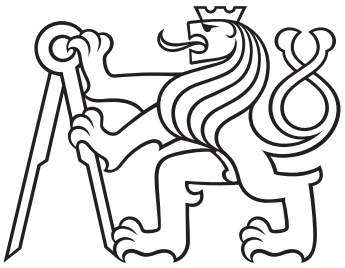
České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra měření

Studijní program: Kybernetika a robotika



Testování robotické jednotky pro měření osvětlenosti v interiéru

BAKALÁŘSKÁ PRÁCE

Vypracovala: Alena Viktorová

Vedoucí práce: Ing. Tomáš Drábek

2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Viktorová** Jméno: **Alena** Osobní číslo: **483784**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Testování robotické jednotky pro měření osvětlenosti v interiéru

Název bakalářské práce anglicky:

Testing of a robotic unit for measuring indoor illuminance

Pokyny pro vypracování:

Seznamte se s metodikou měření osvětlenosti v interiéru.
Seznamte se s robotickou jednotkou a jejím softwarovým vybavením.
Implementujte výpočet kontrolních bodů pro měření osvětlenosti v interiéru na základě dodaných referencí.
Implementujte komunikaci mezi lokalizací robotické jednotky v prostoru a algoritmem pro určování kontrolních bodů.
Otestujte navržený program v reálných podmínkách a upravte případně software.
Vytvořte návod pro obsluhu robotické jednotky.

Seznam doporučené literatury:

[1] ČSN EN 13032-1+A1 Světlo a osvětlení – Měření a uvádění fotometrických údajů světelných zdrojů a svítidel – Část 1: Měření a formát souboru údajů
[2] ČSN EN 12464-1 Světlo a osvětlení - Osvětlení pracovních prostorů - Část 1: Vnitřní pracovní prostory
[3] ČSN 36 0011-1 Měření osvětlení prostorů - Část 1: Základní ustanovení
[4] ČSN 36 0011-3 Měření osvětlení prostorů - Část 3: Měření umělého osvětlení vnitřních prostorů

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Drábek, katedra měření FEL

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

Datum zadání bakalářské práce: **21.01.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce letního semestru 2021/2022

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

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna 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.

5. 5. 2021
Datum převzetí zadání

Podpis studentky

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracovala samostatně a použila jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ....

Alena Viktorová

Poděkování

Děkuji Ing. Tomáši Drábkovi za vedení mé bakalářské práce a za podnětné návrhy, které ji obohatily. Mé poděkováni patři též Ing. Vladimíru Petríkovi, Ph.D. za spolupráci při implementaci softwarové části.

Alena Viktorová

Název práce:

Testování robotické jednotky pro měření osvětlenosti v interiéru

Autor: Alena Viktorová

Studijní program: Kybernetika a robotika

Obor: Kybernetika a robotika

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Tomáš Drábek
Katedra měření

Abstrakt: Tato bakalářská práce řeší nový způsob výpočtu kontrolních bodů pro měření umělého osvětlení v interiéru (viz Úvod). Cílem práce bylo implementovat software, který bude určovat kontrolní body na základě naměřených hodnot osvětlenosti. Výpočet je založen na Gaussovském procesu. Software je otestován na místnostech se simulovanými poruchami svítidel (kapitola 3.1).
Hardwarové vybavení robotické jednotky je popsané v kapitole 1. Po propojení softwaru s robotickou jednotkou byl proces měření osvětlenosti otestován i v reálných podmínkách (kapitola 3.2). Reálným testováním se dospělo k několika návrhům na zlepšení či rozšíření hardwarového i softwarového vybavení robotické jednotky (viz Závěr). Na konci práce je sepsán návod pro obsluhu celé kontroly měřené místnosti (kapitola 4).

Klíčová slova: Gaussovský proces, robotická jednotka, osvětlenost, automatizace

Title:

Testing of a robotic unit for measuring indoor illuminance

Author: Alena Viktorová

Abstract: This bachelor thesis deals with a new way of calculating control points for measuring artificial lighting in the interior (see Introduction). The thesis aimed to implement software that would determine control points based on measured values of illuminance. The calculation is based on the Gaussian process. The software is tested in rooms with simulated luminary faults (chapter 3.1).

The hardware equipment of the robotic unit is described in chapter 1. After connecting the software with the robotic unit, the process of measuring the illumination was tested even in real conditions (chapter 3.2). Through real testing, several suggestions were made to improve or expand the hardware and software equipment of the robotic unit (see Conclusion). There are written instructions for operating the entire control of the measured room at the end of the thesis (chapter 4).

Key words: Gaussian process, robotic unit, illuminance, automation

Obsah

Seznam použitých zkratek

KB Kontrolní body

RJ Robotická jednotka

GP Gaussovský proces

Seznam obrázků

[1] Kontrolní body

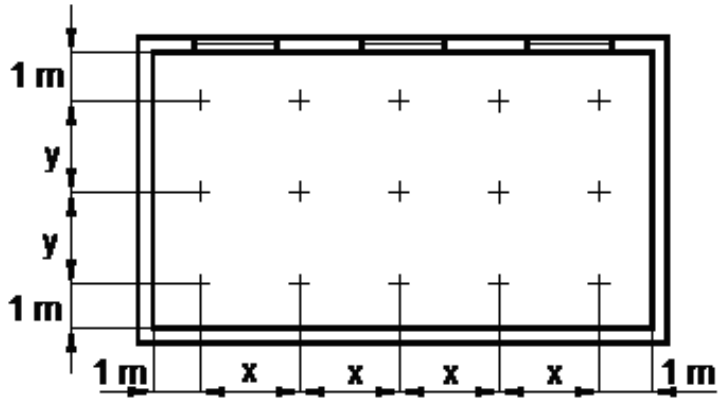
[2] Robotická jednotka

Úvod

Vnitřní umělé osvětlení je nedílnou součástí našeho každodenního života. Používáme jej po setmění pro prodloužení dne a nebo i přes den při nedostatečném denním osvětlení. Při použití nevhodného umělého osvětlení může docházet ke snížení produktivity, bolesti hlavy nebo dokonce poškození zraku. Proto je potřeba zkontrolovat celý prostor, zda je světelná soustava dostatečná a splňuje podmínky, které jsou vypsány v normě [1], pro konkrétní využití. Samotná kontrola probíhá obvykle při stavbě nové budovy či při rekonstrukci světelné soustavy (např. výměna typu žárovek).

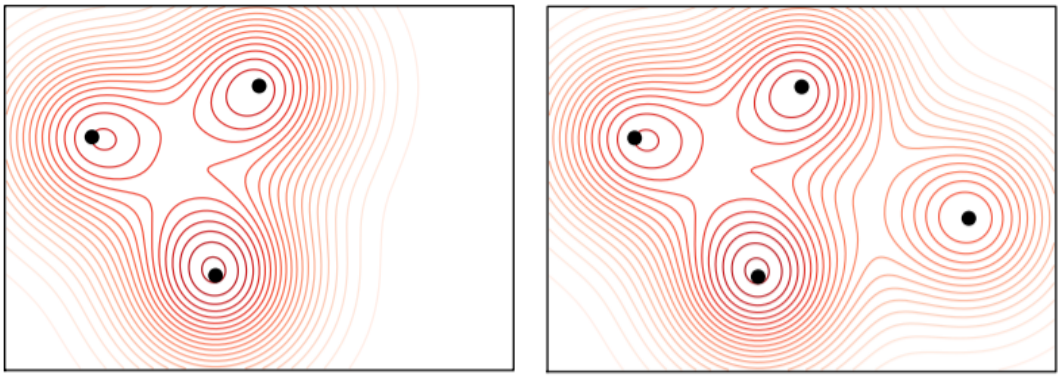
Důvody, proč je měření osvětlenosti tak důležité, jsou vypsané také v normě [2]. Jeden z nich je například již zmiňovaná zraková pohoda během užívání osvětlovacích soustav. Dvě výše jmenované normy se od sebe odlišují v měřeném prostoru. Norma [2] uvádí, že kontrolní body (dále jen KB) jsou umístěny ve výšce 0,85 m a ve vzdálenosti 1 m od zdi (příp. pevné překážky), nicméně norma [1] počítá se vzdáleností 0,5 m. Tato menší vzdálenost se používá v praxi více, jelikož měřený prostor je větší a tím je kontrola přesnější.

Stanovení polohy KB vychází z tvaru a rozměrů místnosti, které jsou vyčteny např. ze stavebních výkresů. KB jsou rozmístěny rovnoměrně do pravoúhlé obdélníkové sítě (viz obrázek 1) ve vzdálenostech, jež jsou určeny dle rozměrů kontrolovaného prostoru. Poté, co obluha určí KB, musí každý bod projít a nechat změřit osvětlenost pomocí luxmetru. Tedy proces samotného měření je v dnešní době téměř celý závislý na lidské obsluze. Jelikož je proces velmi zdlouhavý, zrodila se myšlenka, jak jej co nejvíce automatizovat.



Obrázek 1: Rozložení KB v obdélníkové pravoúhlé síti. Převzato z normy [2].

Ve své práci budu rozšiřovat softwarové vybavení pro již vytvořenou robotickou jednotku [3]. Konkrétně se budu zabývat implementací stanovení polohy KB, budu při tom vycházet z již dříve publikovaného algoritmu, který je popsaný v odborném článku [4]. Výpočet KB již nebude počítat s rozložením v pravoúhlé síti, bude totiž založen na simulaci šíření osvětlenosti dle typu světelné soustavy (viz obrázek 2). Nicméně měřicí oblast bude stále stanovena dle výše uvedené normy [1], tedy 0,5 m od zdi. Dále budu implementovat komunikaci mezi lokalizací robotické jednotky a mým algoritmem pro výpočet KB. Pomocí spojení těchto algoritmů bude zautomatizován téměř celý proces kontroly osvětlenosti.



Obrázek 2: Určení KB na základě simulace šíření osvětlenosti. Černé body znázorňují KB, červené vrstevnice pak zobrazují míru osvětlenosti (čím světlejší červená, tím menší osvětlenost). Černé ohraničení značí půdorys místnosti. Obrázek slouží pouze pro názornou představu o rozložení KB, nejedná se o reálnou simulaci.

V poslední části se budu věnovat samotnému testování softwarového vybavení v reálných podmínkách. Tedy použiji místnost s umělým osvětlením bez nábytku

a otestuji celý proces měření osvětlenosti v interiéru. Budu testovat především správnou komunikaci mezi lokalizací robotické jednotky a algoritmem pro stanovení KB. V případě potřeby software upravím. Aby robotická jednotka spolu s algoritmy mohla být obsluhována téměř kýmkoliv, bude přiložen k této práci návod na jejich obsluhu.

1. Robotická jednotka

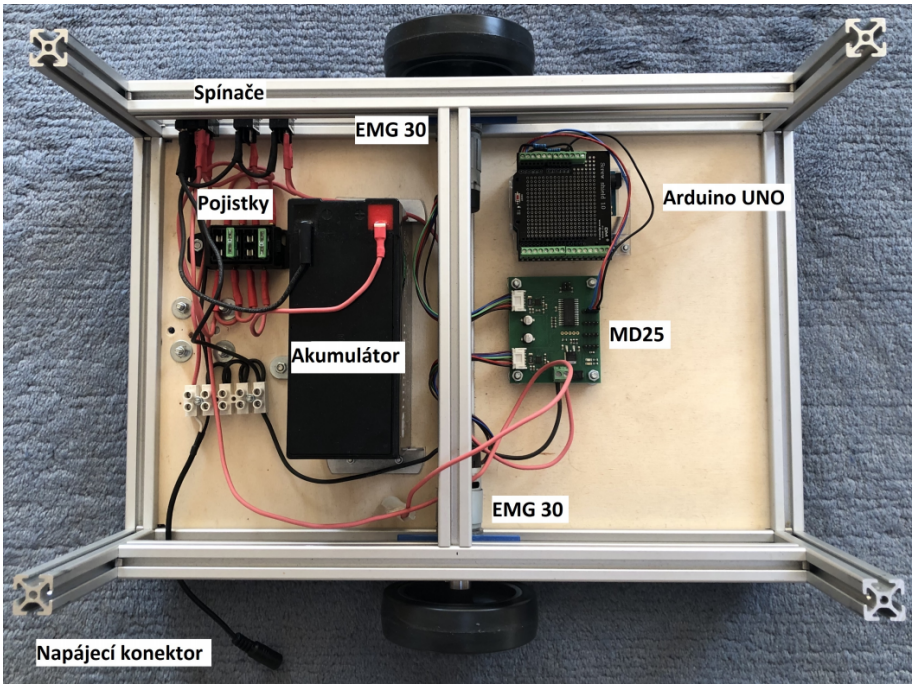
V této kapitole se budu zabývat hardwarovým vybavením robotické jednotky (dále jen RJ), která vychází z diplomové práce pana Drábka [5] a je upravena v rámci bakalářské práce pana Kůrky [3]. Jelikož nebyla součástí mé práce žádná úprava RJ, budu především čerpat z dvou výše uvedených prací. Celá RJ včetně řídicího počítače je zobrazena na obrázku 1.1.

Obrázek 1.1: Robotická jednotka s řídicím počítačem

1.1 Konstrukce

Základní konstrukcí RJ je obdélníkový hliníkový rám, který zajišťuje bytelnost při nárazu a snadnou manipulaci. V rámu jsou pak zasazeny jednotlivé součástky (viz obrázek 1.2), o nichž se zmíním v následujících podkapitolách.

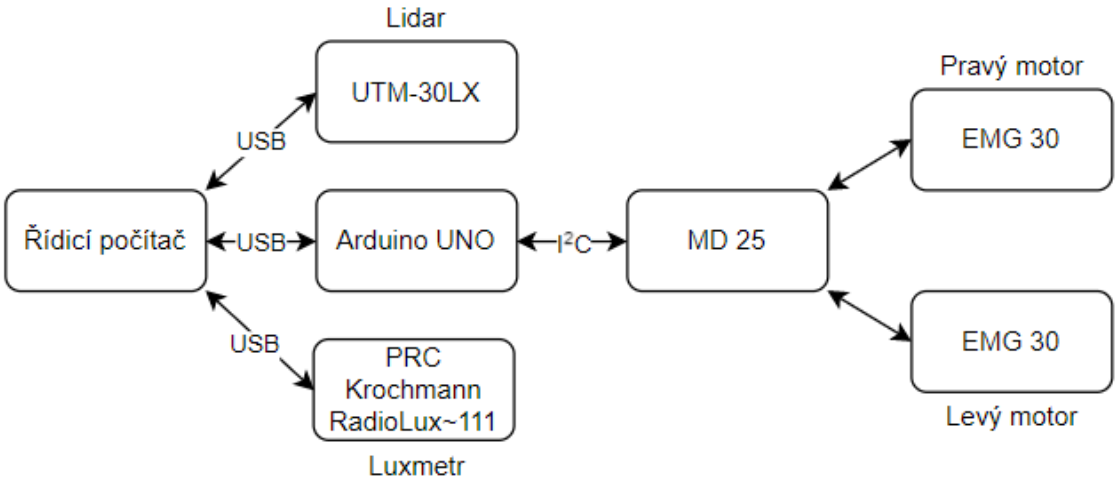
Na RJ je připevněn nástavec (viz obrázek 1.1), na němž je umístěn senzor pro měření osvětlenosti (blíže v kapitole 1.4.2). Senzor je ve výšce 0,85 m, což odpovídá normě [1] pro kontrolu osvětlenosti v interiéru.



Obrázek 1.2: Rozložení součástek v robotické jednotce [3]

1.2 Řídicí počítač

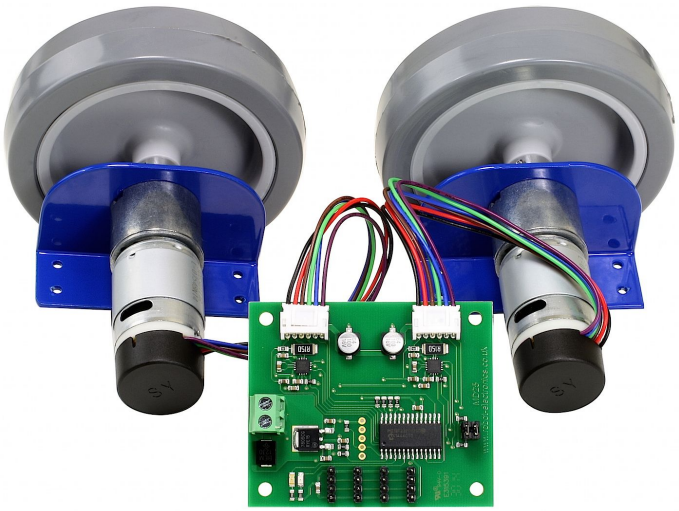
Na RJ je umístěn řídicí počítač Lenovo Yoga 500, který je nezbytnou součástí celého měření. Počítač má procesor Intel Core i5–6200U a operační paměť o velikosti 4 GB. Do počítače je pomocí USB připojen lidar, luxmetr a Arduino UNO. Prostřednictvím Arduino UNO probíhá komunikace s motory, kterým se nastavuje příslušná rychlost. Průběh celé komunikace mezi zmíněnými součástmi RJ je zobrazena na obrázku 1.3.



Obrázek 1.3: Komunikace mezi hardwarovým vybavením RJ

1.3 Pohyb jednotky

RJ se dokáže podle příkazů pohybovat. Pro rozpohybování je použita řídící jednotka motorů MD 25 [6], která je napojena na dva motory EMG 30 [7] s převodovkou a enkodérem. Ten je tvořen dvěma Hallovými sondami a slouží pro určení momentálního natočení hřídele motoru a také je schopen určit směr otáčení. Jeho rozlišení je 360 pulsů na otáčku hřídele. Každý z motorů ovládá jedno kolo (každé o průměru 100 mm). Pro zaručení stability je v zadní části RJ umístěno ještě jedno kolo sloužící jako opěrný bod. Součástí jsou také dva montážní úhelníky. Celý systém pohonu jednotky, který je složen z výše uvedených částí, se nazývá RD 01 [8] (viz obrázek 1.4).



Obrázek 1.4: Kompletní systém pohonu RD01 [8]

1.4 Senzory dle využití

1.4.1 Určení půdorysu místnosti

Nejprve se půdorys místnosti určoval dle nárazů do zdi. Pro tento způsob byly na RJ umístěny senzory nárazu. Nicméně nyní je pro určení mapy místnosti použit lidar UTM-30LX [9] (viz obrázek 1.5). Ten měří vzdálenosti od zdí (případně překážek) pomocí odrazu laseru a později z těchto vzdáleností díky softwaru od pana Kůrky vykreslí půdorys místnosti.

Obrázek 1.5: Dálkoměr UTM-30LX [9]

1.4.2 Měření osvětlenosti

Pro samotné měření osvětlenosti je použit luxmetr PRC Krochmann Radio-Lux 111 (viz obrázek 1.6). Slouží pro fotometrické a radiometrické měření a může být použit jak ve vnitřním prostoru tak i ve venkovním [10]. Luxmetr je připojen k počítači, který má v sobě nahraný potřebný software pro zpracování hodnot z displeje. Na obrázku 1.6 je vidět, že při měření se vypisuje aktuální teplota a hodnota osvětlenosti. Pro účely měření se zaznamenává pouze hodnota osvětlenosti. Řídicí počítač je umístěn na RJ a tedy pohybuje se spolu s jednotkou.

Obrázek 1.6: Luxmetr PRC Krochmann RadioLux 111 – vpravo je senzor osvětlenosti, uprostřed je redukce vedoucí do USB, která se připojuje do řídicího počítače, vpravo nahoře je zvětšený náhled displeje luxmetru během měření

1.5 Napájení

Celou RJ napájí olověný akumulátor Shimastu NPG8-12, 12 V s kapacitou 8 Ah. Tento akumulátor dokáže regulovat velikost proudu a tím i dobu vybíjení [11]. Arduino UNO, které slouží pro zpracování příkazů z počítače, je napájeno pomocí USB palubním počítačem.

2. Software

Jedním z hlavních cílů mé bakalářské práce, který je popsán v první části této kapitoly, byla implementace výpočtu KB pro měření osvětlenosti v interiéru na základě dodaných referencí. Nejprve přiblížím danou problematiku a poté rozeberu konkrétní kroky, které byly nedílnou součásti při procesu implementace. Na závěr zmíním několik problémů, s nimiž jsem se v průběhu setkala.

Dalším z hlavních cílů byla implementace komunikace mezi lokalizací robotické jednotky v prostoru a algoritmem pro určování KB. Zde budu navazovat opět na práci pana Kůrky [3], který se zabýval softwarovým vybavením RJ a v rámci své práce již implementoval najíždění RJ do zadaných bodů v místnosti.

2.1 Výpočet kontrolních bodů

Algoritmus pro výpočet KB vychází z již publikovaného odborného článku [4]. Bylo tedy podle něj v určitých krocích postupováno. Celý výpočet KB je založen na Gaussovském procesu (dále jen GP[1]).

2.1.1 Gaussovský proces v 1D prostoru

GP aproximuje neznámou funkci na základě naučených parametrů z nasbíraných dat v průběhu pozorování. Je definován pomocí funkce mean m(x) (tzv. střední hodnotou) a funkcí covariance k(x) (tzv. rozptylem) [12], zjednodušeně matematicky zapsané jako

$$f \sim GP(m, k). \tag{2.1}$$

V již zmiňovaném odborném článku [4] je použito normální rozdělení, tedy

$$f \sim N(\mu, \sigma^2), \tag{2.2}$$

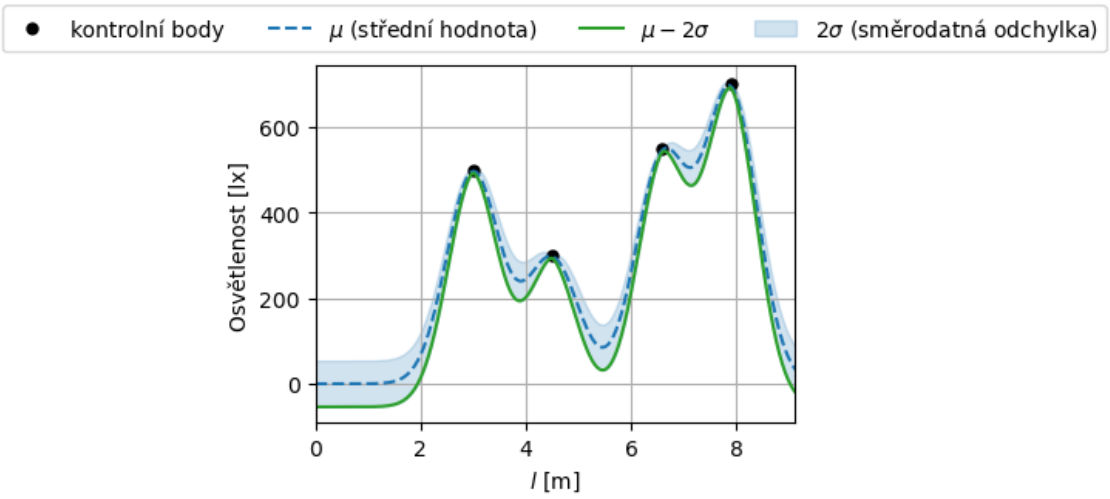
[1] Gaussovský proces

kde μ je střední hodnota a σ je směrodatná odchylka. Pro dosažení tohoto rozdělení byl zvolen model ExactGPModel [13] s pravděpodobností FixedNoiseGaussianLikelihood [14], která má nastavený fixní šum (noise) na σ^2. Pomocí modelu s těmito parametry se spočítá dolní mez (lower bound) jako

$$f_{lb} = \mu - 2\sigma. \tag{2.3}$$

Vizualizace střední hodnoty, směrodatné odchylky a dolní mezi se nachází na obrázku 2.1.

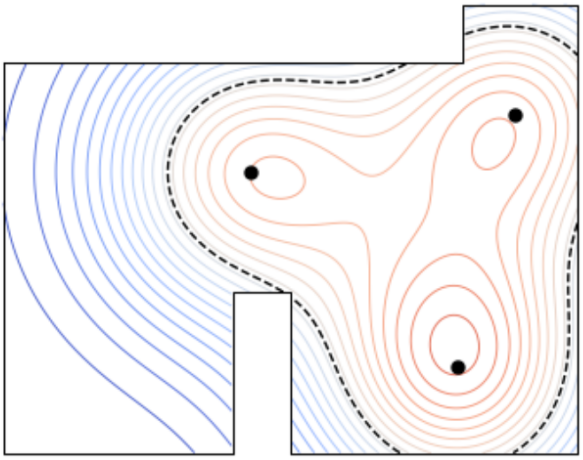
Pro seznámení s modelem GP z knihovny GPyTorch [13] jsem použila zjednodušenou variantu a to v 1D prostoru. Zjednodušení spočívalo v tom, že byly známy parametry modelu, které se tak nemusely získávat pomocí trénování modelu. Dále KB byly zadány pouze skalárními hodnotami, a tak se ještě nejednalo o reálné měření, kdy je KB určen souřadnicí (tedy dvojicí čísel) v daném půdorysu místnosti. Pro názornost jsem vykreslila obrázek 2.1.



Obrázek 2.1: Gaussovský proces v 1D prostoru pro předem dané parametry

2.1.2 Gaussovský proces ve 2D prostoru

Při přechodu z GP v 1D prostoru do 2D prostoru se jednalo stále o zjednodušenou variantu, kde ještě neprobíhalo samotné trénování modelu. Parametry modelu byly definovány manuálně. Opět byl zadán polygon místnosti, souřadnice KB a hodnota osvětlenosti v daných bodech. Pro lepší představu jsem nechala vykreslit šíření osvětlenosti, konkrétně střední hodnotu, viz obrázek 2.2.



Obrázek 2.2: Gaussovský proces ve 2D prostoru pro předem dané parametry. Černé ohraničení polygonu značí půdorys místnosti, černé tečky jsou KB, vrstevnice znázorňují šíření osvětlenosti – červená barva hodnoty 500-1000 lx, modrá barva 0-500 lx, černá přerušovaná křivka označuje, kde hodnota osvětlenosti překračuje mez 500 lx. Značení je stejné pro všechny obrázky v této práci vykreslující osvětlenost.

2.1.3 Zpracování nasimulovaných dat

Simulace šíření osvětlenosti byla provedena pomocí softwaru pro světelný design DIALux [15]. Byl vyexportován soubor typu .csv, jenž obsahoval v prvním sloupečku souřadnice y, v posledním řádku souřadnice x a dále hodnotu osvětlenosti v daných souřadnicích (x, y). K načítání souboru byla použita knihovna pandas [16]. Nejprve se nasimulovala obdélníková místnost, kde zpracování dat proběhlo bez komplikací. Poté jsem zkusila pouze vyměnit polygon (půdorys místnosti) za členitější současně se vstupním souborem s daty, nicméně zjistila jsem, že bude potřeba ještě software upravit.

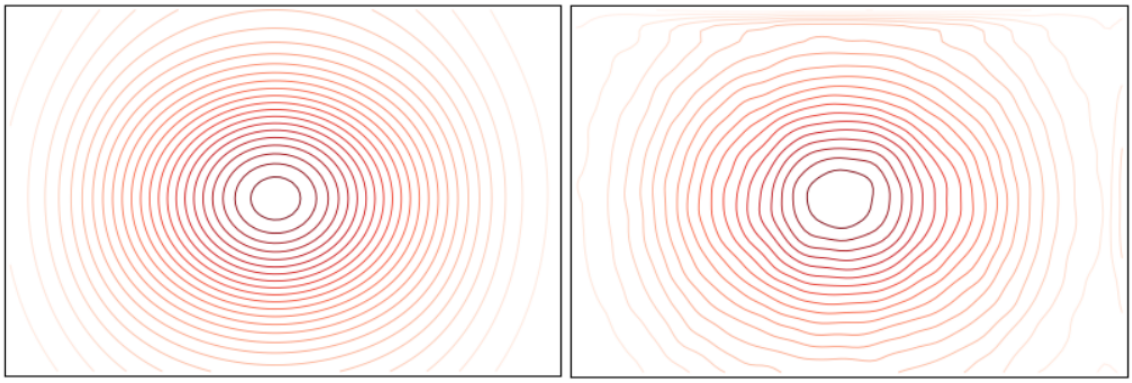
Jelikož se jednalo o složitější půdorys místnosti a nikoliv obdélníkový, byla v několika souřadnicích zaznamenána překážka (nebo zeď), ta byla v souboru označena zpětným lomítkem. Zpětné lomítko jsem nahradila hodnotou NaN, abych mohla pomocí interpolace získat hodnotu osvětlenosti v zadaném bodě. Zmíněné překážky nejspíše ještě způsobily, že některé číselné hodnoty osvětlenosti byly typu string, tedy ty se musely přetypovat na proměnnou typu float, aby se s nimi mohlo dále jednoduše pracovat.

2.1.4 Trénování parametrů modelu

Na trénování modelu se použila nasimulovaná černá obdélníková místnost o velikosti 5 m × 7,5 m, ve které byla jedno svítidlo stejného typu, jako bude použito u reálného měření. Dále byla použita část kódu podle návodu na natrénování parametrů modelu [13]. Do modelu se dala vstupní trénovací data, která jsem obdržela z výše zmíněné simulace šíření osvětlenosti.

Nejprve jsem spustila trénování bez předem nastavených parametrů. Tím jsem po 100 iteracích dostala hodnotu ztrátové funkce 3,278. Jelikož jsem chtěla dosáhnout hodnoty kolem nuly, přednastavila jsem před samotným trénováním parametry modelu, aby ztrátová funkce začínala v první iteraci na nižší hodnotě. Pro názornost jsem vykreslila pro oba případy šíření osvětlenosti po natrénování modelu (viz obrázek 2.3).

Přednastavené hodnoty způsobily, že hodnota ztrátové funkce kolísala, na obrázku 2.3b je i vidět, že šíření osvětlenosti nebylo pravidelné. I přes méně pravidelné šíření osvětlenosti a kolísající ztrátovou funkci byla nakonec zvolena varianta s předem zadanými parametry, jelikož více odpovídala realitě.



(a) Bez přednastavených parametrů **(b)** S přednastavenými parametry

Obrázek 2.3: Simulace šíření osvětlenosti na trénovacím svítidle. Černé ohraničení značí půdorys místnosti, červené vrstevnice znázorňují hodnotu osvětlenosti (čím tmavší červená, tím vyšší osvětlenost).

2.1.5 Určení kontrolních bodů

Stanovení KB je rozděleno do dvou částí. Nejprve jsou body vybírány z hranice vnitřního polygonu (tedy 0,5 m od zdi či překážky). Ve chvíli, kdy již bude měřením

pokrytá celá hranice, tak se pro výběr KB zvolí množina obsahující body z vnitřní části polygonu (tj. body vzdálené od nejbližší zdi či překážky minimálně 0,5 m). Měření končí úspěšně, když je průměrná hodnota spodní hranice vyšší než zvolená mez (v tomto případě je mez 500 lx), nebo končí neúspěšně, pokud všechny body splňují maximální povolenou směrodatnou odchylku (zde je zvolena hodnota 10 lx) a zároveň průměrná hodnota spodní hranice je stále pod zvolenou mezí 500 lx.

První KB je zvolen v levém spodním rohu vnitřního polygonu, další jsou již určeny funkcí na základě výpočtu z odborného článku [4]. Při volání funkce se spočítá aktuální rozptyl, směrodatná odchylka (druhá odmocnina z rozptylu) a střední hodnota, z čehož se určí podle rovnice 2.3 spodní hranice. Pro každý bod, jenž nesplňuje požadovanou směrodatnou odchylku, je vypočtena vzdálenost od nejbližšího KB. Bod, nesplňující požadavky na směrodatnou odchylku a spodní hranici, který má největší vzdálenost od nejbližšího KB, je zvolen jako další KB.

První fáze, tedy určování KB z množiny bodů z hranice vnitřního polygonu, končí, když není na výběr žádný bod, který by podmínky nesplňoval. V tu chvíli se přepne určování bodů do druhé fáze, kdy funkce vybírá z množiny obsahující rovnoměrně rozložené body uvnitř polygonu. V druhé fázi se navíc kontroluje průměrná spodní hranice pro celou měřenou plochu. Pokud průměrná spodní hranice je větší než 500 lx, kontrola měření osvětlenosti se může prohlásit za úspěšnou a již se nemusí určovat další KB.

2.1.6 Problémy při implementaci

Jelikož již existoval kód pro určování KB, měl sloužit jako inspirace. Kód se ovšem nepovedlo zcela zprovoznit z důvodu dostupnosti nových verzí knihoven a tím nefunkčnosti starších verzí. Úkolem tedy bylo implementovat algoritmus pomocí nových verzí knihoven.

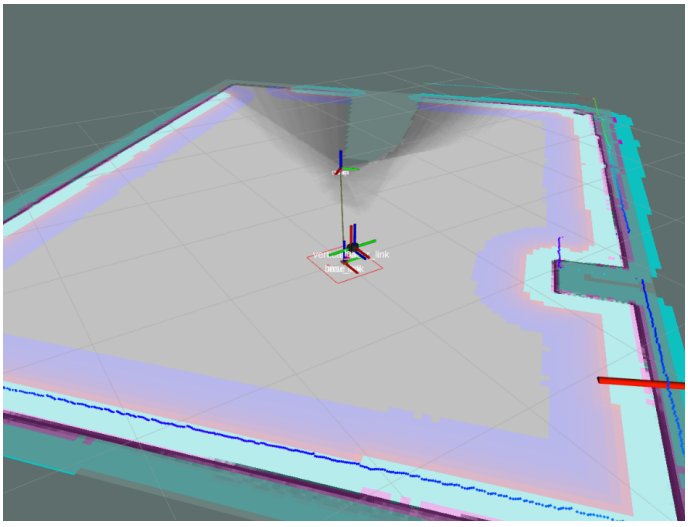
Jednou z nich byla knihovna TensorFlow [17], která přešla z verze 1 do verze 2. TensorFlow 2 ale nebyl kompatibilní se zbytkem programu, a tak byl po domluvě se školitelem nahrazen knihovnou GPyTorch [13]. Celkově jsem tedy přepsala většinu kódu z NumPy [18] do PyTorch [19].

Dalším z problémů byla špatná práce s modelem. Načtený natrénovaný model jsem používala pro výpočet střední hodnoty a rozptylu, což způsobovalo chybné

ukončení programu z důvodu špatných velikostí vstupu do modelu. Po použití kopie natrénovaného modelu se tato chyba vyřešila.

2.2 Lokalizace robotické jednotky v prostoru

Před samotným měřením musí RJ zmapovat daný prostor, pro tyto účely je použit lokalizační algoritmus Cartographer [20]. Ten komunikuje spolu s odometrií a lidarem, ze kterého si přebírá informaci o vzdálenosti od okolních stěn či překážek. Výstupem z Cartographeru je mapa prostoru vizualizovaná pomocí RViz [21] (viz obrázek 2.4), která se dále uloží a zpracuje aproximačním algoritmem, jenž převede mapu na polygon. Dále se již RJ pohybuje pouze ve vnitřní části polygonu. Tento polygon je zmenšen oproti půdorysu místnosti o předem zadaný parametr (tedy zajišťuje vzdálenost od okolních zdí či překážek o zmiňovaný parametr – dle normy [1] je parametr 0,5 m).



Obrázek 2.4: Vizualizace místnosti pomocí RViz – tmavě fialové ohraničení značí hranici místnosti, tedy zdi či překážky, světle modrá označuje hranici, kam už by RJ neměla najet, aby nedošlo k nárazu do zdi, světle fialová značí plochu, kde se RJ může už pouze otáčet, aby nenajela do světle modré plochy

2.3 Komunikace mezi výpočtem kontrolních bodů a lokalizací robotické jednotky v prostoru

Propojení algoritmů na výpočet KB a na najíždění do zadaného bodu mělo být zprvu vyřešeno pouze jednoduchým voláním funkce z jiného souboru. Nicméně zjistilo se, že funkce na najíždění do zadaného bodu, která spadá pod ROS [22], je psaná v Pythonu 2, zatímco výpočet KB je psaný v Pythonu 3.

Prvním nápadem pro vyřešení této kolize bylo sjednotit kódy do jedné verze Pythonu, ale každý z kódů obsahoval nějakou část, která nebyla v druhé verzi podporována. U výpočtu KB se jednalo o knihovnu PyTorch, která požadovala vyšší verzi Pythonu. U lokalizace RJ se jednalo o samotný ROS, jenž fungoval pouze na verzi 2.

Nakonec se tento problém vyřešil pomocí subprocesu. V jednom z kódů, v tomto případě je zvolen kód pro výpočet KB, se zavolá příkaz subprocess.run() [23]. Do závorek je vložen text, kterým by se spouštěl druhý kód pomocí terminálu, následujícím způsobem ["python", "move_to_target.py", "position_x", "position_y"], kde position_x a position_y jsou argumenty souboru move_to_target.py. Pro spuštění celého procesu se pak jen obvyklým způsobem zavolá první soubor pomocí spouštěcího příkazu python3.

3. Testování

Testování je rozděleno do několika částí. První část se zabývá testováním softwaru v průběhu implementace, což bylo provedeno při práci s nasimulovanými daty. Později se postupně přecházelo ze simulovaných dat na data z reálného měření. Například pro hodnoty osvětlenosti se zprvu brala data pouze ze souboru z DIALuxu (viz kapitola 2.1.3), později byla data nahrazena manuálním zadáváním hodnoty, což následně vedlo k plynulejšímu přechodu při zpracovávání dat z luxmetru. Nicméně jednalo se v první části pouze o testování samotného softwaru pro výpočet KB, kdy ještě nebylo propojení s RJ.

V druhé části se již propojil implementovaný software s RJ a začalo se testovat na reálných podmínkách (viz obrázek 3.1). K testování byly využity dvě učebny v prostorách FEL ČVUT v Praze. Vzhledem k tomu, že testování probíhalo přes den a nikoliv v noci, tak se jednalo o měření vnitřního kombinovaného osvětlení, tedy denního i umělého zároveň. Což pro testovací účely bylo dostatečné, nicméně klasické měření umělé osvětlenosti v interiéru by nemělo být ovlivňováno denním osvětlením, a tak by bylo vhodnější ho provádět v noci.

Obrázek 3.1: Testování v reálných podmínkách (místnost 1)

Poslední část této kapitoly zahrnuje finální testy provedené ve dvou místnostech. První místnost má rozměry 7,177 m × 5,413 m a druhá 7,415 m × 5,42 m. Při finálních testech se kladl důraz na přesnost najíždění do koncového bodu.

3.1 Testování softwaru pro výpočet kontrolních bodů – simulace

Software pro výpočet kontrolních bodů byl testován nejdříve na obdélníkové místnosti o velikosti 5 m × 7,5 m a poté se přešlo k členitější místnosti o velikosti 5,52 m × 7,32 m. Při přechodu na členitější místnost se musel software upravit, aby byly správně detekovány všechny zdi či překážky (viz kapitola 2.1.3).

3.1.1 Množina kandidátů na kontrolní body

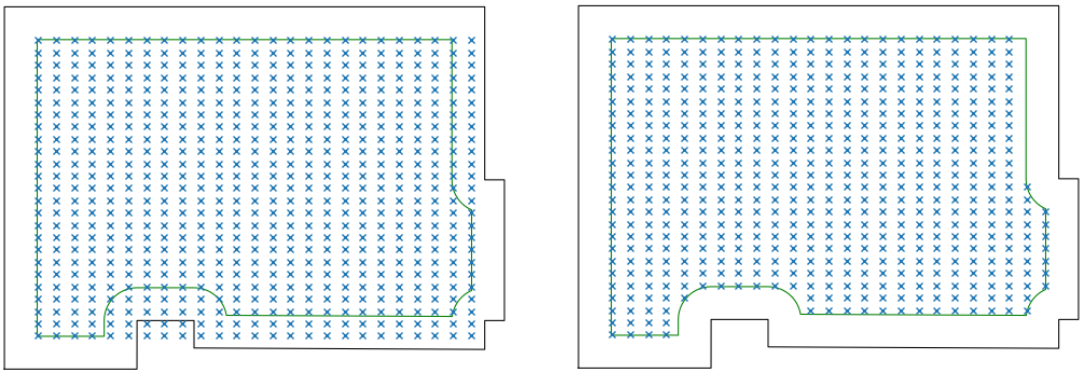
Jelikož je určování KB rozděleno na dvě fáze (viz kapitola 2.1.5), bylo potřeba zjistit, zda se kandidáti na KB vybírají ze správné množiny. Nejprve bylo otestováno, zda tyto dvě množiny obsahují správné body. U první fáze, kdy mělo být vybíráno z bodů na hranici vnitřního polygonu, bylo graficky ověřeno, že funkce funguje správně.

U druhé fáze se zjistilo, že množina, která měla obsahovat rovnoměrně rozložené body z vnitřní části polygonu, obsahovala body navíc (viz obrázek 3.2a). Byla tedy přidána podmínka na ověřování, zda daný bod opravdu patří do množiny bodů z vnitřní části polygonu nebo nikoliv. Po přidání podmínky již byly body určeny správně (viz obrázek 3.2b).

Velikost množiny kandidátů na KB je určena zadaným parametrem, který určuje, na kolik bodů bude rozdělena místnost v jedné ose. V simulované místnosti o velikosti 5,5 m × 7,32 m byl parametr nastaven na 25, tedy celkově je umístěno 625 možných kandidátů. Z nichž jsou ještě nějaké vyřazeny pro nesplnění podmínky, kdy mají být uvnitř měřicí plochy.

Počet těchto bodů určuje, jak dlouho bude algoritmus pro výpočet KB trvat, nicméně je třeba tento počet volit dle rozměrů měřené místnosti. Na zmíněné místnosti byly zkoušeny parametry 50 a 100, každopádně při těchto hodnotách již výpočet trval déle a nelišily se vybírané KB, tak byl zvolen parametr 25. Stejná hodnota

byla použita i pro reálné testování, jelikož měřicí místnosti byly rozměrově podobné.



(a) Chybné určení bodů **(b)** Správné určení bodů

Obrázek 3.2: Množina bodů z vnitřní části polygonu – černé ohraničení značí půdorys místnosti, zelené ohraničení označuje měřenou plochu (tedy vnitřní polygon, který je zmenšen o 0,5 m oproti půdorysu místnosti), modré křížky označují body, které dále v algoritmu mohou být zvoleny jako KB.

3.1.2 Kritéria pro výběr kontrolních bodů

Pro otestování implementovaného softwaru celý proces výpočtu KB nejprve vycházel ze simulovaných dat bez propojení s RJ. Zkoušela se různá kritéria, podle kterých se mohou KB z množiny kandidátů vybírat. První kritérium bylo, že se vybírají body, které nesplňují spodní hranici 500 lx. Nicméně brzy se zjistilo, že toto kritérium fungovat nebude. Změřená data by byla zkreslená, jelikož by se vybíraly jen body, ve kterých je větší pravděpodobnost, že bude osvětlenost vyšší než 500 lx. Výsledné kritérium pro výběr KB je hodnota směrodatné odchylky v daném bodě. Aby byl bod zvolen, tak musí mít větší směrodatnou odchylku než 10 lx.

3.1.3 Počet kontrolních bodů a délka výpočtu

Zjišťovalo se, zda program reaguje dle očekávání na měnění parametru vzdálenosti od zdi a jaký vliv má vzdálenost na počet KB. Při simulaci se zkoušely vzdálenosti 0,5 m a 1 m od zdi či překážek. Závěrem je, že parametr je možné jakkoliv měnit a proces výběru KB se tomu plně přizpůsobí. Počet KB je také závislý na hodnotách osvětlenosti, čím větší osvětlenost bude naměřena, tím menší počet

bodů bude potřeba. Pro možnost porovnání byla zadávána po celý proces stejná hodnota osvětlenosti. Změřené závislosti jsou uvedeny v tabulce 3.1.

Tabulka 3.1: Počet KB a délka výpočtu v závislosti na hodnotě osvětlenosti a vzdálenosti vnitřního polygonu od zdi

		Osvětlenost [lx]					
		600	800	1000	600	800	1000
Vzdálenost [m]	0,5	36	31	28	5:34	4:37	4:07
	1	26	22	21	3:52	3:12	3:05
		Počet KB [-]			Délka výpočtu [min]		

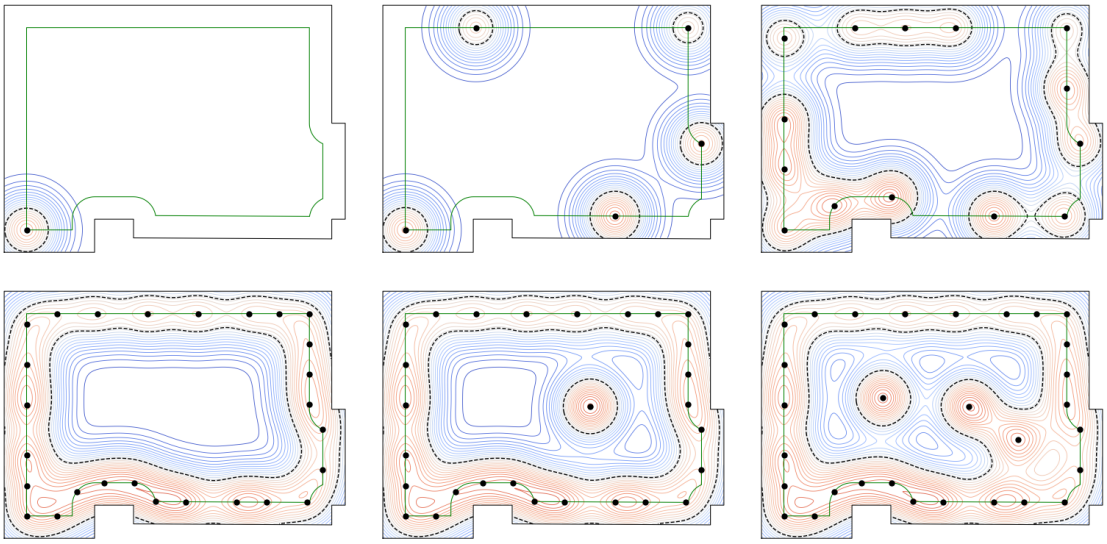
3.1.4 Simulace procesu měření

Úspěšná kontrola

K testování byla použita členitá místnost obsahující 9 svítidel, která byla nasimulována pomocí DIALuxu (viz kapitola 2.1.3). Tedy hodnoty odpovídají simulaci reálného šíření osvětlenosti daného typu svítidla. Vzdálenost vnitřního polygonu od zdi je 0,5 m. Nejprve se simulovala místnost, která má funkční všechna svítidla.

Proces výpočtu KB ve zmiňované místnosti je zobrazen na obrázku 3.3. Na obrázku je také možno vidět, že aby kontrola byla úspěšná nemusí být celá plocha pokryta KB, jak je tomu při manuálním kontrole podle normy [2]. Je to způsobeno podmínkou, která prohlásí kontrolu za úspěšnou, jakmile je průměrná spodní hranice nad 500 lx.

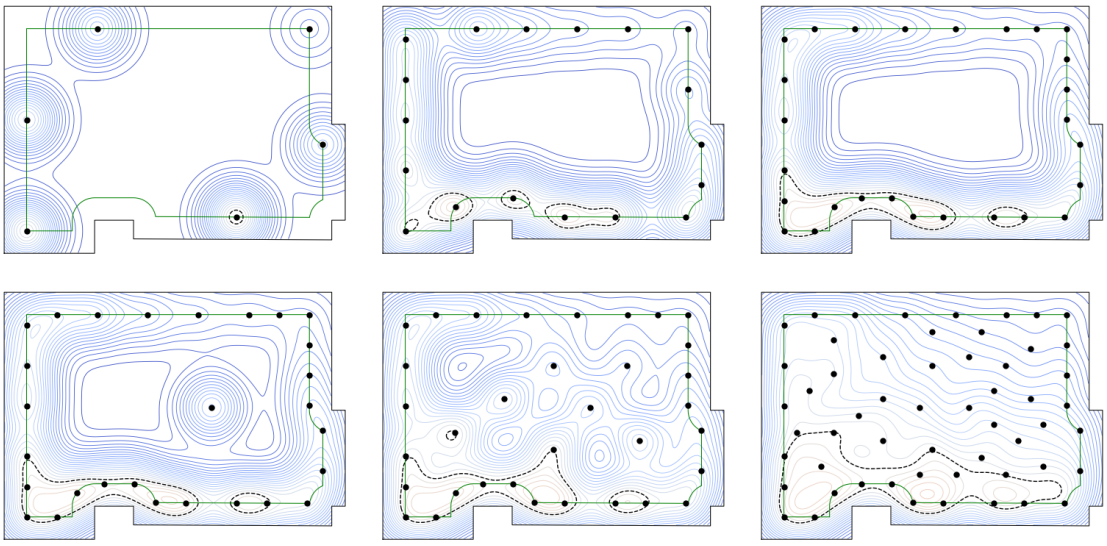
Pro testovací účely se simulovaly různé výpadky svítidel (viz příloha A). První simulací s výpadkem svítidla byla místnost, ve které fungovalo 8 svítidel (obrázek 3), tedy oproti původní místnosti zde bylo jedno svítidlo s poruchou. Nicméně i přes poruchu místnost kontrolou prošla. Stejný průběh byl i u dalších dvou místnostech, v jedné bylo funkčních 7 svítidel (obrázek 4) a v druhé bylo sice opět 8 svítidel, ale byly rozloženy jen na kraji místnosti a uprostřed místnosti byla osvětlenost nižší než 500 lx (obrázek 5). Ve výsledku všechny tyto místnosti i přes různé typy poruch kontrolou úspěšně prošly.



Obrázek 3.3: Simulovaný proces výpočtu KB (úspěšná kontrola)

Neúspěšná kontrola

Pro simulaci neúspěšné kontroly byla zvolena místnost, kde jsou umístěna pouze 4 svítidla. Rozvržení svítidel včetně nasimulované osvětlenosti je zobrazeno v příloze A na obrázku 6. Jelikož je pro ukončení měření potřeba splnit podmínku, kdy všechny body budou změřeny s maximální dovolenou odchylkou 10 lx, tak byl potřeba větší počet KB, konkrétně 56 bodů. Celý proces výpočtu KB u neúspěšné kontroly je pro porovnání s úspěšnou kontrolou z obrázku 3.3 zobrazen na obrázku 3.4.



Obrázek 3.4: Simulovaný proces výpočtu KB (neúspěšná kontrola)

3.2 Testování robotické jednotky pro měření osvětlenosti – reálné podmínky

3.2.1 Povrch

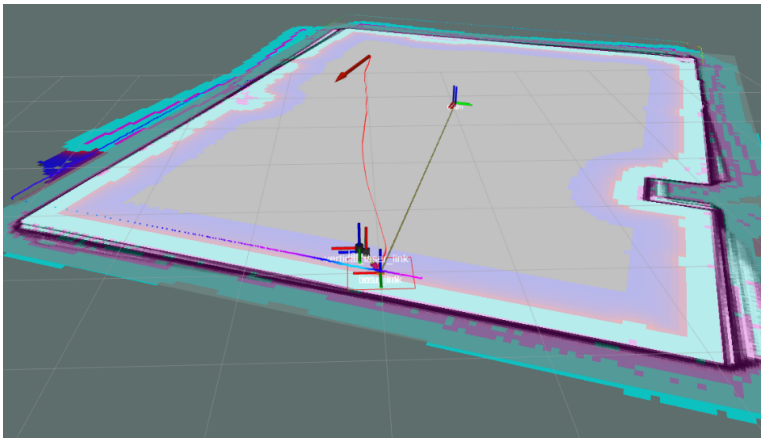
Testování RJ v reálných podmínkách nejprve začalo v místnosti, kde byl koberec. Nicméně RJ se nezvládala ani normálně pohybovat při začátku mapování. Často se stávalo, že motory ukazovaly, že jedou, ale žádný pohyb se nevykonával. Zjistilo se, že nejpravděpodobněji to způsobuje právě daný povrch. Přešlo se tedy do místnosti, kde bylo linoleum, a tam RJ začala jezdit plynuleji.

3.2.2 Rozjíždění robotické jednotky a najíždění do zakázané zóny

Při rozjíždění RJ je potřeba překonat prvotní tření. Momentálně je tento problém vyřešen tak, že jsou zavedeny parametry "anti_dead_zones", jeden pro pohyb vpřed a jeden pro otáčivý pohyb. Těmito parametry se požadovaná rychlost násobí, dokud se RJ nepohne. Nicméně při testování bylo zjištěno, že tento způsob rozjíždění zapříčiní, že pokud bude moc malá nastavená povolená vzdálenost od zdi, tzv. "safe_zone", tak RJ při rozjezdu blízko stěny nezvládne včas zareagovat a mohlo by dojít ke srážce.

Z tohoto důvodu nebylo možné při reálném testování použít vzdálenost 0,5 m od zdi, který je uváděn v normě [1]. Byla použita vzdálenost nejprve 2 m od zdi a poté 1,5 m od zdi, při menší vzdálenosti vždy došlo najetí RJ do zakázané zóny, ze které již nebyla schopna vyjet po naplánované trase (viz obrázek 3.5).

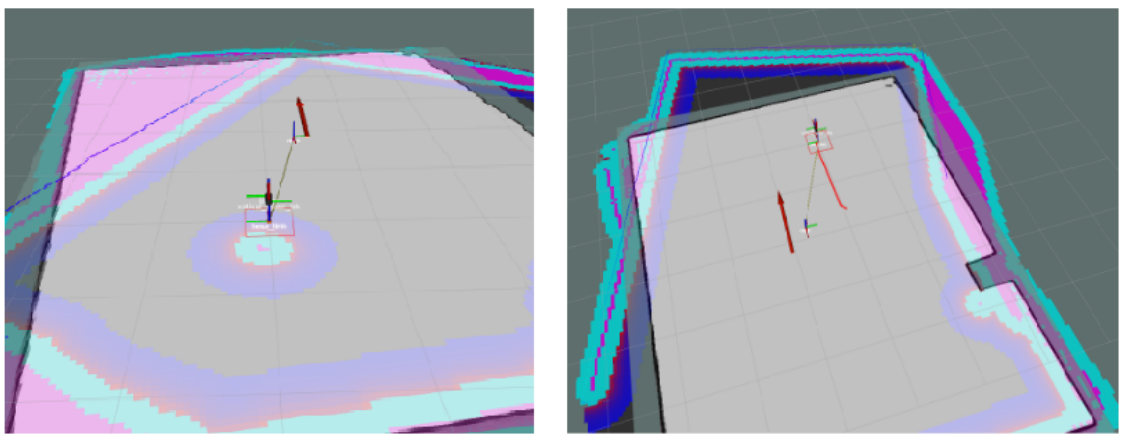
Problém s rozjížděním by mohl být vyřešen odstraněním parametrů na násobení rychlosti "anti_dead_zones". Možným řešením by mohlo být implementování postupného rozjíždění, které by nebylo tak razantní. Jelikož u momentálního řešení se stávalo, že někdy bylo tření překonáno hned a deaktivace "anti_dead_zones" nebyla včas zahájena. Tím právě mohlo vzniknout, že RJ nestihla detekovat, že je v zóně jen pro otáčení, a vykonala pohyb směrem ke zdi.



Obrázek 3.5: Najetí RJ do zakázané zóny – červená křivka zobrazuje plánovanou trasu k dalšímu bodu, souřadnicový systém v červeném obdélníku značí aktuální polohu RJ, souřadnicový systém uprostřed místnosti ukazuje bod, ve kterém byla RJ spuštěna, zakázaná zóna je vyznačena světle modrou barvou, světle fialovou je označena zóna, kde by mělo dojít už jen k otáčení a nikoliv pohybu blíže ke stěně. Značení je stejné pro všechny obrázky v této kapitole vykreslující mapu pomocí RViz.

3.2.3 Rozpadání mapy

Během prvních dnů testování docházelo často k rozpadu mapy. Většinou se tomu stalo při rozjíždění ze zastavení, kdy se chce po motorech velká změna v rychlosti, kvůli počátečnímu tření. Rozpad mapy se poznal tak, že půdorys místnosti ve vizualizaci v RVizu byl několikrát přes sebe různě otočený. V případě rozpadu mapy nebylo bezpečné v měření pokračovat, jelikož mapa neodpovídala reálnému prostoru a RJ by mohla narazit do zdi.



Obrázek 3.6: Příklady rozpadu mapy vizualizované přes RViz

Zjistilo se, že rozpad mapy způsobuje protočení hřídele v kolech, jelikož odometrie určila, že se RJ pohnula, avšak k žádnému reálnému pohybu nedošlo. Po přidání druhého upevňovacího bodu na hřídel docházelo k rozpadu mapy méně. Nicméně stále to zcela problém nevyřešilo, protože k tomuto problému i poté několikrát došlo. Je tedy spíše potřeba danému problému předcházet a před každým měřením kola na hřídel utáhnout.

3.2.4 Motory

Problém při rozjezdu robotické jednotky

Při testování se narazilo na další problém spojený s rozjížděním RJ (metoda, jak je vyřešené rozjíždění RJ, je blíže popsaná v kapitole 3.2.2). Když mělo dojít k rozjezdu, či otočení na místě, tak RJ nebyla schopna pohyb vykonat. Nejednalo se zde o protáčení hřídeli v kolu, ale o samotný problém překonání prvotního tření, jelikož když se RJ zvedla do vzduchu, kola se točila. Pokud se do RJ nepatrně postrčilo, pak zvládla kontrolu ve většině případech dokončit, nicméně při pár testech musela tato pomoc přijít vícekrát.

Bylo zjištěno, že tento problém je nejspíše způsoben poklesem napětí v akumulátoru. Po výměně akumulátoru za plně nabitý se ve většině případech rozjíždění zprovoznilo. Nicméně proběhly i testy, kdy byl akumulátor plně nabitý a ke stejnému problému došlo. Dalším nápadem bylo, že parametry "anti_dead_zones" jsou nastavené na moc malou hodnotu a bylo by potřeba je zvýšit. Po zvýšení těchto parametrů se ale stávala RJ nekontrolovatelnou a bylo riziko nárazu do překážky.

Možným řešením by pravděpodobně opět bylo implementování jiného způsobu rozjezdu RJ, např. již zmiňované postupné rozjíždění.

Nechtěné zatáčení na jednu stranu

Během jízdy do následujícího KB bylo detekováno zatáčení RJ na levou stranu, ačkoliv vypočtená trasa byla rovná. Tento jev způsoboval najetí do zakázané zóny. Bylo podezření na chybný levý motor, který nezvládal fungovat stejně jako ten pravý. Chyba se vyskytovala především u delších tras, které měly plánovanou téměř rovnou trasu. Vyměnil se tedy levý motor za jiný a daný jev se již nevyskytl.

3.3 Finální testy a přesnost najíždění do koncového bodu

Při testování celého procesu na reálných podmínkách byly použity vzdálenosti od zdi 1,5 m a 2 m. Při menší vzdálenosti kontrola nebyla dokončena, jelikož se RJ dostala do zakázané zóny. Cílem bylo zajistit co nejpřesnější dojezd do koncového bodu, který se shodoval s počátečním bodem. V průběhu testování se zkoušel měnit parametr, který určoval toleranci, s jakou bude uskutečněn nájezd do zvoleného bodu. Parametr byl nastavován na 0,2 m a 0,1 m, při nižší toleranci docházelo k dlouhému nastavování správné polohy v daném KB.

V první místnosti, kde probíhalo i řešení problémů, byly zkoušeny všechny zmíněné parametry. Po každé změně parametrů se provedlo měření a zaznamenala se vzdálenost reálné polohy RJ od předpokládaného koncového bodu. Naměřené hodnoty jsou uvedeny v tabulce 3.2. Pro vzdálenost od zdi 1 m nebyla schopna RJ kontrolu dokončit pro žádnou hodnotu tolerance.

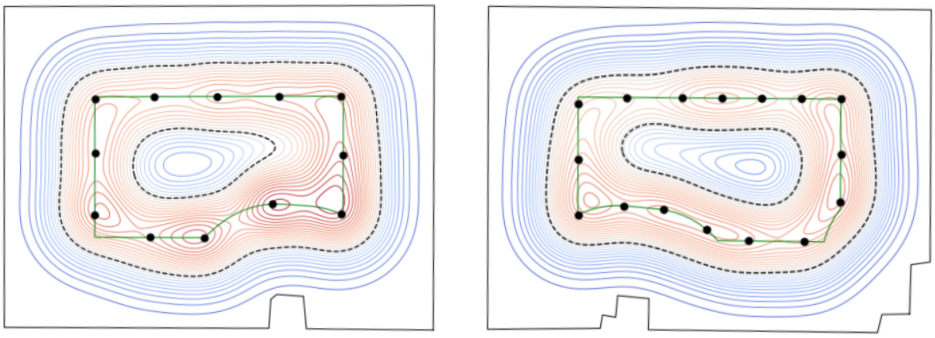
Tabulka 3.2: Vzdálenost RJ od koncového bodu v závislosti na toleranci najíždění a vzdálenosti od zdi, pro 1 m nebyla kontrola dokončena a tak nebylo možné vzdálenost určit

		Vzdálenost od zdi [m]		
		2	1,5	1
Tolerance [m]	0,1	8	2	-
	0,2	16	8	-
		Vzdálenost od koncového bodu [cm]		

U měření pro vzdálenost 2 m od zdi bylo potřeba k celé kontrole 8 KB, pro vzdálenost 1,5 m jich bylo 10. Jelikož u tolerance 0,1 m došlo dle očekávání k přesnějšímu nájezdu do finálního bodu, byla tato hodnota zvolena pro finální test. Chtěla jsem se co nejvíce přiblížit normě, tak jsem pro finální test zvolila i menší vzdálenost od zdi, tedy 1,5 m.

Finální test v obou místnostech je vykreslen na obrázku 3.7. V obou případech proběhlo měření bez problémů a místnosti kontrolou úspěšně prošly. Vzhledem k velké vzdálenosti od zdi byly KB voleny pouze z hranice vnitřního polygonu. V první místnosti stačilo k celkové kontrole 12 KB a do koncového bodu najela

RJ s odchylkou 11 cm. V druhé místnosti bylo potřeba 16 KB a odchylka byla 8 cm. Nižší počet bodů v první místnosti byl způsoben převážně naměřením vyšších hodnot osvětlenosti.



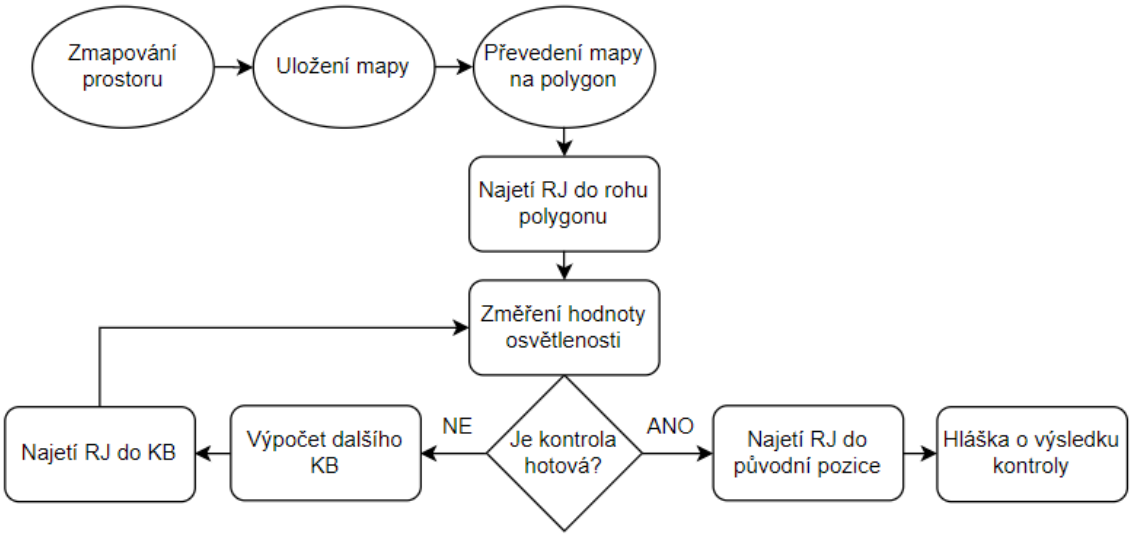
(a) Místnost 1 **(b)** Místnost 2

Obrázek 3.7: Finální test v reálných podmínkách v porovnání pro dvě místnosti různých půdorysů – vykreslení naměřené osvětlenosti pro parametry: tolerance 0,1 m a vzdálenost od zdi 1,5 m.

4. Návod

V této kapitole je sepsán podrobný návod na obsluhu RJ. Je zde uveden všechen potřebný software, např. knihovny, které jsou nezbytné pro spuštění jednotlivých kódů, a samotný návod na jejich instalaci. Dále je uvedeno, co je potřeba připravit před samotným měřením a jak s RJ pracovat v průběhu měření. Podrobně je sepsáno, jak se jaký kód spouští a v jakém adresáři se nachází.

Blokový diagram procesu měření je zobrazen na obrázku 4.1. Jednotlivé kroky procesu jsou popsány v následujících podkapitolách. Odlišení softwarové části, která byla nově implementována v rámci této práce (viz kapitola 2) a té, která již byla implementována dříve, je znázorněno tvary bloků.



Obrázek 4.1: Proces kontroly osvětlenosti v interiéru. Obdélníkové bloky jsou součástí kódu, jenž byl implementován v rámci této práce, více v kapitole 2.

4.1 Instalace knihoven

Pro zajištění bezchybného spuštění všech používaných kódů je potřeba mít nainstalované požadované verze knihoven. Knihovny, potřebné pro spuštění kódu pro výpočet KB, který byl implementován v rámci této práce, jsou uvedeny v tabulce 4.1. Kompletní seznam všech knihoven potřebných pro celý proces měření osvětlenosti včetně najíždění do zadaných bodů je uveden v příloze B.

Pro zjištění seznamu již nainstalovaných knihoven stačí zadat do příkazového řádku příkaz "python -m pip list --user". Případně je možné je podle uvedeného příkazu nainstalovat. U většiny knihoven stačí pro instalaci do příkazového řádku napsat "pip install jmeno_knihovny==verze", nicméně jak je vidět, tak u několika knihoven je příkaz pro instalaci jiný.

Je to způsobeno tím, že řídicí počítač, který je k robotu dostupný, má velikost operační paměti pouze 4 GB a nainstalování těchto knihoven byla náročná operace. Proto je zvolena tato varianta instalování, kterou již řídicí počítač byl schopen zvládnout. Dalším důvodem bylo použití dvou verzí Pythonu, jak jsem již zmiňovala v kapitole 2.3.

Tabulka 4.1: Seznam knihoven potřebných pro implementaci výpočtu KB včetně návodu na jejich instalaci

Název knihovny	**Verze**	**Příkaz pro instalaci**
matplotlib	2.2.5	pip install matplotlib==2.2.5
numpy	1.16.6	pip install numpy==1.16.6
pandas	0.24.2	pip install pandas==0.24.2
pyserial	3.4	pip install pyserial==3.4
scipy	1.2.3	pip install scipy==1.2.3
Shapely	1.7.1	pip install Shapely==1.7.1
subprocess32	3.5.4	pip install subprocess32==3.5.4
torch	1.8.1	python -m pip install --default-timeout=100 torch==1.8.1 --no-cache-dir
gpytorch	1.4.1	python -m pip install --default-timeout=100 gpytorch==1.4.1 --no-cache-dir

4.2 Příprava před měřením

Před samotným měřením je potřeba vědět, jaký typ svítidla je v prostoru, kde bude probíhat reálné měření. Pomocí DIALux [15] se nasimuluje černá obdélníková místnost, ve které bude jedno svítidlo stejného typu. Vyexportuje se soubor typu .csv a ten se uloží do stejné složky jako jsou kódy, tedy adresář Plocha/codes/data.

V kódu na trénování parametrů modelu podle šíření osvětlenosti, který má název train_param.py, se upraví cesta k souboru podle jeho názvu. V případě, že je zvolena jiná velikost místnosti než 5 × 7,5 m, je třeba ještě upravit nastavení polygonu. Po všech zmíněných úpravách se program spustí otevřením terminálu ve správném adresáři a samotným spuštěním, zadá se tedy do příkazového řádku po jednom:

- cd Plocha/codes
- python3 train_param.py

Po spuštění kódu se začnou vypisovat parametry po jedné iteraci. Po skončení programu bude výstupem natrénovaný model uložen v souboru model.pth, který je dále využit v kódu na výpočet KB.

4.3 Proces měření

4.3.1 Zapnutí

Před zapnutím je nutné zkontrolovat, zda je nabitý a připojený akumulátor. RJ se spustí pomocí zapnutí třech přepínačů, které jsou umístěny na levém boku jednotky, do polohy I. Také je nutné spustit řídicí počítač. Luxmetr se spustí tlačítkem na zapnutí, které se nachází vpravo dole, poté je třeba ho přepnout do stavu měření (tlačítko vlevo nahoře).

4.3.2 Nastavení vstupů počítače

Během měření je použita sériová komunikace mezi počítačem, lidarem, arduinem a luxmetrem. Aby se porty nemusely v kódech pokaždé měnit, stačí připojené zařízení pojmenovat podle následujícího nastavení. Tedy zapojí se jedno zařízení

pomocí USB do řídicího počítače RJ a hned poté se napíše do příkazového řádku příkaz z následující nabídky, tato operace se provede postupně pro všechna zařízení v pořadí lidar, arduino a nakonec luxmetr:

- lidar: sudo chmod a+rw /dev/ttyACM0
- arduino: sudo chmod a+rw /dev/ttyACM1
- luxmetr: sudo chmod a+rw /dev/ttyUSB0

4.3.3 Spuštění programů

Příkazy, které jsou v této podkapitole, se zadávají do příkazového řádku po jednom v uvedeném pořadí. Nejprve se spustí hlavní uzel ROSu "roscore" a poté "urg_node", který spouští komunikaci mezi řídicím počítačem a lidarem:

- roscore
- rosrun urg_node urg_node

Pro další příkazy musí být příkazový řádek otevřený v zadaném adresáři, ten se otevře pomocí příkazu:

- cd BAKALARKA/PC/main

Zde je již možné spustit daný kód. Kód "my_robot_node.py" slouží pro komunikaci s Arduinem, díky kterému se na terminálu budou vypisovat rychlosti pravého a levého motoru:

- python my_robot_node.py

Pro vrácení zpět do domovského adresáře se použije:

- cd

Další kódy se budou spouštět obdobným způsobem:

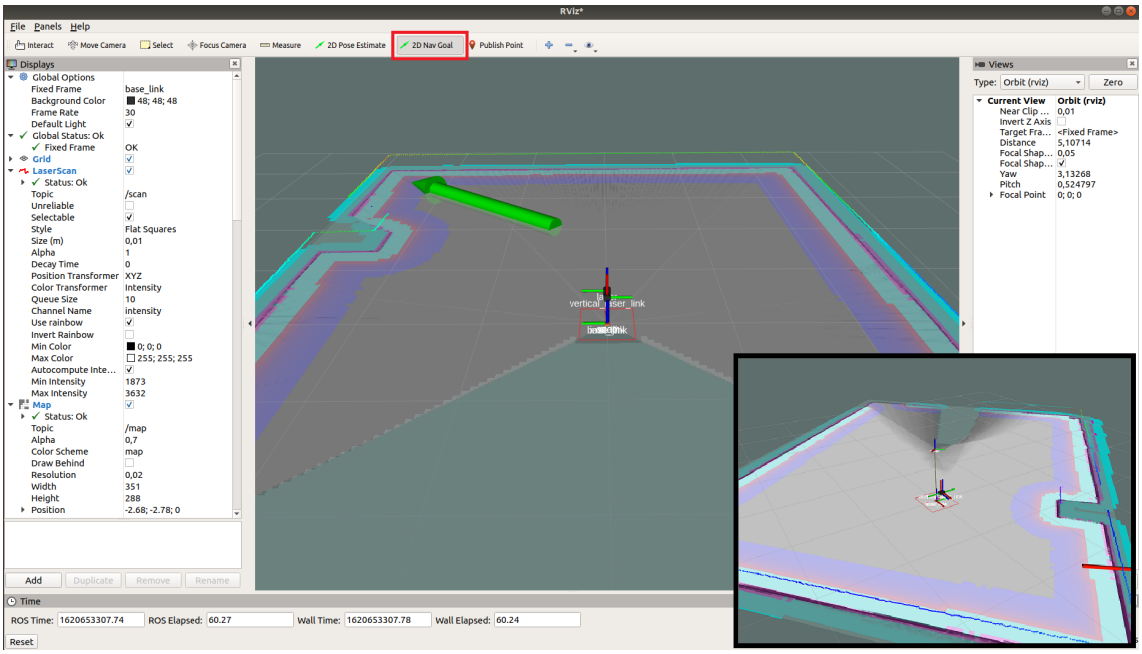
- cd catkin_ws/move_base
- roslaunch my_move_base.launch
- cd

- cd catkin_ws/cartographer
- roslaunch my_backpack_2d.launch

Dalším uzlem ROSu je "my_move_base", ten zajišťuje plánování trasy do zadaného cíle. Poslední příkaz "my_backpack_2d" spouští vizualizaci mapy a plánované trasy.

4.3.4 Průběh

Po spuštění posledního programu se otevře okno RViz [21], kde je vizualizace části místnosti, kterou má RJ v zorném poli. Pro zmapování celé místnosti je potřeba nechat RJ projet po celé místnosti. To se provede tak, že se vybere v horní liště "2D Nav Goal" a zvolí se místo na mapě, kam se má RJ přemístit. Místo se zvolí kliknutím do mapy, v případě delšího podržení tlačítka a současným pohybem myši lze určit směr výsledného natočení (viz obrázek 4.2). Pokud je vizualizace místnosti hotová, tedy celá plocha je vyplněna šedou barvou a hranice plochy odpovídá půdorysu reálné místnosti, může se pokročit k uložení mapy.



Obrázek 4.2: Vizualizace mapy pomocí RViz před začátkem měření – v horní liště je červeně vyznačena nabídka pro výběr následujícího bodu, po nakliknutí je zobrazena na mapě zelená šipka znázorňující výsledné natočení RJ, modro-fialové ohraničení plochy značí zdi či překážky v místnosti. Vpravo dole je přidaná vizualizace mapy po najetí do zvoleného bodu – již zmapovaná oblast je označena šedou barvou.

Před uložením mapy je potřeba zvolit adresář, kde jsou uložené kódy, které s mapou pracují. Dále je pak možné mapu uložit. Celý proces otevření adresáře a uložení mapy se provede zadáním do příkazového řádku:

- cd Plocha/codes
- rosrun map_server map_server -f mymap

Pro převedení mapy na polygon se ve stejném adresáři spustí kód na aproximaci jako:

- python contour_approximation.py

Tím se uloží soubor obsahující vrcholy polygonu, které jsou dále zpracovávané v kódu na výpočet KB. Pro započetí samotné kontroly měřené osvětlenosti se spustí program na výpočet KB, který v sobě zahrnuje i najíždění konkrétních míst. Opět ve stejném adresáři se v terminálu napíše:

- python3 room_meas.py

4.3.5 Ukončení

Když kontrola skončí, RJ dojede na stejné místo, na které byla umístěna před začátkem mapování. Tím je možné si ověřit, jak přesné najíždění do KB bylo. V terminálu, kde byl spuštěn kód na výpočet KB, se vyskytuje upozornění, zda byla kontrola měření osvětlenosti v dané místnosti úspěšná či nikoliv. Je možné také shlédnout vizualizaci šíření osvětlenosti, jelikož se po každém obdržení nového měření uloží simulace šíření vytvořena z již naměřených dat. Snímky jsou ukládány do adresáře Plocha/codes/data.

Závěr

Hlavním cílem této práce bylo částečné zautomatizování procesu měření umělého osvětlení v interiéru. Pro dosažení zmíněného cíle bylo potřeba se seznámit s robotickou jednotkou (viz kapitola 1) a algoritmem pro výpočet kontrolních bodů pro měření osvětlenosti (viz Úvod).

Průběh implementace softwaru je zaznamenán v kapitole 2. Během něj došlo k několika problémům, které se podařily vyřešit. Zmínila bych ten z mého pohledu největší, který se potýkal s nekompatibilitou knihoven, řešením bylo použití jiné knihovny. Při implementaci nenastala žádná překážka, která by se v průběhu práce nevyřešila. Software byl otestován na různých typech kontroly (viz kapitola 3.1) a nebyly zjištěny žádné nedostatky. Výstupem je tedy zcela funkční již otestovaný software na výpočet kontrolních bodů pro měření osvětlenosti v interiéru.

Pro možnost budoucí obsluhy robotické jednotky bylo nutné sepsat uživatelský návod (viz kapitola 4). Návod je strukturován krok po kroku a pomocí něj by měla být schopna i neznalá osoba kontrolu osvětlenosti v dané místnosti provést. Další část návodu je uvedena přímo v kódu v podobě komentářů, s nimiž je provázen celý kód pro výpočet kontrolních bodů, aby bylo zřejmé co jaká funkce dělá.

K hlubšímu seznámení s robotickou jednotkou proběhlo až při testování. Během testování bylo nalezeno několik nedostatků při pohybu jednotky jak už hardwarových tak softwarových. Jedním z hardwarových nedostatků bych uvedla nešťastné řešení připevnění kol na hřídeli, jelikož kola se po delším užívání povolí a dojde k protáčení hřídeli, což způsobuje návazné potíže zmiňované v kapitole 3.2.

Hlavním softwarovým problémem, který provázel celé testování, bylo implementované rozjíždění robotické jednotky pomocí násobení prvotní rychlosti koeficienty. Zmíněné řešení způsobovalo mnoho dalších problémů, a tak pro hladký průběh měření by bylo nejlepším řešením implementovat jiný způsob rozjezdu, např. navrho-

vané postupné uvedení do pohybu.

V souhrnu je tedy dostupný otestovaný software pro výpočet kontrolních bodů pro měření umělé osvětlenosti v interiéru, který je možné použít pro jakoukoliv robotickou platformu. Otestováním algoritmu za pomoci robotické jednotky v reálných podmínkách byly splněny a zkontrolovány cíle této práce. Možným rozšířením by byla automatizace prvotního mapování prostoru, po čemž bylo potřeba celý proces znovu otestovat. Tím by se proces plně zautomatizoval. Bylo by vhodné navíc navrhnout a přidat řešení, jak dále zpracovávat naměřená data do odpovídajícího protokolu o kontrole osvětlenosti v interiéru.

Bibliografie

1. ČSN EN 12464-1: Světlo a osvětlení - Osvětlení pracovních prostorů - Část 1: Vnitřní pracovní prostory. *Český normalizační institut*. 2012.
2. ČSN 36001-1: Měření osvětlení vnitřních prostorů – Část 1: Základní ustanovení. *Český normalizační institut*. 2015.
3. PETR, Kůrka. *Software pro robotickou jednotku, která měří osvětlenost v interiérech v síti kontrolních bodů*. 2020. Bakalářská práce. České vysoké učení technické v Praze. Výpočetní a informační centrum.
4. DRÁBEK, Tomáš; PETRÍK, Vladimír; HOLUB, Jan. Statistical-Based Control Points Selection for Indoor Illuminance Measurement. *IEEE Transactions on Instrumentation and Measurement*. 2020, roč. 69, č. 10, s. 8362–8371.
5. TOMÁŠ, Drábek. *Robotická jednotka pro měření osvětlenosti v interiérech v rovnoměrné optimalizované síti kontrolních bodů*. 2014.
6. *MD25 - Dual 12Volt 2.8Amp H Bridge Motor Drive* [online] [cit. 2021-04-18]. Dostupné z: `https://www.robot-electronics.co.uk/htm/md25tech.htm`.
7. *EMG30, mounting bracket and wheel specification* [online] [cit. 2021-04-18]. Dostupné z: `https://www.robot-electronics.co.uk/htm/emg30.htm`.
8. *RD02 - 12v robot drive* [online] [cit. 2021-04-18]. Dostupné z: `https://www.robot-electronics.co.uk/rd02-12v-robot-drive.html`.
9. *Distance Data Output/UTM-30LX* [online] [cit. 2021-04-18]. Dostupné z: `https://www.hokuyo-aut.jp/search/single.php?serial=169`.
10. *Radiolux 111* [online]. 2017 [cit. 2021-04-09]. Dostupné z: `https://luxmetry.cz/product/radiolux/`.

11. *Olověný akumulátor Shimastu NPG8-12, 12V 8Ah* [online]. 2006 [cit. 2021-04-09]. Dostupné z: https://www.emerx.cz/oloveny-akumulator-shimastu-npg8-12-12v-8ah.html.

12. RASMUSSEN, Carl Edward. Gaussian processes in machine learning. In: *Summer school on machine learning*. 2003, s. 63–71.

13. *GPyTorch Regression Tutorial* [online] [cit. 2021-04-11]. Dostupné z: https://docs.gpytorch.ai/en/stable/examples/01_Exact_GPs/Simple_GP_Regression.html.

14. *gpytorch.likelihoods* [online] [cit. 2021-04-18]. Dostupné z: https://docs.gpytorch.ai/en/stable/likelihoods.html.

15. *DIALux - DIAL* [online] [cit. 2021-04-22]. Dostupné z: https://www.dial.de/en/dialux/.

16. *pandas documentation* [online] [cit. 2021-04-22]. Dostupné z: https://pandas.pydata.org/docs/index.html.

17. *TensorFlow Core* [online] [cit. 2021-05-04]. Dostupné z: https://www.tensorflow.org/overview.

18. *NumPy* [online] [cit. 2021-05-04]. Dostupné z: https://numpy.org/.

19. *PyTorch* [online] [cit. 2021-05-04]. Dostupné z: https://pytorch.org/.

20. *Cartographer* [online] [cit. 2021-04-30]. Dostupné z: https://google-cartographer.readthedocs.io/en/latest/.

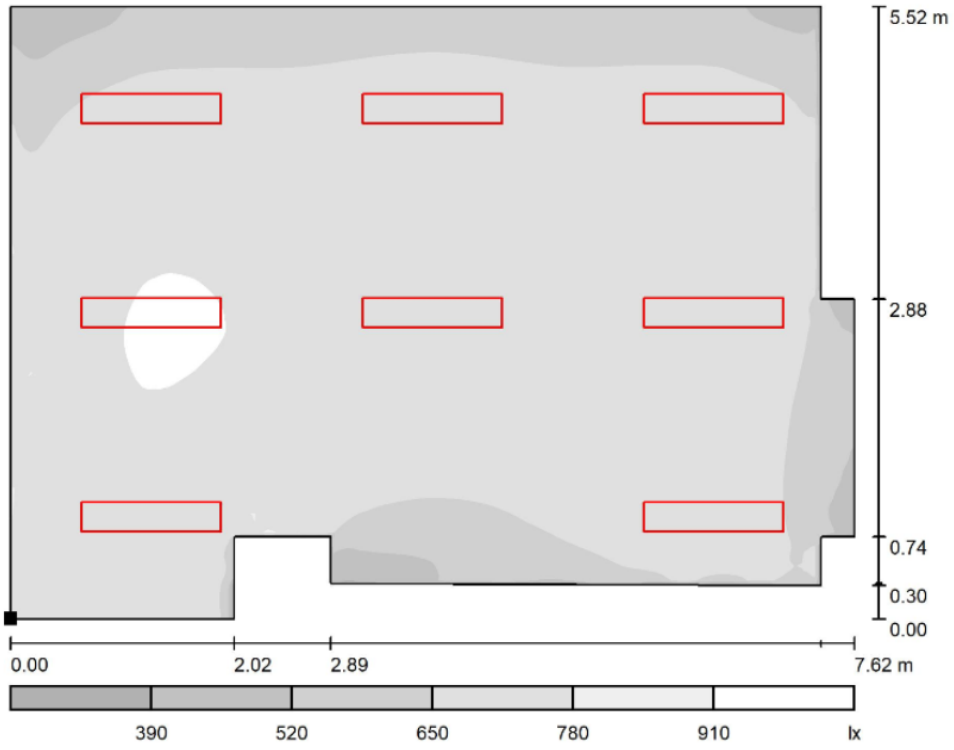
21. *rviz/UserGuide* [online] [cit. 2021-05-07]. Dostupné z: http://wiki.ros.org/rviz/UserGuide.

22. *ROS* [online] [cit. 2021-05-02]. Dostupné z: https://www.ros.org/.

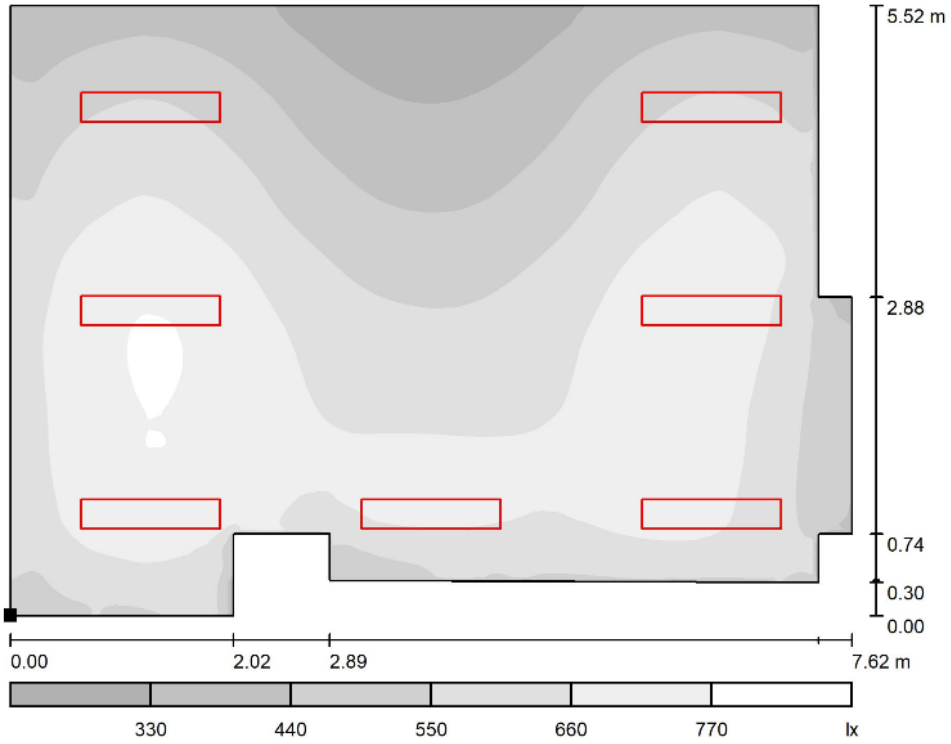
23. *subprocess — Subprocess management* [online] [cit. 2021-05-02]. Dostupné z: https://docs.python.org/3/library/subprocess.html.

Přílohy

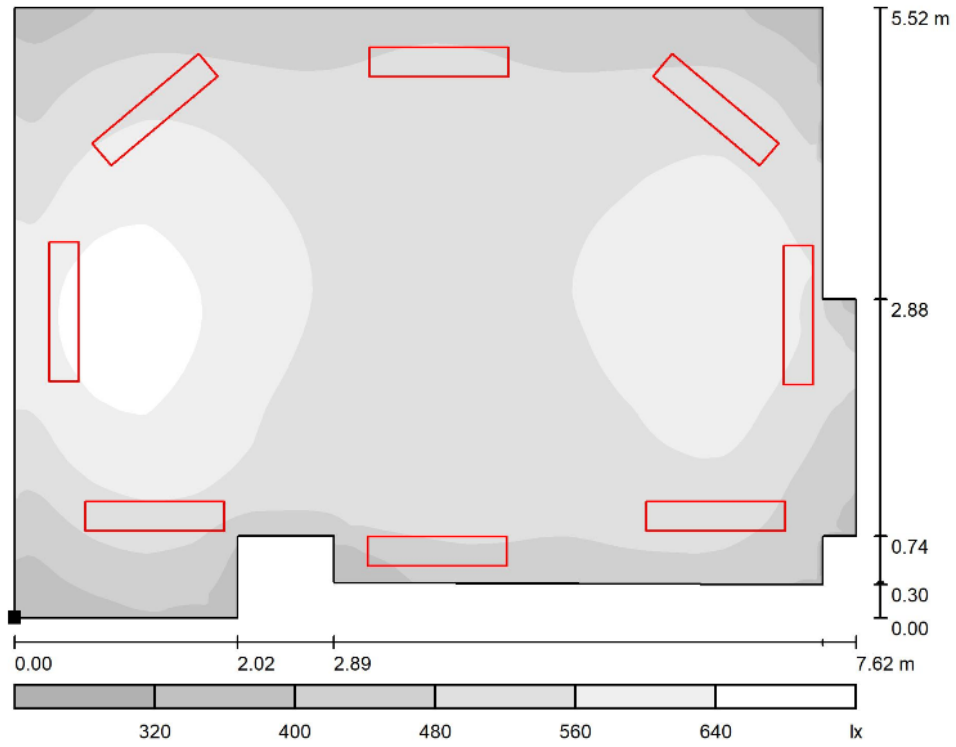
A Rozvržení svítidel při poruchách v simulované místnosti



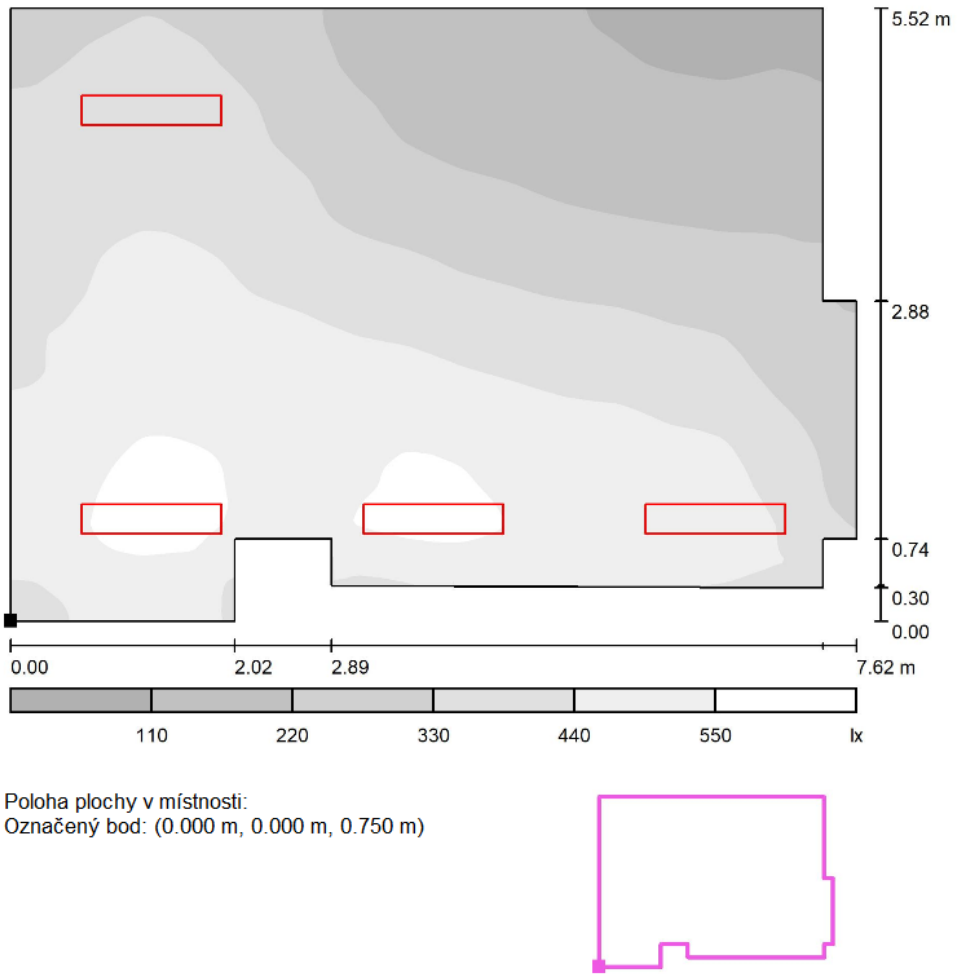
Obrázek 3: Simulovaná osvětlenost pro 8 svítidel pomocí DIALuxu



Obrázek 4: Simulovaná osvětlenost pro 7 svítidel pomocí DIALuxu



Obrázek 5: Simulovaná osvětlenost pro 4 svítidla s chybějícím středem pomocí DIALuxu



Obrázek 6: Simulovaná osvětlenost pro 4 svítidla pomocí DIALuxu

B Seznam nainstalovaných knihoven

Tabulka 2: Seznam nainstalovaných knihoven včetně návodu na jejich instalaci – 1. část

Název knihovny	**Verze**	**Příkaz pro instalaci**
backports.functools -lru-cache	1.6.4	pip install backports.functools-lru-cache==1.6.4
cycler	0.10.0	pip install Cycler==0.10.0
distro	1.5.0	pip install distro==1.5.0
future	0.18.2	pip install future==0.18.2
kiwisolver	1.1.0	pip install kiwisolver==1.1.0
matplotlib	2.2.5	pip install matplotlib==2.2.5
numpy	1.16.6	pip install numpy==1.16.6
opencv-python	4.2.0.32	pip install opencv-python==4.2.0.32
packaging	20.9	pip install packaging==20.9
pandas	0.24.2	pip install pandas==0.24.2
pyparsing	2.4.7	pip install pyparsing==2.4.7
pyserial	3.4	pip install pyserial==3.4
python-dateutil	2.8.1	pip install python-dateutil==2.8.1
pytz	2021.1	pip install pytz==2021.1
PyYAML	5.4.1	pip install PyYAML==5.4.1
scikit-build	0.11.1	pip install scikit-build==0.11.1
scipy	1.2.3	pip install scipy==1.2.3

Tabulka 3: Seznam nainstalovaných knihoven včetně návodu na jejich instalaci – 2. část

setuptools	44.1.1	pip install setuptools==44.1.1
Shapely	1.7.1	pip install Shapely==1.7.1
six	1.15.0	pip install six==1.15.0
subprocess32	3.5.4	pip install subprocess32==3.5.4
tqdm	4.60.0	pip install tqdm==4.60.0
tripy	1.0.0	pip install tripy==1.0.0
wheel	0.36.2	pip install wheel==0.36.2
torch	1.8.1	python -m pip install --default-timeout=100 torch==1.8.1 --no-cache-dir
gpytorch	1.4.1	python -m pip install --default-timeout=100 gpytorch==1.4.1 --no-cache-dir
future	0.18.2	python -m pip install --default-timeout=100 future==0.18.2 --no-cache-dir