšné, detekce překážky dosáhla hodnoty 100% a přesnost měření vzdálenosti 98,2%. Na vzdálenosti 90cm se jedenkrát nepodařilo překážku detekovat, mimo to byla ale měření vzhledem k velikosti překážky velmi přesná.

Výsledky testu s překážkou o šíři 4cm znázorňuje graf č.2.

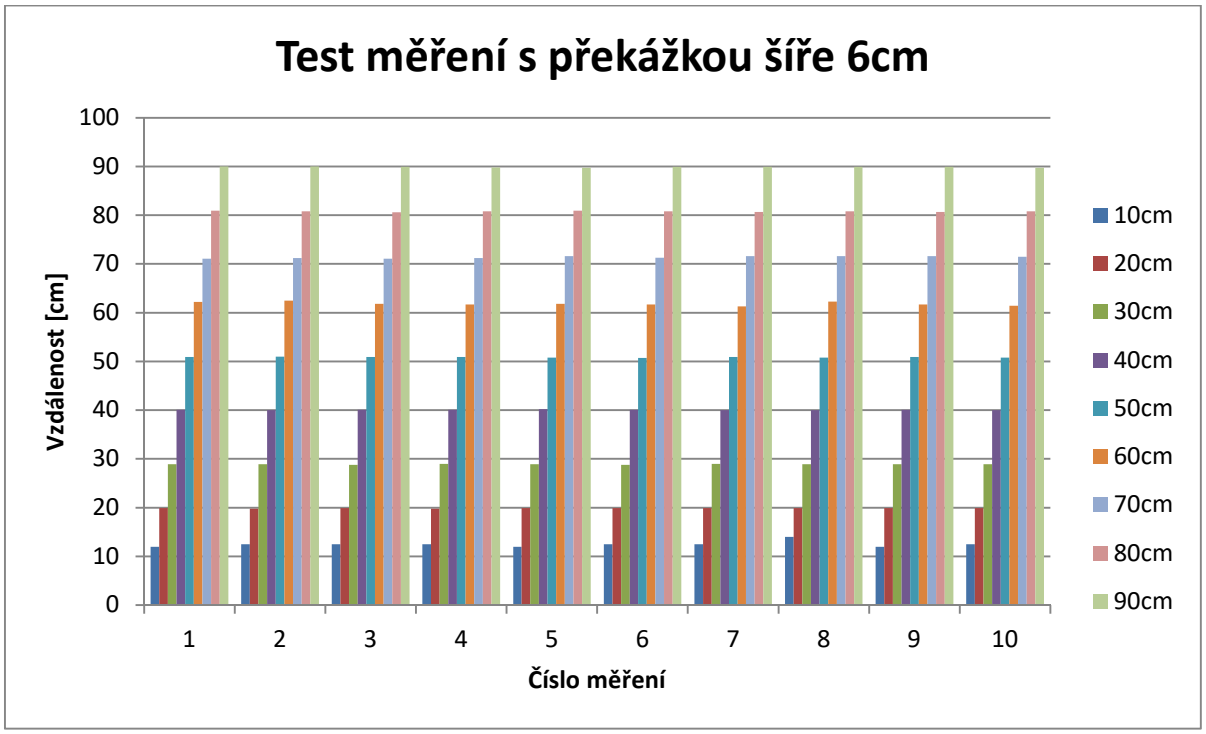


Obrázek 17: Graf č.2 – Přesnost měření s překážkou šíře 4cm

(Zdroj: vlastní tvorba)

Při šířce překážky 4cm byl problém s detekcí pouze v jediném případě ze všech devadesáti měření, což vystihuje průměrná detekce 98,9%. Měření byla velmi přesná. Výsledný průměr přesnosti 96% nejvíce ovlivnily hodnoty měřené při vzdálenosti 10cm, kdy bylo pokaždé naměřeno 12cm.

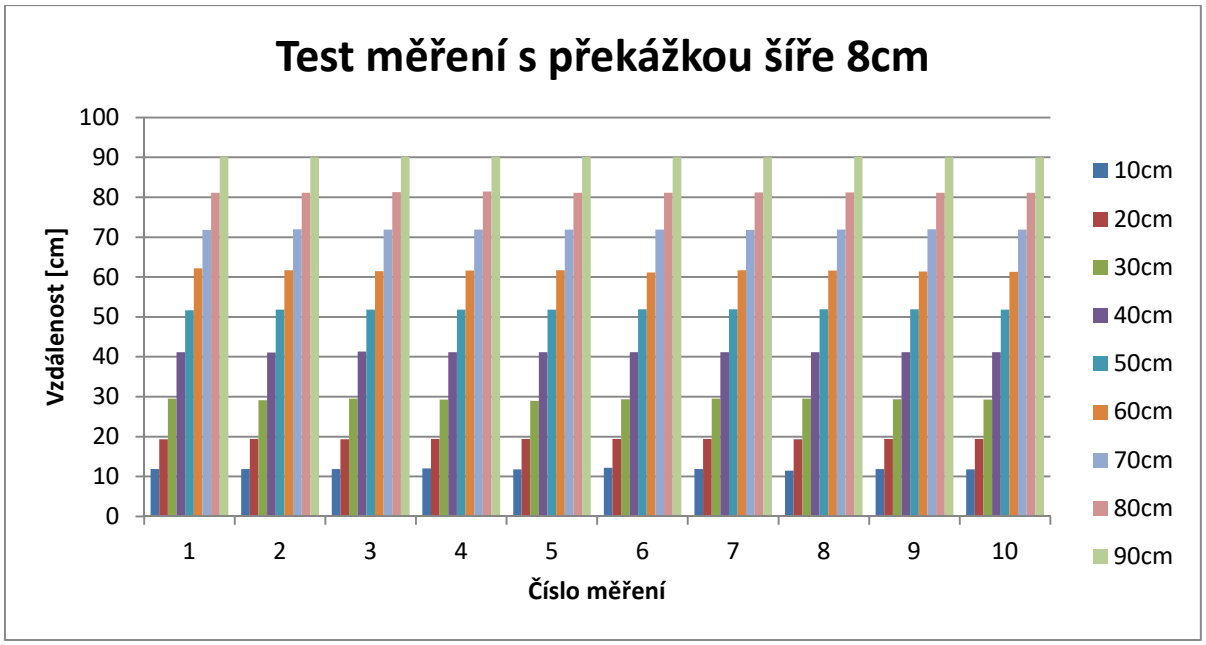
Podobných výsledků bylo dosaženo také s překážkou o šíři 6cm. Průměrná detekce se vyšplhala na krásných 100% a přesnost měření 95,8% negativně ovlivnilo opět pouze měření ze vzdálenosti 10cm což dokazuje graf č.3.



Obrázek 18: Graf č.3 – Přesnost měření s překážkou šíře 6cm

(Zdroj: vlastní tvorba)

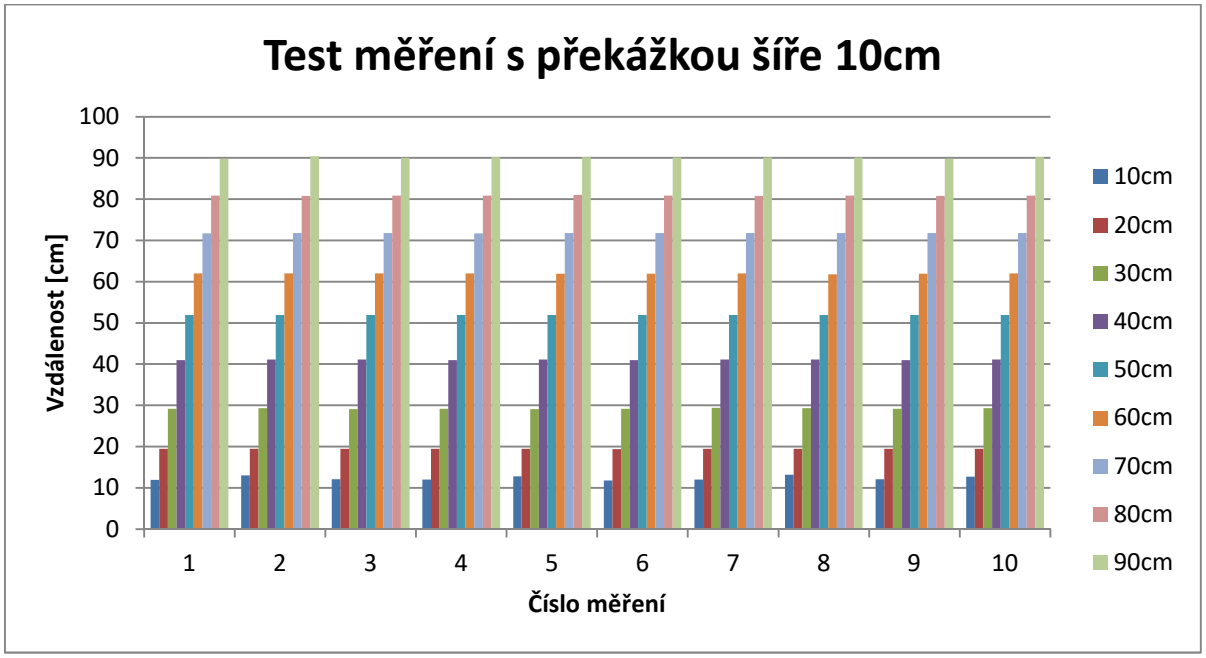
Při testování překážky šíře 8cm nebyl také žádný problém, krom již zmíněné horší přesnosti měření při blízké vzdálenosti překážky. Tento výrok potvrzují výsledky, kdy byla detekce opět 100% a přesnost měření vzdálenosti 95,8%. Znázorněno na grafu č.4 níže.



Obrázek 19: Graf č.4 – Přesnost měření s překážkou šíře 8cm

(Zdroj: vlastní tvorba)

Detekce překážky široké 10cm byla opět 100%, přesnost měření ale oproti předešlým výsledkům mírně klesla na hodnotu 95,3%. Důvodem byly horší výsledky měření při vzdálenostech 50 a 60cm. Pokud vezmeme v úvahu toleranci přesnosti 2cm udávanou v parametrech modulu, byly výsledky v určených mezích přesnosti.



Obrázek 20: Graf č.5 – Přesnost měření s překážkou šíře 10cm

(Zdroj: vlastní tvorba)

Kompletní výsledky testování, včetně grafů, výpočtů odchylek a přesnosti jsou k dispozici v souboru s názvem „testMereni.xlsx“ na přiloženém CD.

K testování polohy překážek jsem vytvořil podložku o velikosti 30x42cm, kde jsou nakresleny jednotlivé paprsky úhlů 0°-360° upravené podle požadavků modulu na interval -180° až 180°. Po umístění překážky jsem vždy k nakreslenému paprsku na podložce přiložil rovnou dřevěnou lať a pomocí ní ověřil skutečnou polohu překážky. Pokud se u měření vzdálenosti bavíme o průměrné přesnosti 95,7%, která byla snížena především měřením při vzdálenosti 10cm, tak u testování polohy musím modul vyzdvihnout. Z hlediska polohy byla měření tak přesná, že je v některých případech nebylo možné řádně ověřit. Pomocná podložka umožňuje kontrolní měření po 1°, výsledky z modulu jsou ale s přesností na desetinu stupně.

Další částí testů modulu mělo být otestování na přesnost měření a detekci v případě použití různých materiálů. Tuto část zpracuji v rámci testování celého systému, které bude provedeno pomocí pohybu po bytové jednotce a bude tak zajištěna různorodost povrchů u jednotlivých překážek.

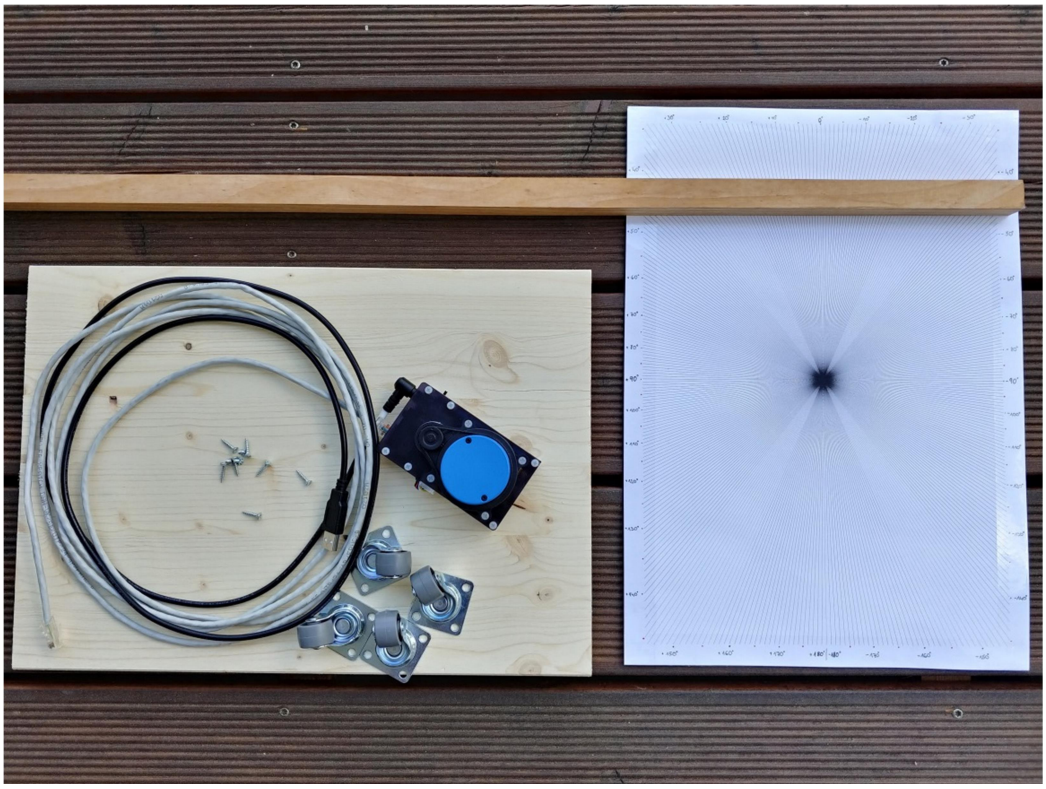
Posledním krokem testování bylo ověření stability aplikace. Tento atribut jsem ověřil pomocí nepřetržitého provozu aplikace po dobu 8mi hodin. Za tuto dobu bylo provedeno 5015 měřících cyklů bez sebemenších problémů. Vzhledem k výsledku testu mohu aplikaci prohlásit jako vhodnou pro použití.

3.5 Kompletace a ověření funkčnosti

Po testování přesnosti modulu přichází na řadu kompletace všech komponent do celistvého systému, který bude s pomocí nevidomého schopen pohybu po jeho boku. Následně je také nutné otestovat celý systém simulováním zamýšleného použití.

3.5.1 Kompletace

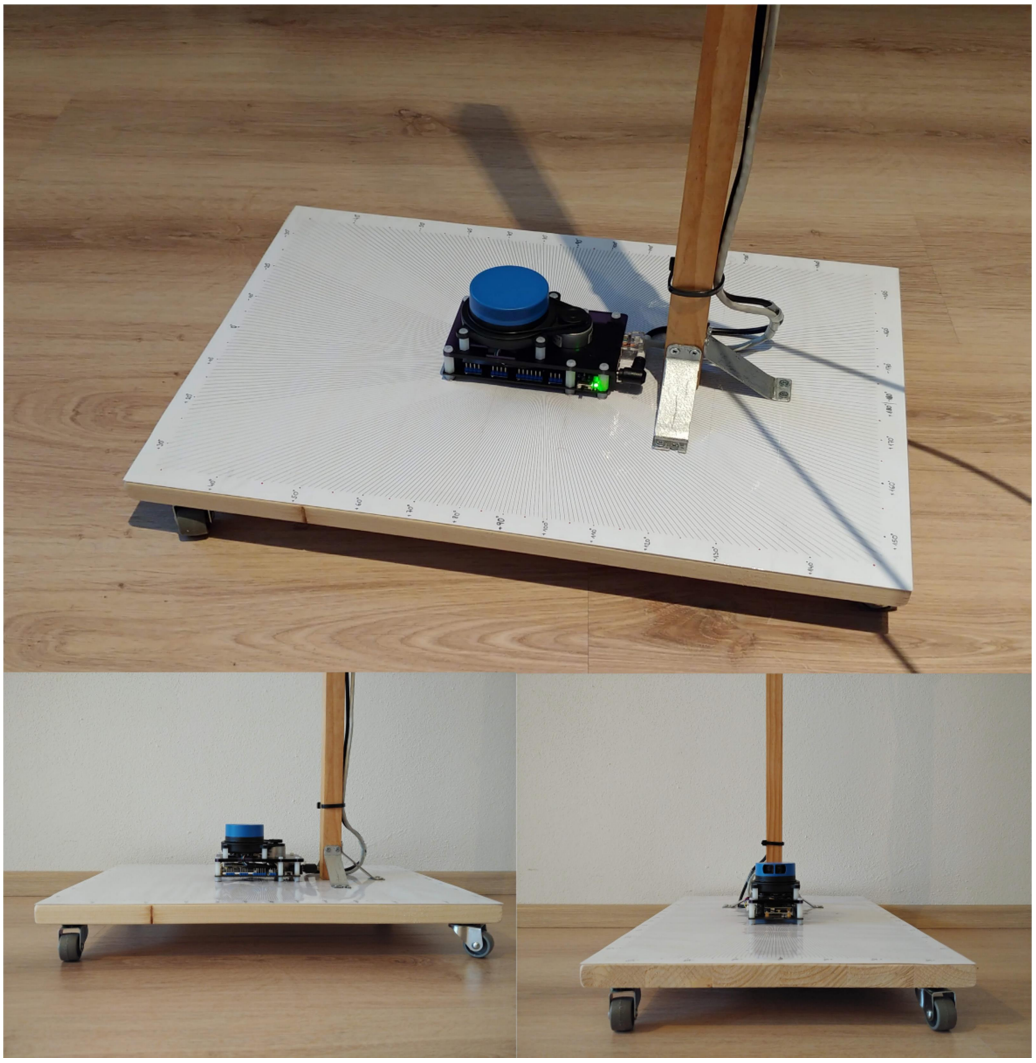
Přehled veškerých komponent pro navigační systém kromě notebooku s vyhodnocovací aplikací je na přiloženém obrázku níže.



Obrázek 21: Komponenty pro kompletaci systému

(Zdroj: vlastní tvorba)

Pohyb celého systému je možný díky nábytkovým kolečkům, která jsem připevnil samořeznými vruty na spodní stranu dřevěné desky. Před instalací mapovacího modulu k desce jsem na ni přilepil papírovou šablonu pro kontrolu úhlů. Samotný modul je připevněn pomocí závitové tyče M3, kterou jsem do desky zašrouboval v požadovaných rozměrech. Rozměry byly určeny distančními sloupky měřícího modulu, pomocí kterých je prvek k desce přišroubován. Dále jsem k desce pomocí vrutu přišrouboval i dřevěnou tyč. Pomocí této tyče bude nevidomý s měřícím systémem pohybovat. Vzhledem k délce tyče (110cm) bylo po prvním testování nutné připevnění tyče vyztužit pomocí předem ohnutých plechů, aby byla dostatečně stabilní pro pohyb systémem. Výsledek mého snažení je znázorněn pomocí následující koláže.



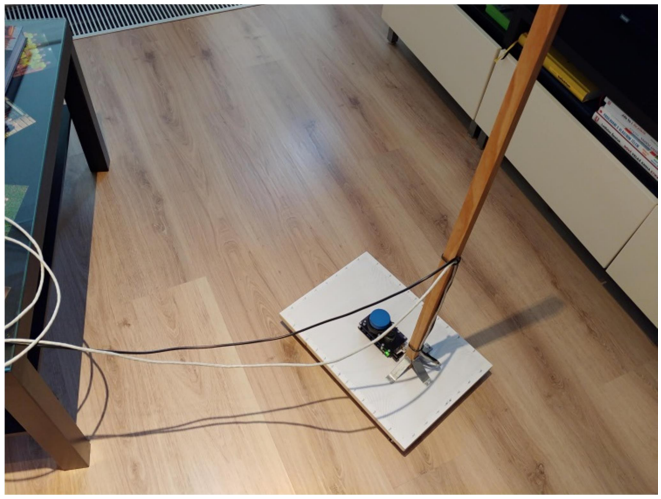
Obrázek 22: Koláž znázorňující navigační systém třemi pohledy

(Zdroj: vlastní tvorba)

3.5.2 Ověření funkčnosti

Po kompletaci systému bylo možné přistoupit k ověření funkčnosti. V první části testu ověřím přesnost modulu po kompletaci na dřevěnou desku. Druhá část testu bude provedena pomocí pohybu po bytové jednotce, jak bylo zmíněno v kapitole testování modulu.

V první části jsem systém umístil mezi televizní stolek s lesklým dekorem bílé barvy a konferenční stolek ze dřeva v barvě černé. Systém jsem spustil a navolil jsem režim pro levou ruku. Nechal jsem provést 13 měření, kde se jednotlivé výsledky lišily v řádech desetin. Mohu tudíž konstatovat, že přesnost systému je při měření z místa výborná. Výsledky testu dokládají obrázky níže, kde můžeme vidět i formát výstupního souboru pro přídavné systémy.



Obrázek 23: Rozložení překážek při testování systému z místa

(Zdroj: vlastní tvorba)

```
C:\Users\Lumik\Documents\Visual Studio 2010\Projects\ObstacleDetector\Debug\ObstacleDetector.exe
Starting new measuring cycle:
Obstacle LP: -54.4deg. and 0.999m  Obstacle CP: -108.8deg. and 0.619m  Obstacle RP: -124.9deg. and 0.645m
Obstacle LP: 13.7deg. and 0.975m  Obstacle CP: 11.1deg. and 0.948m  Obstacle RP: 10.8deg. and 0.975m
Obstacle LP: 73.5deg. and 0.685m  Obstacle CP: 71.1deg. and 0.672m  Obstacle RP: 70.2deg. and 0.697m
Obstacle LP: 132.8deg. and 0.999m  Obstacle CP: 128.3deg. and 0.928m  Obstacle RP: 127.8deg. and 0.951m
Number of obstacles: 4
Measuring cycle 9 ended.
Starting new measuring cycle:
Obstacle LP: -54.3deg. and 1.000m  Obstacle CP: -108.9deg. and 0.619m  Obstacle RP: -124.9deg. and 0.644m
Obstacle LP: 13.7deg. and 0.976m  Obstacle CP: 11.9deg. and 0.945m  Obstacle RP: 11.0deg. and 0.972m
Obstacle LP: 73.5deg. and 0.680m  Obstacle CP: 71.2deg. and 0.672m  Obstacle RP: 70.1deg. and 0.707m
Obstacle LP: 132.8deg. and 0.998m  Obstacle CP: 128.3deg. and 0.928m  Obstacle RP: 127.9deg. and 0.951m
Number of obstacles: 4
Measuring cycle 10 ended.
Starting new measuring cycle:
Obstacle LP: -54.3deg. and 0.997m  Obstacle CP: -109.1deg. and 0.620m  Obstacle RP: -124.9deg. and 0.646m
Obstacle LP: 13.7deg. and 0.976m  Obstacle CP: 11.9deg. and 0.942m  Obstacle RP: 10.8deg. and 0.982m
Obstacle LP: 73.3deg. and 0.685m  Obstacle CP: 71.1deg. and 0.669m  Obstacle RP: 70.3deg. and 0.695m
Obstacle LP: 132.8deg. and 0.998m  Obstacle CP: 128.2deg. and 0.927m  Obstacle RP: 127.7deg. and 0.952m
Number of obstacles: 4
Measuring cycle 11 ended.
Starting new measuring cycle:
Obstacle LP: -54.4deg. and 0.996m  Obstacle CP: -109.1deg. and 0.621m  Obstacle RP: -125.0deg. and 0.646m
Obstacle LP: 13.6deg. and 0.968m  Obstacle CP: 11.7deg. and 0.945m  Obstacle RP: 10.8deg. and 0.975m
Obstacle LP: 73.3deg. and 0.690m  Obstacle CP: 71.4deg. and 0.670m  Obstacle RP: 70.3deg. and 0.685m
Obstacle LP: 132.8deg. and 0.995m  Obstacle CP: 128.0deg. and 0.933m  Obstacle RP: 127.6deg. and 0.953m
Number of obstacles: 4
Measuring cycle 12 ended.
Starting new measuring cycle:
Obstacle LP: -54.4deg. and 0.996m  Obstacle CP: -109.3deg. and 0.621m  Obstacle RP: -124.9deg. and 0.647m
Obstacle LP: 13.7deg. and 0.976m  Obstacle CP: 11.9deg. and 0.943m  Obstacle RP: 10.8deg. and 0.971m
Obstacle LP: 73.3deg. and 0.683m  Obstacle CP: 70.3deg. and 0.674m  Obstacle RP: 70.1deg. and 0.702m
Obstacle LP: 118.6deg. and 0.544m  Obstacle CP: 109.1deg. and 0.505m  Obstacle RP: 105.9deg. and 0.568m
Number of obstacles: 4
Measuring cycle 13 ended.
Measuring ended.
Proceeding to write obstacles into CSV file: output.csv
CSV file created.
Press any key to continue . . .
```

Obrázek 24: Data měřená systémem při testování z místa

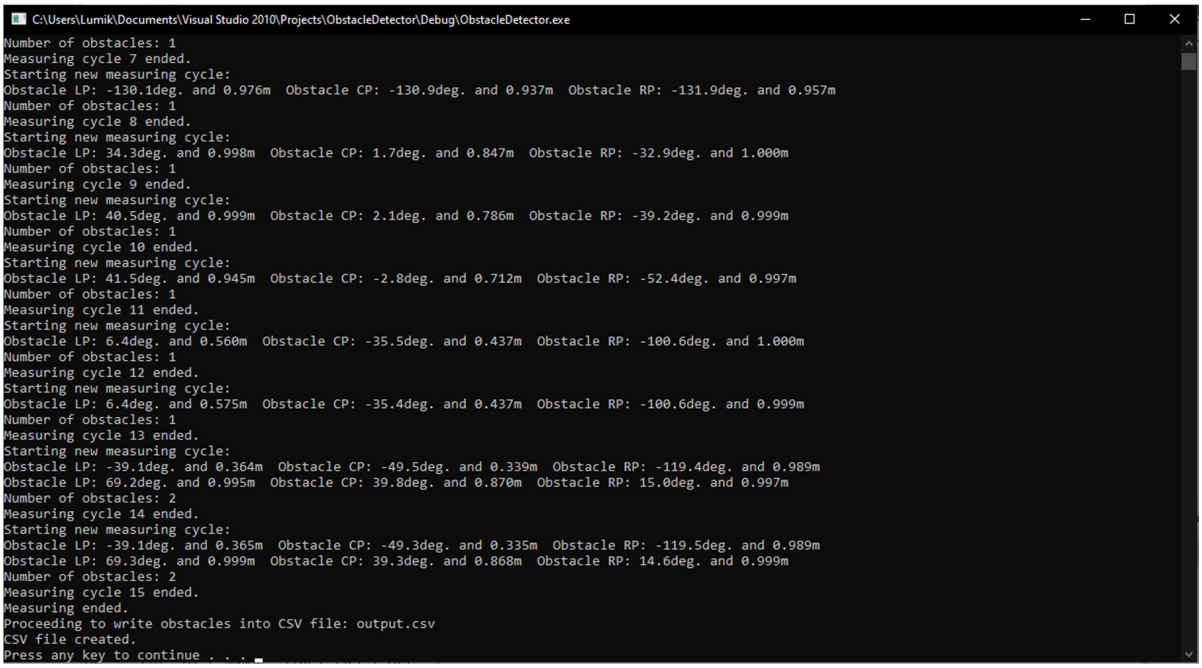
(Zdroj: vlastní tvorba)

```
Lister - [c:\Users\Lumik\Documents\Visual Studio 2010\Pr...
Soubor  Upravit  Možnosti  Kódování  Nápověda  100 %
-0.949213;0.996439;-1.90714;0.621247;-2.17924;0.646516
0.239058;0.975936;0.207785;0.942977;0.187867;0.97075
1.27892;0.68335;1.22782;0.673658;1.22311;0.702199
2.07024;0.543712;1.90497;0.505207;1.84792;0.568306
```

Obrázek 25: Příklad výstupního souboru pro přídavné systémy

(Zdroj: vlastní tvorba)

Další částí testování systému byl pohyb po místnosti. Pro tuto část testování jsem zvolil režim pohybu pravou rukou. Naměřená data jsou k dispozici zde.



Obrázek 26: Data při simulaci pohybu po místnosti

(Zdroj: vlastní tvorba)

Z přiložených dat můžeme pozorovat funkčnost systému. V měřícím cyklu 8 jsem minul nohu od jídelního stolu. K mému překvapení zde nebyl problém s detekcí i přes skutečnost, že je noha z nerezového materiálu. V cyklu 9 se poprvé detekuje v prostoru přede mnou kuchyňská linka s lesklým dekorem jako měl v předchozím testu televizní stolek. Cykly 10 a 11 odhalují nepatrné přiblížení směrem k lince. Při snaze vyhnout se překážce jsem zatočil mírně vlevo, což dokazuje měření č.12 a 13. Měření číslo 14 a 15 znázorňuje můj pokus trefit se do dveří vedle kuchyňské linky. Dveřní zárubně se skutečné nacházejí mezi úhly -39,1° a +15°. Výsledkem testování je závěr, že měření při pohybu je rovněž velmi přesné.

3.6 Optimalizace

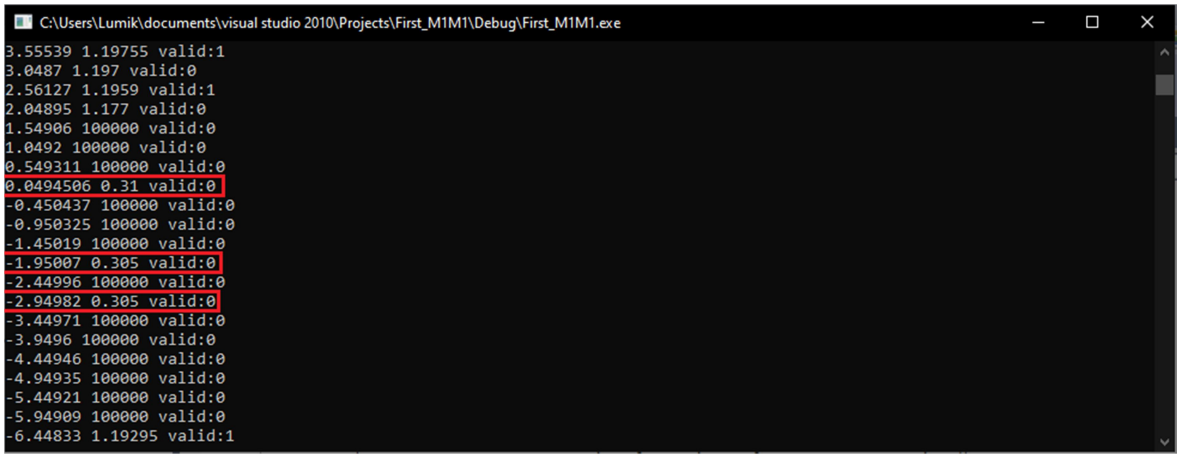
Při testování se objevilo hned několik nedostatků. Souhrn nedostatků a jejich řešení shrnuje tato podkapitola.

Jedním z prvních nedostatků byla nutnost restartu aplikace při výpadku komunikace. Na tento problém jsem narazil hned při začátku testování, když jsem spustil aplikaci dříve, než došlo ke spuštění samotného modulu. Proces připojení k modulu byl sice ošetřen blokem Try/Catch, takže aplikace vypsala hlášku „CONNECTION FAILED, CHECK CABLE

CONNECTION“ a dále nebylo možné s modulem navázat spojení bez nutnosti restartu aplikace. Stejná chyba se vyskytla i při situaci, kdy došlo během zpracování nebo výpisu zpracovaných dat ke ztrátě komunikace. Tato situace byla pouze simulovaná v rámci testování. Během zátěžového testu aplikace nedošlo naštěstí ke ztrátě komunikace ani jednou. Tento problém byl vyřešen vytvořením vlastní metody *connectToPlatform()* pro připojení k modulu. Nová metoda stále využívá původní metodu *connect* z dodávaného SDK. Při kombinaci vlastní metody s cyklem *while* je nyní možné zajistit opětovné spojení s aplikací bez nutnosti restartu.

Další nedostatek byl odhalen při zpracování dat. Modul, umístěný na dřevěné desce zhruba 10cm nad zemí hlásí jako překážku i nohy postiženého. Řešení problému proběhlo přidáním proměnné *rightHandMode* a modifikací metody *validLaserPoints()*. Při spuštění aplikace je uživatel dotázán na režim v kterém aplikace poběží. Podle toho, z které strany modulu chce uživatel být při používání, zvolí na začátku režim používání aplikace. Dále musel být ze stejného důvodu zablokován interval úhlů, kde se nachází tyč pro držení. Naštěstí jsou tyto intervaly v zadní části systému, tudíž by neměly ovlivnit bezpečnost při používání tím, že by se v zablokovaných intervalech nacházela překážka. Interval zablokovaný pro tyč sloužící k pohybu plynule navazuje na blokaci pro nohy nevidomého. V režimu pro pravou ruku měří tedy systém v intervalu -165° až +100°. Pro režim levé ruky je použit interval -100° až +165°.

Jednou z největších komplikací byla nedokonalá detekce překážek o šířce 2cm ve vzdálenostech 30 a 40cm zmíněné v kapitole testování. Pro řešení problému jsem si úpravou aplikace vypsal paprsky včetně jejich parametrů, ale bez veškerého filtrování. Příklad vypsaných dat lze vidět na obrázku níže.



Obrázek 27: Ukázka nefiltrovaných dat

(Zdroj: vlastní tvorba)

Z přiloženého obrázku vidíme, že modul z nepochopitelných důvodů nahlásil paprsky detekující překážku na vzdálenostech 30,5 a 31cm s parametrem „valid = 0“. Stejný problém se objevil u řady dalších pokusů o detekci překážky na vzdálenosti 40cm. Jak můžeme vidět u dalších paprsků vzorových dat, pokud byl paprsek vadný, byla mu zároveň přiřazena nesmyslná hodnota vzdálenosti 100 000m. U mnou řešeného paprsku modul vzdálenost bezchybně změřil, pouze parametru „valid“ přiřadil hodnotu 0. Bohužel nejsou k dispozici kritéria, podle kterých se modul rozhoduje při řešení validity paprsku. Pro eliminování problému by bylo nutné vyřadit filtrování paprsků podle parametru validity. Paprsky je možné nadále filtrovat podle vzdálenosti, kde se případně vadné paprsky odhalí pomocí nesmyslné hodnoty vzdálenosti 100 000m, kterou modul těmto paprskům přiřazuje. Při takto zvoleném řešení není možné zaručit, že se do filtrovaných paprsků nedostanou vadné paprsky z hlediska validity. Protože není známo, jak se validita vyhodnocuje, není možné tento problém odstranit a nezbývá jiná možnost, než ho klasifikovat, jako problém na straně vydavatele SDK pro konkrétní modul. Pokud tedy zanechám původní filtrování dvěma způsoby a na straně vydavatele dojde k opravě, zařízení bude po aktualizaci SDK plně funkční.

4 Závěr

Cílem práce byl návrh systému pro orientaci v objektech. Součástí tohoto systému mělo být vhodné rozhraní pro následnou komunikaci s přídavnými systémy typu řečových procesorů apod. Jako poslední cíl této práce bylo otestování systému a stanovení mezí použitelnosti.

Dle zadání měl být celý systém postaven na mapovacím modulu SLAMTEC Mapper M1M1. Po nastudování oficiální dokumentace, kde se příliš mnoho užitečných dat nenacházelo, došla řada také na oficiální podporu produktu. Po několika e-mailech překládaných z čínštiny jsem nabyl dojmu, že plánovaná cesta přes mikropočítače nebude možná a bude nutné zvolit klasický počítač.

Pro funkční systém sloužící k orientaci v interiéru bylo nutné naprogramování vlastní nadřazené aplikace, která data z modulu zpracovává. Tato aplikace vznikala postupným rozšiřováním jednoduchých úkonů s naměřenými daty. Od počátečního získávání dat, přes jejich filtrování, až ke konečnému zpracování pro přídavné systémy. Nejprve jsem diagnostikoval data jedné otočky modulu, dále pak již několik tisíc měření za vteřinu. Aplikaci bylo nutné neustále zdokonalovat. V momentě, kdy jsem byl schopen rozeznat podle měřených paprsků jednotlivé překážky, nastaly další důležité úkony. Bylo třeba zohlednit pohyb nevidomého současně s měřením nových dat. Toho bylo docíleno trvalým během aplikace a měřením v cyklech s intervalem 5s.

Neméně složitá byla i kompletace celého systému. Vzhledem k okolnostem, že veškeré úkony spojené s kompletací byly realizovány v bytě panelového domu bez využití sklepa či dílny, jsem s výslednou podobou systému spokojen.

Při pohledu na výsledky testování modulu musím konstatovat, že mě velice mile překvapil a naměřené hodnoty jsou velmi přesné. Jako slabý článek modulu bych hodnotil mizernou detekci úzké překážky ve vzdálenostech 30 a 40cm, kterou způsobila chyba na straně vývojářů SDK pro daný modul. Tato chyba bohužel nemohla být z mé strany odstraněna, aniž by bylo zasaženo do filtrování paprsků z hlediska validity, což by mohlo mít vliv na funkčnost systému, protože není známa definice, podle které se paprsky validují. Při testování systému jako celku se tato slabina modulu neprojevila, ale považuji za důležité na ni upozornit.

Testování systému jako celku se bohužel velmi špatně hodnotí. Cílem této práce byla pouze detekce překážek a předání těchto dat přídavným systémům. Pokusil jsem se alespoň

pohybovat po místnosti podle naměřených dat, které aplikace vypisuje pro lepší znázornění funkčnosti, a musím konstatovat, že naměřená data z modulu jsou velmi přesná. Dovolím si tvrdit, že za předpokladu vhodného zprostředkování dat nevidomému uživateli, by měl být mnou vytvořený systém funkční pro orientaci v interiéru.

Výstupem práce je základ pro funkční systém detekce překážek v objektu, který data o okolí předává pomocí zápisu do souboru ve formátu CSV pro další přídavné systémy. Měřící rozsah zařízení je programově omezen na 1m, snadno může být změnou parametru v rámci aplikace upraven. Při testování systému jsem neshledal žádné závažné problémy, které by měli mít vliv na bezpečnost uživatele. Cíle práce tedy tímto považuji za splněné.

Návrh na zlepšení bych viděl v použití menšího počítače, případně využití mobilního zařízení a použití jiné platformy pro zpracovávání naměřených dat. Tímto krokem by se zařízení mohlo značně zmenšit. Dále bych u komunikace přistoupil k použití technologie WiFi pro komunikaci s modulem, čím by byla zajištěna jednodušší manipulace. Bylo by ovšem nutné otestovat stabilitu připojení. Tento krok nemohl být realizován pravděpodobně díky problému na síťové kartě použitého notebooku. Mohlo by také dojít ke zmenšení velikosti dřevěné desky, na které byl samotný modul umístěn. V mém případě nemohla být deska zmenšena díky asistenční podložce, která sloužila při testování pro snadnou kontrolu lokace překážky. Vzhledem ke konstrukci samotného modulu, který nedisponuje žádným krytem, by bylo dobré zapracovat na ochraně měřícího modulu. Při tomto kroku je ale nutné myslet na měřící okno pro modul, bez kterého není systém schopen zpracovávat informace.

I. Seznam použitých zdrojů

[1] BUBENÍČKOVÁ, Hana, Petr KARÁSEK a Radek PAVLÍČEK. *Kompenzační pomůcky pro uživatele se zrakovým postižením*. Brno: TyfloCentrum Brno, 2012. ISBN 978-80-260-1538-3.

[2] PILMANNOVÁ, Lenka. *Systém pro navigaci nevidomých uvnitř budov*. Praha, 2007. Diplomová práce. ČVUT. Vedoucí práce Ing. Zdeněk Míkovec.

[3] UKO, Tomáš. *GPS navigace na FPGA*. Praha, 2008. Diplomová práce. ČVUT. Vedoucí práce Ing. Martin Daněk, Ph.D.

[4] Navigační systémy pro nevidomé. *TyfloCentrum Brno, o.p.s.* [online]. 2002-2020 [cit. 2020-02-16]. Dostupné z: http://bariery.centrumpronevidome.cz/bariery/navigint.htm

[5] *Navigační centrum SONS ČR* [online]. 2008 [cit. 2020-02-19]. Dostupné z: http://navigace.sons.cz/

[6] BENEŠ, Pavel. *Automatizace a automatizační technika*. Vyd. 2. Brno: CP Books, 2005. ISBN 80-251-0795-7.

[7] *Elektronika zmírňuje zrakový handicap* [online]. [cit. 2020-02-25]. Dostupné z: http://www.pronevidome.cz

[8] Chytrý náramek usnadňuje cestování nevidomých. *Helpnet.cz* [online]. 2020 [cit. 2020-02-21]. Dostupné z: https://www.helpnet.cz/aktualne/chytry-naramek-usnadnuje-cestovani-nevidomych

[9] RFID Selection Guide. *EBV Elektronik* [online]. [cit. 2020-02-24]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/rfid+guide.pdf

[10] Navigační bílá hůl. *TyfloCentrum Brno, o.p.s.* [online]. 2020 [cit. 2020-02-24]. Dostupné z: http://bariery.centrumpronevidome.cz/bariery/rfid.htm

[11] Diskrétní informační systém. *TyfloCentrum Brno, o.p.s.* [online]. 2020 [cit. 2020-02-26]. Dostupné z: http://bariery.centrumpronevidome.cz/bariery/dinasys.htm

[12] Vědci z Fakulty elektrotechnické ČVUT v Praze představili unikátní řešení hole pro nevidomé. In: w*ww.odbornecasopisy.cz* [online]. FCC PUBLIC, 27.11.2015 [cit. 2020-02-15]. Dostupné z: http://www.odbornecasopisy.cz/clanek/vedci-z-fakulty-elektrotechnicke-cvut-v-praze-predstavili-unikatni-reseni-hole-pro-nevidome--1286

[13] *Detekce objektů v blízkosti vozidla pomocí LiDARu*. Praha, 2017. Diplomová práce. České Vysoké Učení Technické v Praze Fakulta dopravní. Vedoucí práce Ing. Václav Jirovský, Ph.D.

[14] KACHTÍK, Lukáš. Princip laseru. *Laser a vše o něm* [online]. 2010 [cit. 2020-03-02]. Dostupné z: http://lasery.wz.cz/princip.html

[15] MADA, Martin. *Mapování 3D prostoru pomocí LIDARových dat*. Ostrava, 2015. Bakalářská práce. VŠB. Vedoucí práce Ing. Branislav Holý.

[16] *SLAMTEC Mapper - Introduction and Datasheet(PREVIEW)* [online]. Shanghai Slamtec Co., 2019 [cit. 2020-03-5]. Dostupné z: http://bucket.download.slamtec.com/975678f44875b6cb4db5ecf30202163c8910519b/SLAMTEC_mapper_datasheet_M1M1_v1.2_en.pdf

[17] C++ Knihovna kontejnerů. *Cppreference.com* [online]. 2014 [cit. 2020-03-15]. Dostupné z: https://cs.cppreference.com/w/cpp/container/vector

[18] Základní principy a pojmy OOP. *Péhápko.cz* [online]. [cit. 2020-03-17]. Dostupné z: http://pehapko.cz/oop/uvod

II. Seznam obrázků a grafů

III. Seznam tabulek

IV. Přílohy

Na přiloženém CD se nachází kompletní aplikace včetně zdrojových kódů a souborů potřebných ke spuštění. Dále zde najdeme soubor s výsledky testování modulu.