FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Diplomová práce

Nový řídící systém pro robota Hexor II

Studijní program: N0714A270010 Mechatronika
Autor práce: **Bc. Tomáš Hmiro**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání diplomové práce

Nový řídící systém pro robota Hexor II

Jméno a příjmení:	**Bc. Tomáš Hmiro**
Osobní číslo:	M21000176
Studijní program:	N0714A270010 Mechatronika
Zadávající katedra:	Ústav informačních technologií a elektroniky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Seznamte se se stávajícím stavem softwarového a hardwarového vybavení mobilního robota Hexor II na pracovišti školitele.
2. Navrhněte nový řídící systém pro robota s využitím mikrokontroleru řady STM32Fxxx, možností bezdrátové komunikace přes wiffi rozhraní a možností inerciální navigace.
3. Pro detekci překážek a okolí použijte stávající senzorické vybavení, případně navrhněte vhodné rozšíření. Dále aktualizujte power managment napájecího aumulátoru.
4. Navržený systém realizujte a ověřte jeho funkčnost. Zhodnoďte dosažené výsledky.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40-50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. 1. vydání. Praha: BEN – technická literatura, 2005. ISBN 80-7300-141-1.

[2] ĎAĎO, Stanislav, KREIDL, Marcel. Senzory a měřící obvody. Praha : Vydavatelství ČVUT, 1996. 315 s. ISBN 80-01-02057-6. www.st.com

Vedoucí práce:	Ing. Miroslav Holada, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	24. října 2022
Předpokládaný termín odevzdání:	15. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Dr. Ing. Jaroslav Hlava
garant studijního programu

V Liberci dne 24. října 2022

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

8. května 2024 Bc. Tomáš Hmiro

Nový řídící systém pro robota Hexor® II

Abstrakt

Tato práce se zabývala robotem Hexor® II a jeho novým ovládáním, zprostředkovaným webovou stránkou přes ESP32, komunikační jednotkou, v práci nazvanou jako HUB, postavenou kolem STM32F722RET6, a senzorickou deskou, řízenou pomocí STM32F407VGT6.

Práce obsahuje shrnutí prací, které byly provedeny na hardwaru robotu i jeho softwarovém vybavení. Účelem práce bylo vybavit robot snáze dostupným řízením, jež poskytuje ESP prostřednictvím Wi-Fi a interního webserveru. Dále se věnovala spojením všech zmíněných čipů do ovládací komunikace robotu a kombinací tohoto všeho měla ambice vytvořit nový řídicí systém robotu. Také se zabývala změnou na napájecím systému v podobě nové baterie a nabíjecího zdroje.

Klíčová slova: LaTeX, Robot, TUL, Hexor® II, ESP32, STM32

Abstract

This work dealt with the Hexor® II robot and its new web control via ESP32, a communication unit in the work named as HUB built around STM32F722RET6, and a sensor board controlled by STM32F407VGT6.

The thesis contains a summary of the work that was done on the robot's hardware and its software equipment. The purpose of the work was to equip the robot with more accessible control provided by ESP via Wi-Fi and an internal web-server. It was also focused on connecting all the mentioned chips to the robot's control communication, and by combining all of this, the ambition was to create a new robot control system. It also dealt with a change to the power system in the form of a new battery and charging power supply.

Keywords: LaTeX, Robot, TUL, Hexor® II, ESP32, STM32

Poděkování

Rád bych poděkoval všem, kteří přispěli ke vzniku této práce. Počínaje rodinou, jež mi poskytla dostatek prostoru se této práci věnovat, také kolegům v práci, kteří nejednou přispěli radou, některé invence byly dokonce i implementovány do programu robota. A samozřejmě i vedoucímu práce Ing. Miroslavu Holadovi, Ph.D., s nímž bohužel nebylo tolik času konzultovat, kolik by možná bylo vhodné, avšak to jsem komplikoval já a mé zaměstnání na plný úvazek. Také bych rád poděkoval tomu, kdo bude tuto práci číst, a doufám, že byla alespoň v něčem nápomocna.

Obsah

Seznam obrázků

Seznam zkratek

TUL	Technická univerzita v Liberci
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
ESP32	system na čipu (mikrokontroler) od společnosti Espressif
STM32	mikrokontroler od společnosti STMicroelectronics
UART	Sběrnice – Universal asynchronous receiver-transmitter – Universální asynchronní přijímač-vysílač
USART	Sběrnice – Universal Synchronous/Asynchronous receiver-transmitter – Universální Synchronní/Asynchronní přijímač-vysílač
MSPS	MegaSamples Per Second – Milionů vzorků za sekundu
I2C	Sběrnice – Inter-Integrated Circuit – mezi obvodová komunikace
I2S	Sběrnice – Inter-Integrated Circuit Sound – I2C pro zvuk
SPI	Sběrnice – Serial Peripheral Interface – mezi periferní sériová komunikace
CAN	Sběrnice – Controller Area Network
PETG	Polyethylentereftalát glykol
PLA	polylactic acid – kyselina polymléčná
ABS	Akrylonitril Butadien Styren
TPE	Termoplastický elastomer
FDM	Fused deposition modeling – Výroba taveným materiálem
FFF	Fused filament fabrication – Výroba taveným vláknem
SLA	Stereolitografie
MSLA	Maskovaná stereolitografie
FPV	First person view – pohled první osoby
ELRS	Express (extremely sensitive) Long Range System – velmi citlivý daleko-dosahový systém

1 Úvod

Roboty a mobilní roboty se postupně dostávají z prostředí průmyslu i do ulic a domácností, kde plní rozmanité spektrum úkonů. Již před mnoha lety si lidé představovali, že nás roboty nahradí v těch nejvíce stereotypních činnostech. Nyní již není žádné překvapení vidět roboty, které v obchodech zametají a vytírají podlahy a po ulicích některých měst rozvážejí poštu, to vše díky pokrokům v oblastech počítačového vidění, automatizace, strojového učení a mnoha dalších. V dnešním světě se tak setkáváme s opravdu různými typy robotů. Od těch jednoúčelových stacionárních nebo pohyblivých operátorem přes ty s velmi jednoduchým autonomním režimem, jenž své okolí zkoumají jako slepci pouze za pomoci přímého dotyku s překážkami, až po ty plně autonomní, které rozeznávají své okolí do detailů za pomoci kamer, lidaru, ultrazvuku a dalších senzorů. Zároveň pomocí nich určují svou polohu v prostoru, již ještě zpřesňují pomocí matematických operací a dat z akcelerometrů, gyroskopu i jí dopočítávají z ujetých či ušlých vzdáleností.

Na tolik senzorů a jejich obsluhu už nestačí jeden osmibitový počítač. Každý z dnešních robotů obsahuje procesor, jenž celou funkci robota koordinuje, a mnoho dílčích procesorů. Ty chytřejší jsou jich doslova plné. Na každý i jen malý podsystém mohou mít alespoň jednu procesorovou jednotku.

Od koprocesoru, který řídí napájení, přes ty obstarávající chod a sled pohybů nebo těch starajících se o zpracování obrazu až k těm, jež jednotlivé podsystémy spojují a na základě všech posbíraných dat plánují další pohyby. To vše dělá robota robotem a nejen kamerou či manipulátorem. Základní strukturou autonomních robotů jsou následující podsystémy.

Základním subsystémem mobilního robotu je, jak ze samotného názvu vyplývá, systém pohybový nebo také akční. Ten robotu poskytuje možnost přesunu v prostoru a nebo se stará o jiné pohyblivé části robota, jako jsou chapadla, otočné držáky kamer či ultrazvukových senzorů a další. Pohyb v prostoru je možné zajistit mnoha způsoby. Základním dělením dle prostředí se dají mobilní roboty rozdělit do skupin létacích, plovacích, pozemních, kosmických a popřípadě i v různých kombinacích. Pohyb v jednotlivých kategoriích může být zajištěn opět více způsoby. U létacích robotů bychom jej mohli dělit například podle typu vzletu, kolmý nebo vodorovný, u robotů přizpůsobených vodnímu prostředí bychom asi dělili na ty, které se mohou ponořit, a ty, které jsou určeny k pohybu na hladině. U pozemních lze dělit na holonomní a neholonomní. Tedy jednoduše na ty, které se dovedou otočit na jednom místě, a ty, jež se musí při otáčení i jinak pohybovat prostorem. Dalším členěním je dělení z hlediska podvozku a typu pohybu na kolové, pásové, plazící, skákající, šplhající, kráčející a mnohé další kombinace a mutace. Dále lze dělit i podle stability

pohybu, například skákající robot má několik stabilních poloh, při nichž se dotýká podložky, a minimálně jednu nestabilní, při níž se nachází ve skoku, v němž se nemůže být stabilně a pohybuje se nejen v závislosti na počátečních podmínkách robota, ale i jeho prostředí. V závislosti na tom, do které kategorie robot spadá, se vyvíjí komplexnost akčního subsystému jak z hlediska hardwarového, tak i softwarového.

Kromě samotného systému pohybu je třeba zmínit i zpětné vazby vnitřního senzorického systému, které jsou k jeho správnému fungování potřebné. Bez zpětných vazeb o provedeném pohybu by roboty nebyly nikdy schopny dosáhnout požadovaného cíle. U teleoperovaných robotů se o část zpětných vazeb stará operátor, to samé platí i pro některé semiautonomní roboty. Některé zpětné vazby si zpracují samotné pohony díky vlastním regulačním smyčkám. Za pomoci matematického modelu a dat o natočení motorů je také možné dopočíst polohu, již robot urazil od startu. Avšak tento typ odometrie není zcela přesný a i v případě přesného měření je velmi náchylný na chyby, které se s časem integrují, nejčastěji chyby vzniklé zejména prokluzem mezi robotem a podložkou. Bez informace o stavu baterie by se zase mohly roboty dostat do situace, že už nebudou schopny se znovu nabít, nebo vrátit do své domovské stanice. Informace o teplotách motorů, řídicích jednotek a dalších součástí zabezpečují, že robot samotný se nestane nebezpečím pro své okolí nebo sám pro sebe. Data z většiny těchto senzorů tak slouží k sebediagnostice, tedy určení polohy a stavu robota.

Pro přesnější pohyb v prostoru jsou potřebná i další data, která nám poskytuje vnější senzorický podsystém. Ten může být složen z různých typů senzorů, jež jsou opět závislé i na prostředí, v němž se robot pohybuje. Asi do všech prostředí může robot využít kamery, které však samy o sobě ne vždy dávají plný rozsah informace o okolí robota. Proto jsou roboty nezřídkakdy doplněny senzory, které měří vzdálenosti v různých osách nebo plochách a zároveň jsou často využité senzory různých druhů, aby navzájem kompenzovaly své chyby. Obvykle se také využívají jednoduché kontaktní senzory v nějaké formě spínačů, běžně se umisťují pod nárazníky nebo třeba jako součást tykadel.

Autonomní roboty jsou sice vybaveny všemožnými senzory, ale aby procházely prostorem bez nebezpečí fatální srážky se svým okolím, musí být data ze senzorů zpracována a musí z nich být robot schopen rozhodnout, kudy se má vydat za svým cílem. O to se stará kognitivní subsystém, který za pomoci dat ze senzorického subsystému určuje své pracovní prostředí a v něm například i objekty, s nimiž může manipulovat, plánuje trasy v okolním prostředí a akce k dosažení zadaného cíle.

Přestože se roboty dokáží zorientovat a naplánovat si akce k tomu, aby cíle dosáhli, cíl svého konání si často samy zadat nedovedou. Tuhle možnost robotům lidstvo ještě neposkytlo, a tak schopnost samostatného rozhodování ve všech aspektech svého konání postrádají. Kupříkladu řidič autonomního vozu zadává cílovou trasu a jestli tam chce být rychle nebo pohodlně, robot (auto) zároveň do volby trasy může zahrnout i body významné pro něj, jako třeba čerpací stanice. Stejně tak pán domácnosti dává robotu povel, aby začal uklízet, ale trasu, kterou se k místu své činnosti přemístí, robot volí většinou sám, případně se přizpůsobí novému prostředí nebo novým povelům a následnou činnost koná také často na základě vlastního rozhodování. Dalším příkladem může být operátor dávající povel k vyskladnění kon-

krétního dílu, volbu odkud a jak ho vezme, si už řeší robot sám. A proto je potřeba předávat povely nebo úkoly i v autonomním režimu. O to se stará vnější komunikační subsystém, který může mít mnoho podob. V dnešní době je již možné robota ovládat hlasovými nebo pohybovými gesty, avšak stále mají převahu ty ovládané pomocí nějaké fyzické formy dálkového ovladače. Může se jednat o klasické dálkové ovladače s jednosměrnou komunikací, ale i obousměrně komunikující ovladače, například v podobě počítače komunikujícího po drátové nebo bezdrátové sběrnici, kde probíhá výměna povelů a zpětné vazby, které mohou být zobrazeny v uživatelském rozhraní.

Všechny zmíněné subsystémy navzájem komunikují a společnými silami dovolují robotu splnit požadavek operátora, jenž může robotovu činnost sledovat z dálky, a to jak již víme i opravdu veliké, přesahující hranice naší planety a sahající v aktuální situaci dokonce i za hranici sluneční soustavy, přesněji do vzdálenosti $24,23 \cdot 10^9$ km, k vesmírné sondě Voyager 1, jež je zatím od Země nejvzdálenějším člověkem vytvořeným tělesem. Cesta signálu k samotné sondě trvá už více než 22 hodin, na odpověď si tedy operátor počká dvakrát tak dlouho. Kdyby sonda neměla autonomní režim, jen těžko by docestovala tak daleko.

2 Hexor® II

Robot Hexor® II byl navržen a vytvořen ve spolupráci polské Silesian University of Technology a Stenzel Ltd. Jedná se o platformu navrženou pro účely výuky. Studenti mají možnost se na ní učit nebo testovat své schopnosti v programování, zkoušet hardwarová vylepšení, jako jsou nové senzory a jejich využití v řízení. Samotní tvůrci dávali návrhy na možné práce.

Obrázek 2.1: Hexor® II, vzhled robota [13]

Hexor® II se řadí do kategorie holonomních chodících robotů a samotný pohyb je zajištěn šesticí nohou dohromady poháněných trojicí servomotorů. Každá dvojice nohou je poháněna jedním servopohonem. Mechanikou jsou svázány po dvojicích levá přední a zadní noha, pravá přední a zadní noha a levá a pravá střední noha. Střední nohy robota nadzdvihají, čímž mu dovolují pohybovat pravou nebo levou stranou nad podložkou. Zbylé nohy se pohybují jen v rovině podložky. Robotův

pohyb je díky této konstrukci dynamicky stabilní, jelikož v každém okamžiku svého pohybu má alespoň tři opěrné body. To ho ovšem omezuje z hlediska terénu, není vhodný do hrbolatých prostředí, ale spíše na rovinu, takže je určen primárně do vnitřních prostorů. Hexor® II má zároveň ještě jednu pohyblivou část, jíž je ocas, jehož konec má dva stupně volnosti a dokáže se otáčet kolem svislé a vodorovné osy.

Robot zkoumá svět kolem sebe dvojicí tykadel, která jej informují o přímém kontaktu s překážkami. Je osazen infračervenými senzory na měření vzdálenosti, jedním vpředu a čtyřmi vzadu, avšak jejich dosah je jen do asi 30 cm, v závislosti na světelných podmínkách a odrazivosti předmětů. A na ocasu si nese ultrazvukový senzor, kterým může zkoumat okolí ve větší vzdálenosti do 3 m s přesností na 1 cm v ideálních podmínkách. Společně s ultrazvukovým senzorem je na ocasu umístěna i CCD kamera pootočená o 90° kolem horizontální osy, jež není přímo spojena s řídicí deskou robota, takže zpracování dat, která posílá, je třeba provést na samostatném zařízení.

Ani jeden ze zmíněných senzorů není úplně bez slabiny. U hmatových tykadel je slabé místo převážně v jejich délce, kvůli níž sice robot ví o kontaktu s překážkou a na které polovině robota k ní došlo, ale přesnější informaci o směru a vzdálenosti od středu robota nám neposkytne. Ultrazvukový senzor má dva hlavní nedostatky, tím prvním je samotné prostředí, poněvadž některé materiály pohlcují nebo tlumí zvukové vlny, jež ultrazvukový senzor vysílá, a tak narušují jeho přesnost. Druhým slabým místem je vysílací úhel, jenž omezuje pozorovací rozlišení. Slabinou infračerveného senzoru jsou zrcadla a materiály pohlcující nebo tlumící světlo, ale to jsou nevýhody, s nimiž se zase moc často nesetkáme, a v případě že ano, tak se poměrně dobře kompenzují za pomoci ultrazvukového senzoru, takže vzdálenost přímo před robotem můžeme znát poměrně přesně.

Největší nevýhodou kamery je způsob, jímž posílá svá data, a to, že jich bývá poměrně velké množství. Jelikož data nedostává přímo robot, bude do budoucna vhodné ji buď předělat, aby data předávala přímo, nebo vytvořit modul, který data přijme, zpracuje a předá do systému v jiné podobě. I s novým mikroprocesorem však nejspíše přejde všechna práce na nějaké jiné zařízení, protože řídicí mikroprocesor by byl zpracováváním dat z kamery dost zatížen.

U Hexora® II je komunikace zajištěna digitálním vysílačem pracujícím na frekvenci 433 MHz s přenosovou rychlostí 9600 Bd. Ten komunikuje s protějškem připojeným k počítači, kde je ovládací rozhraní zobrazeno v samostatném programu. Jedním z cílů této práce je komunikaci rozšířit o další komunikační kanál – o Wi-Fi s webovým klientem, kde by se kromě například datalogování mohlo jednat i o druhý přístupový bod k robotu, a tak by mohl případně dostávat povely odtud, a to nejen od lidských operátorů, ale i od jiných robotů, s nimiž bude připojen k internetové síti. Zařízení připojené k robotu by zároveň mohlo zpracovávat data z kamery nebo je jen zobrazovat.

Hlavní snahou této práce bude kromě přechodu řídicího systému na novější typ mikroprocesoru i přechod na systém, který by nepotřeboval speciální hardware. Tedy v ideálním případě by mělo být možné se k robotu připojit pomocí jakékoliv chytré technologie, jež podporuje Wi-Fi a přístup do webového rozhraní.

3 Řídicí systém

Oproti předchozímu řešení řízení robota je tentokrát snaha dostat co největší část řízení přímo do robota, zároveň by měla zůstat zachována kompatibilita s původním řešením. Jmenovitě se jedná o kompatibilitu s nadřazeným systémem ve formě PC a na něm běžící aplikace. Díky zvolenému novému komunikačnímu čipu ESP32 bude možné většinu řídicího programu uschovat přímo na čipu, který bude neustále součástí robota. Je však velmi pravděpodobné, že skutečné výpočty pro povely a samotné řízení robota zůstane plně v rukách připojeného zařízení. Je tedy možné, že robota nebude možné ovládat, pokud nebude připojen k ovládacímu zařízení, respektive pokud dojde k odpojení, robot nebude vykonávat většinu své autonomity. Toto ovládací zařízení však již bude moci být i chytrý telefon, díky čemuž by robot ve své podstatě byl obohacen o ovládací panel přímo na svém těle. Telefon by totiž mohl být upevněn na vnějším plášti a pomocí webové aplikace by po zvolení módu mohl být více samostatný než doposud.

Předpokladem této práce je náhrada staré řídicí desky za novou. Aby tedy bylo možné se nadále pohybovat v prostoru, bude potřeba vyvinout program, který bude přijímat povely pro pohyb nebo pohyb plánovat, měnit mód ovládání a reakcí na povely, sledovat svůj pohyb prostorem a zároveň prostor mapovat a interpretovat operátorovi. Pro tvorbu map a následné hledání tras je jistě mnoho metod. V této práci budou zmíněny především dvě hlavní.

3.1 Teorie k tvorbě map a hledání trasy

Snahou většiny autonomních robotů bývá sestavit si plán prostředí, v němž se pohybují. Pokud tedy nechceme, aby robot jednal čistě na základě aktuálních surových dat ze senzorů, budeme muset vytvořit nějakou formu plánu prostředí. Mapy můžeme podle jejich složitosti a způsobu tvorby rozdělit na geometrické, topologické a sémantické.

3.1.1 Typy map

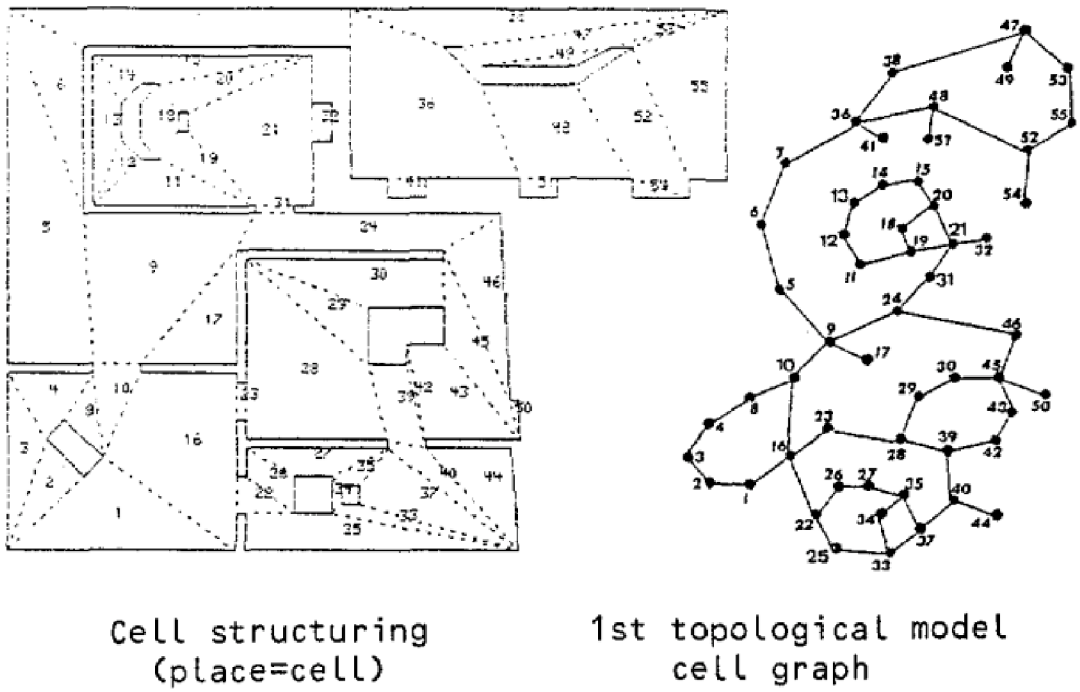
Geometrické mapy

Geometrické mapy jsou přímou geometrickou reprezentací dat naměřených pro určitou oblast pomocí senzorů a uložených v paměti. V případě robotů, které se stejně jako ten v této práci dokáží pohybovat primárně jen po rovině, lze tyto mapy brát

jako průmět světa v okolí robota do podložky. V případě, kdy by byla u robota naprosto dokonalá odometrie, by se jednalo o velice snadný a veskrze bezproblémový přístup. Pomocí lidaru, sonaru, radaru nebo jiného měřicího senzoru se změří vzdálenost k překážkám a na základě změřené vzdálenosti a úhlu natočení měřícího senzoru se do mapy zapíší překážky v okolí. Průchodem skrze celé prostředí vznikne geometrická mapa. Avšak u robotů nelze docílit dokonalé odometrie, takže je nutné před zápisem do mapy buď chybu v odometrii kompenzovat a dopočítávat reálnou polohu z robotova okolí, nebo udržovat jen malou mapu a tu přepisovat novými daty i v případě, že se dostáváme na stejné místo jen z jiného směru. V našem případě se tedy budeme pohybovat v mapách o rozměrech asi 10×10 metrů ($10m^2$), nebo také 100×100 pixelů, kde každý pixel má rozměr 10 cm. I v takto značně zaokrouhleném prostředí, a možná i právě proto, bude docházet k nepřesnostem jak z hlediska vyznačení překážek, tak z hlediska polohy robota v prostředí.

Topologické mapy

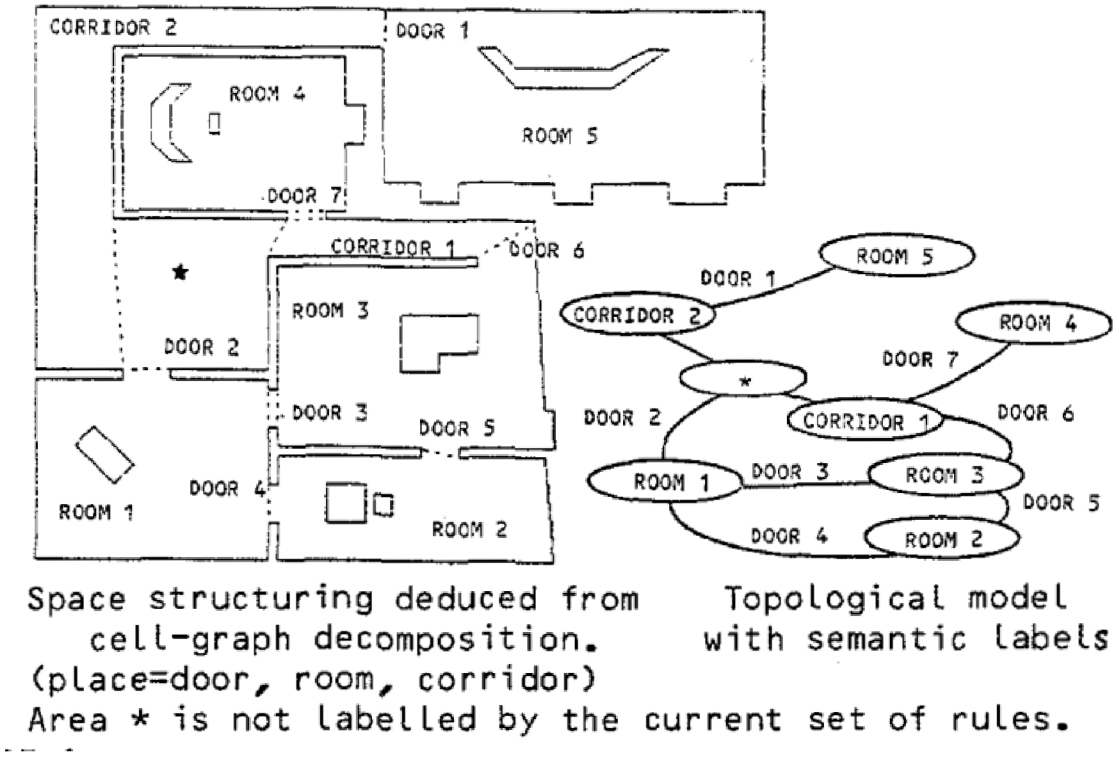
Topologické mapy mohou být vytvořeny po přezkoumání geometrických map. Rozdělením vnitřního prostředí geometrické mapy na jednotlivé buňky jde například určit přechody mezi místnostmi. Zároveň tím lze jasně rozdělit překážky v jednotlivých místnostech a hranice jednotlivých místností, tedy zdi. Toto přezkoumání geometrické mapy tedy prozradí jakousi síť spojující jednotlivé buňky prostoru. Díky topologii prostředí je následně možné se snáze rozhodnout jakou cestu k cíli zvolit, nebo jako cíl zvolit přímo část topologie.



Obrázek 3.1: Náhled rozšíření geometrické mapy na topologickou [6]

Sémantické mapy

Sémantická mapa je výsledkem přezkoumání topologické mapy. Sémantická mapa dodává smysl oběma předcházejícím. Spojením jednotlivých buněk v topologii dokáže rozlišit místnost od chodby či dveří. Následně nám tedy poskytne logické pochopení jednotlivých místností, průchodů (dveří), chodeb a jejich vzájemné provázání. Tedy které dveře nebo chodby spojují konkrétní místnosti a které místnosti spolu sousedí. Ve své silnější formě může například i poskytnout informaci o účelu daných místností na základě analýzy dat z kamer nebo jiných vlastností místnosti, jako například že chodby jsou úzkým prostorem, jenž často spojuje jiné místnosti. Zároveň se dá říct, že například obývací pokoj bude jednou z největších místností na mapě.



Obrázek 3.2: Náhled rozšíření topologické mapy na sémantickou [6]

Tak daleko ale bohužel v této práci nezajdeme a víceméně zůstaneme jen u geometrických map, v nichž budou vykresleny překážky, trasa, robot, start, cíl a body na zlomech kolem překážek, tedy často jejich rohy.

Díky novému mikropočítači získáme možnost zvýšit rozlišení a na menší ploše by snad mohl být robot plně autonomní. Zároveň by mělo stačit zadat polohu cíle. Trasa by se poté vypočítala přímo u robota a ne u operátora, respektive u zadavatele cíle. V případě větších prostorových celků by se o trasu opět staral operátor, respektive zařízení s vyšším výkonem v master-slave režimu. Nebo je možnost, jak již bylo zmíněno, zanechat výpočty na straně chytrého zařízení, které bude-li na to požadavek, může být oddělitelnou součástí těla robota. Ponechání všech výpočtů

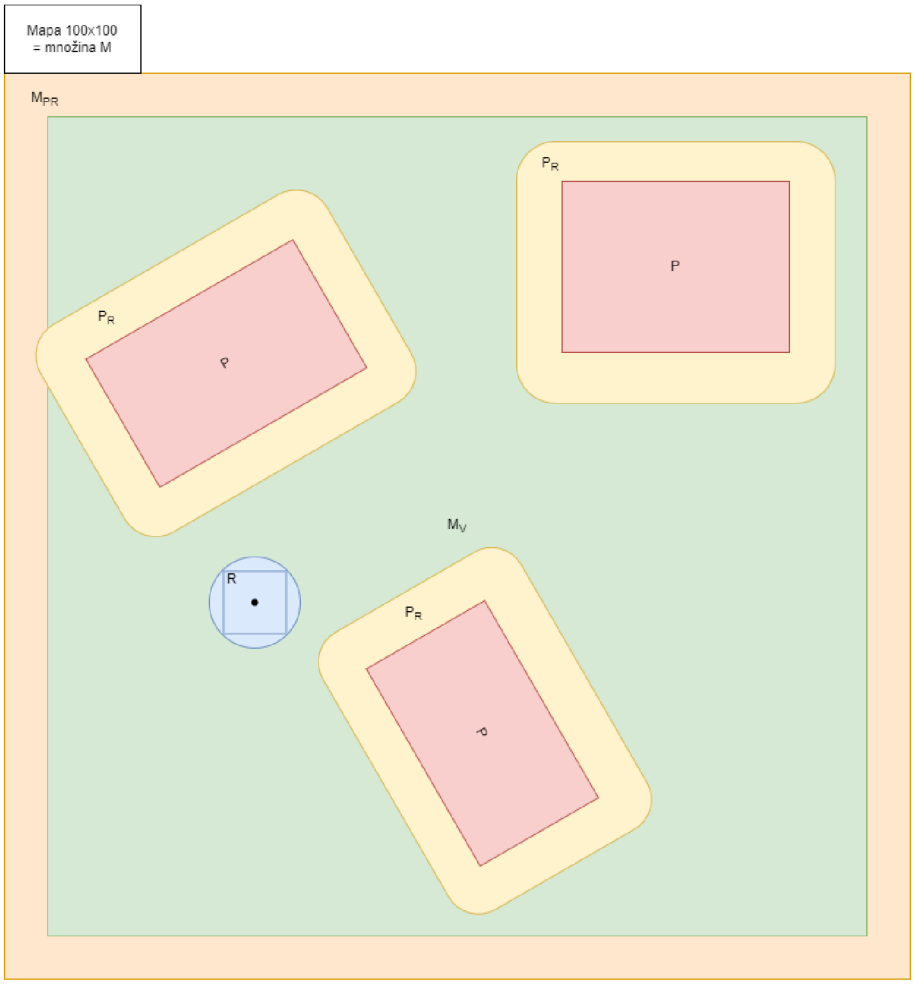
na pouze jednom zařízení sníží složitost a případnou nespolehlivost z hlediska programu, jelikož i malé prostorové celky mohou obsahovat větší množství překážek, což vede ke složitým výsledným trasám, jejichž řešení by mohlo být v robotu příliš náročné.

3.1.2 Pracovní prostor robota (Configuration space)

Tvorba geometrické mapy nám vyřeší prostor okolí robota z hlediska rozmístění překážek v jakékoliv formě. Samotné překážky nám ale nestačí k tomu, abychom mohli určit kam, se robot dostane a kam ne. K tomu slouží tzv. pracovní prostor. Určení pracovního prostoru vychází z geometrických vlastností robota a geometrických vlastností konfiguračního prostoru. Celý problém lze popsat pomocí množin. Množina M je naše mapa, překážky P jsou součástí M a naším cílem je najít prostor, kterým může projít robot R tak, aby byl součástí M, ale nikdy nebyl součástí P.

$$R, P \in M \wedge R \cap P = \emptyset \tag{3.1}$$

Body (pixely), respektive spojitá množina splňující tuto podmínku, je tzv. bezpečný prostor, tedy pro robota množina bodů, kde se robot nachází mimo překážky, maximálně v dotyku s nimi. Pokud chceme problém a jeho řešení zjednodušit, můžeme simplifikovat tvar robota pro 2D mapu na polygon. Na polygonu, který reprezentuje robota, zvolíme bod, ideálně těžiště, pro složitější tvary robota se může vyplatit i bod jiný. Následně vezmeme námi stanovený polygon a překážku rozšíříme o prostor, jenž je při kontaktu polygonu a překážky mezi zvoleným bodem a překážkou, tento prostor je vázaný na natočení polygonu vůči stanovenému bodu [9]. Pro ještě větší zjednodušení lze vzít místo polygonu reprezentujícího robota kružnici a jako bod si zvolit její střed, stále tedy těžiště. Překážky se touto transformací následně rozšíří ve všech směrech o poloměr kružnice, které je robot nebo jeho zjednodušený polygon vepsaný. Po rozšíření prostoru překážek o rozměry robota je možné robota reprezentovat jako bod a všechny body množiny M, které nejsou překážky, jsou body, kde se může nacházet robot. Tedy opět v podobě množiny: $M_V = M - (P + P_R)$, kde M_V reprezentuje volné místo v mapě.



Obrázek 3.3: Znázornění pracovního prostoru z pohledu množin

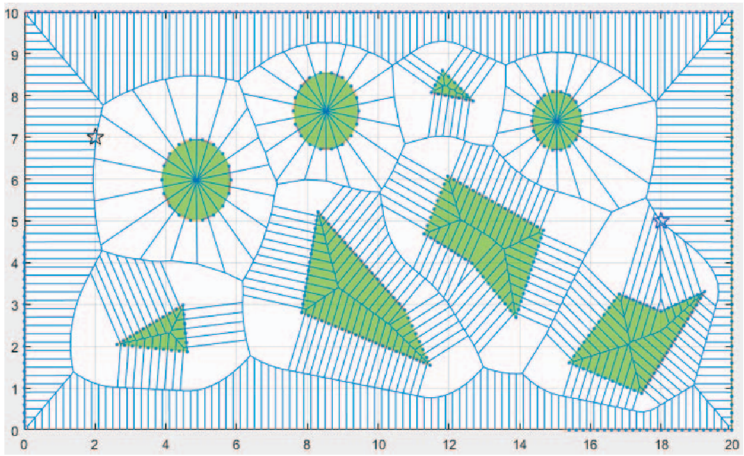
Žlutá část P_R reprezentuje množinu, o kterou je rozšířena překážka P. Robot R může být reprezentován bodem, jenž je vyznačen v jeho středu. Zelená množina M_V je tedy prostorem, ve kterém by se měl robot pohybovat beze srážky s překážkami P. V případě, že kraje mapy jsou zdi, oranžová oblast M_{PR} je další množinou, o niž bude pracovní prostor robota M_V zmenšen. V opačném případě, pokud kraj mapy neodpovídá fyzické bariéře, je $M_V = M_V + M_{PR}$.

3.1.3 Hledání trasy

Robot, kterým se tato práce zabývá, se řadí mezi holonomní roboty, roboty schopné se otáčet bez nutnosti dalšího pohybu v prostoru, není potřeba do hledání trasy zanášet dynamiku jeho pohybu. V podstatě stačí najít jakoukoliv cestu prostorem z bodu A (bod aktuální pozice robota) do bodu B (bod cíle cesty). K tomu lze využít mnoho metod, zde jsou některé z nich.

Voronoi diagram

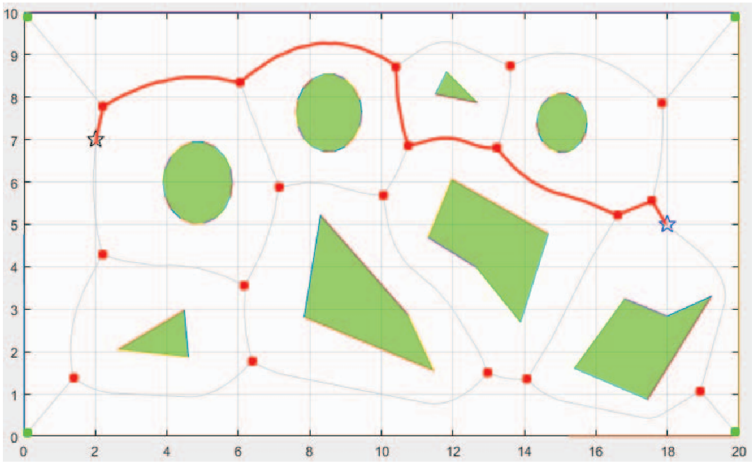
Jednou z metod hledání ideálních tras prostorem je tzv. Voronoiův diagram. Části tras, které v rámci tohoto diagramu vzniknou, jsou trasy ve středu prostoru mezi překážkami. Trasy takto vzniklé nemusejí být, a pravděpodobně ani nebudou, vždy nejkratší možnou trasou mezi body startu a cíle. Stejně jako i v dalších metodách je možné a často i nutné je následně zjednodušovat. Avšak trasa takto vzniklá vede nejbezpečnější cestou na střední vzdálenosti od překážek. Voronoiův diagram vzniká tak, že pomocí interakcí mezi dvěma body, bodem a přímkou, či dvěma přímkami vytváříme střed kolmých vzdáleností. Toho lze dosáhnout tím, že mezi interagující prvky vykreslíme tečnou kružnici, jejíž střed je právě střed kolmých vzdáleností. Interakcí mezi dvěma body je tedy bod, mezi bodem a úsečkou či přímkou je to křivka a mezi dvěma přímkami/úsečkami je to opět přímka/úsečka. Pro tvorbu Voronoiova diagramu není nutné rozšiřovat překážky o rozměry robota. Průchodu mezi překážkami je možné zamezit i v závislosti na vzdálenosti trasy od překážek, a tak je možné některé křivky při následném hledání trasy ignorovat nebo je vůbec nevykreslit, pokud omezíme nejmenší možný průměr výše zmíněné kružnice. Rozšířením překážek a zmenšením robota na jeden bod zajistíme, že pokud se překážky přímo nedotýkají, jsou všechny křivky Voronoiova diagramu schůdné trasy.



Obrázek 3.4: Ukázka tvorby křivek Voronoiova diagramu mapy [10]

Volba následné trasy by pak mohla probíhat zjednodušeně tak, že pokud robot neleží přímo na křivce, pak se po nejkratší trase nejlépe po přímce přesune na vzniklou křivku trasy mezi překážkami, po níž se dopraví k cíli, kterého dosáhne, pokud není přímo na křivce, opět po nejkratší možné trase (přímce). Pro použití v mikroprocesoru je tedy tato metoda značně nevýhodná, především v tom, že pokud nepřepočítáme mapu na vektorový popis jednotlivých překážek, musíme provést interakci každého bodu obrysu překážky s každým dalším bodem jiné překážky. V takovém případě bychom pravděpodobně značně zahltili mikroprocesor pouze snahou dopočítat trasy, avšak jako geometrické řešení průchodu prostorem v bezpečných vzdálenostech od překážek může být vhodná a při dostatku výpočetního výkonu

a dobré mapě i výhodnější než metoda následující.



Obrázek 3.5: Ukázka trasy vzniklé z Voronoiova diagramu [10]

Pracovní prostor - brute force

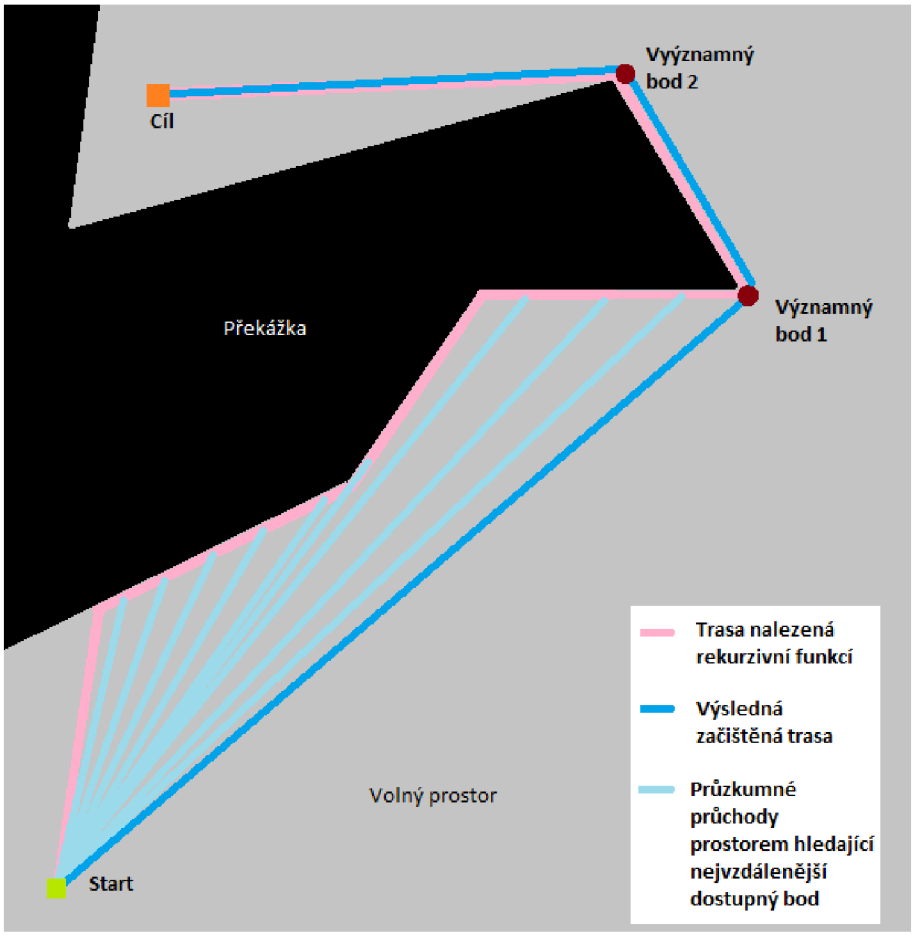
Více méně přímá metoda pracovního prostoru je založena na již zmíněném principu tvorby mapy. Ideálně po nejkratší možné trase ve vytvořeném pracovním prostoru se robot přesouvá z výchozí do cílové pozice.

Robot je tedy zjednodušen do podoby bodu, jeho trasa může vést těsně podél překážek, které jsou virtuálně rozšířeny do pracovního prostoru o plochu, kterou by potřeboval k jejich objetí v případě, že by nebyl zjednodušen na bod. Oproti Voronoiově diagramu není potřeba pokaždé, když se při procházení prostoru objeví nová překážka, přepočítat celou mapu, ale jen znovu dopočítat trasu. Logika hledání trasy je poměrně jednoduchá, snažíme se jít přímo k cíli, pro řešení na papír jdeme ve směru k cíli a zároveň rovnou obcházíme překážky cestou, která by měla být kratší. V případě, že stejný úkol zadáme počítači, musíme dojít k překážce, tu obejít buď ve směru, nebo v protisměru hodinových ručiček a ve chvíli, kdy v cestě nepřekáží žádná překážka, jdeme zase přímo k cíli.

Díky větší jednoduchosti na výpočty byla zvolena metoda navazující na pracovní prostor a procházení pole pomocí rekurzivní funkce. Ta se snaží projít trasu z bodu A do bodu B co možná nejpřímější cestou, tedy po směrovém vektoru $\overrightarrow{AB}$. Pokud narazí na překážku v mapě, zkouší ji obejít po nebo proti směru hodinových ručiček střídavě v dalších sedmi různých směrech pootočených o 45° proti směrovému vektoru $\overrightarrow{AB}$. Směr, který dosáhne cíle s nejkratší trasou, je vrácen jako trasa z bodu A do bodu B. Tu si uložíme jako pole bodů trasy W (3.2) s délkou k. Pokud se stane, že trasa cíle nedosáhne, zavolá se hledání trasy v opačném směru. Pokud ani v této konfiguraci nedosáhneme cíle, pak je cíl nebo robot plně uvězněn, tedy je cíl nedostupný.

Podobnou metodu využívá i původní program pro počítač. Pomocí rekurzivní funkce se snaží dojít do cíle. To však bylo zjištěno až později a funkce se tak od sebe možná i zbytečně liší. Avšak v nové funkci bylo nezávisle přidáno ještě pár výhod.

Tou první je více směrů, jimiž se robot může vydat, změna ze čtyř směrů na osm. Druhou změnou je začištění nalezené trasy vrácené rekurzivní funkcí. Toto začištění zajistí, že se robot nebude pohybovat neustále v kontaktu se zdmi a překážkami, ale po přímých trasách mezi nejvzdálenějšími možnými po úsečkách dosažitelnými body trasy. Z čehož vzešlo i hledání trasy po úhlopříčkách, které v původní metodě bylo pravděpodobně nevhodné, protože by se robot musel vždy otáčet o 90°. Robot by měl být schopný natočit se i o jiný úhel, takže nyní by se měl natáčet novým směrem a pak půjde k rohu překážky (k význačnému bodu).



Obrázek 3.6: Ukázka logiky začišťovací funkce

Tohoto začištění lze dosáhnout prostým vnořením dvou, respektive tří for cyklů, z nichž jedním vybereme bod S pro start vektoru a druhým projdeme všechny následující body trasy. Postupně projdeme pole nebo také matici trasy W 3.2 tak, že třetím for cyklem procházíme mezi bodem S – start (nový) a G – goal původní trasy, dokud nenarazíme na překážku nebo nedojdeme do cíle. Pro $max(|\vec{v}|)$ 3.3 si úsečku mapy mezi S a G uložíme a před změnou n si je zapíšeme jako nové body trasy W od n do i. Následně do n přepíšeme hodnotu posledního G a celý cyklus končí, když $n \geq k-2$. Jednotlivé G můžeme zároveň ukládat jako stěžejní/význačné body trasy. Tedy v podstatě subgoaly nebo kontrolní body, do jejichž okolí se snaží robot

dostat, protože se mezi nimi dokáže pohybovat v podstatě po přímce, respektive po úsečce.

$$W_{2\times k} = \begin{bmatrix} X_1 & X_2 & \cdots & X_k \\ Y_1 & Y_2 & \cdots & Y_k \end{bmatrix} \tag{3.2}$$

$$\begin{gathered} S = (X_n, Y_n) \\ G = (X_i, Y_i) \\ \vec{v} = \overrightarrow{SG} \\ \text{pro } n \in (0, k);\ i \in (n+2, k\rangle; \end{gathered} \tag{3.3}$$

3.2 Simulace zvolené a navržené metody hledání a začištění trasy

Princip metody je poměrně jednoduchý, ale debug rekurzivní funkce přímo na procesoru robota, kde jsou zatím data mapy reprezentována jako holá čísla, respektive jako vektor, případně registr nul a jedniček reprezentující obsazenost jednotlivých polí mapy, je nepřehledný. Proto před aplikací této funkce do řídicí jednotky robota proběhla simulace v Matlabu, který podporuje zobrazování bitmap, zároveň jde program krokovat. Díky tomu by měla být výsledná funkce poměrně robustní pro přepis do c/c++ pro ESP32 nebo do jiného jazyka.

Díky možnostem Matlabu a použití počítače jde víceméně odsimulovat celé chování rekurzivní funkce trasování i funkce zjednodušení trasy na větším množství mapových dat, které se vloží ve formě bitmap.

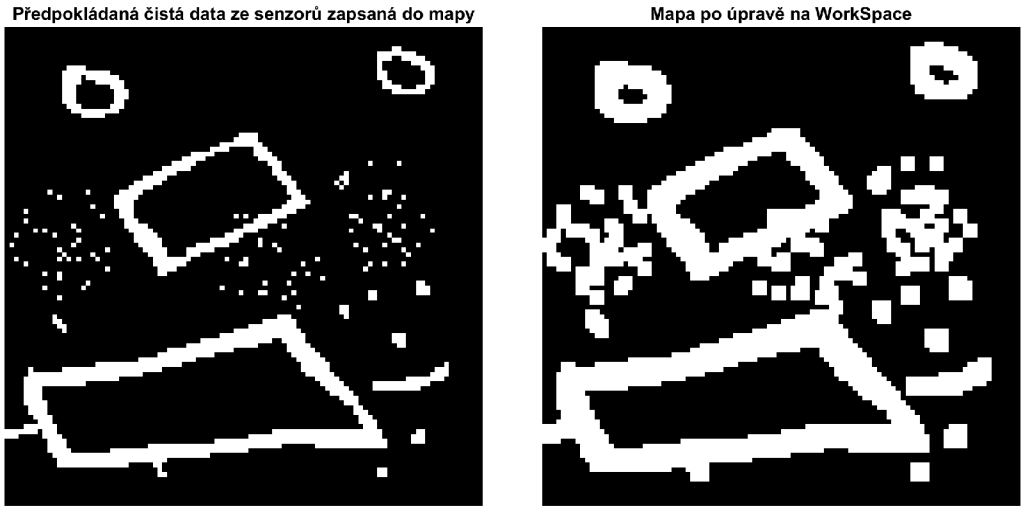
Zároveň tak bylo možné vyzkoušet i to, jaká velikost mapy bude ještě pro robota vhodná z hlediska výpočetního výkonu a místa v paměti řídícího čipu při zachování rozlišení z původního programu, tedy že pixel mapy $\equiv$ 20 cm. Pak jsme schopni bez problému do paměti dostat mapu o rozměrech 200 $\times$ 200 metrů, pokud bude v čistě binární podobě (0 = volno; 1 = překážka), což může být až zbytečně veliké, teoreticky by mělo stačit i mnohem méně. Pokud zlepšíme rozlišení na 10 cm, výsledná mapa bude mít 100 m^2. To je pořád poměrně velká plocha a zároveň získáme dvojnásobné rozlišení oproti původnímu řešení.

3.2.1 Postup vývoje v Matlabu

Prvním krokem při tvorbě funkce bylo navržení mapy. Data z robotu se budou lišit, ale musíme počítat s tím, že najít trasu se budeme snažit jak v zaplněné, tak prázdné mapě. Byla tedy navržena mapa, která splňuje jednotlivé požadavky, obsahuje dostatek míst s překážkami i místa, kde se dokážeme dostatečně daleko dostat i volným prostorem. Tato mapa byla během testování mnohokrát vylepšena a změněna. Zároveň se robot asi bude častěji pohybovat v prostorách menších než 100 m^2, takže bylo zvoleno měřítko mapy desetkrát menší, ale interní test funkcí

proběhl i na mapě o plných rozměrech s cestami jak krátkými, tak i těmi přes celou mapu.

Dalším krokem bylo zvolenou mapu upravit tak, jak to bude muset dělat sám robot v případě, kdy načítá překážku a musí ji rozšířit podle pravidla, které jsme si již popsali výše 3.1.2. Ve zkratce musí detekovanou překážku rozšířit o rozměr, který mu dovolí v mapě svým těžištěm chodit těsně vedle překážek, ale ne skrze ně.



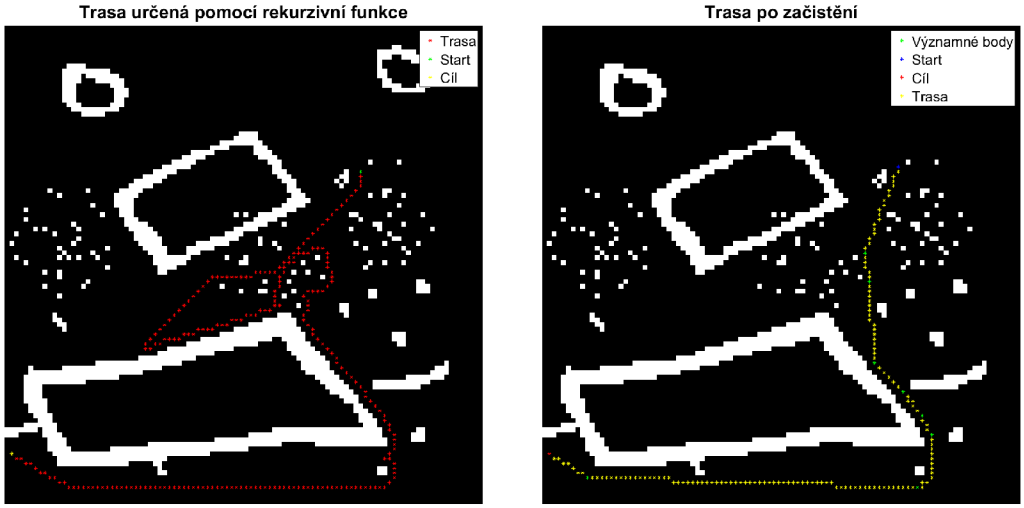
Obrázek 3.7: Změna čistých mapových dat na mapu pracovního prostoru robota

V takto vytvořeném pracovním prostoru se robot bude snažit najít trasu. V prvním pokusu o hledání trasy bylo funkci dovoleno pokračovat téměř neomezeně. Jediným limitem byl hardware počítače a interní omezení Matlabu na využití prostředků. Při těchto pokusech bylo zjištěno, že vhodným řešením pro hledání trasy by bylo vnést do hledání větvení a výběr směru obcházení překážky jen v bodech kdy trasovací funkce dorazí k překážce a ne v každém bodě, kdy nemůžeme jít přímo k cíli.

Detekce nové překážky je zajištěna jen tak, že trasovací funkce na cestě k cíli opustí obrys překážky a projde volným prostorem mapy. Tento způsob řešení je zvolen převážně proto, že u robota bude potřeba v těchto případech šetřit datovým prostorem, a tedy nepůjde například využít funkcí obrazové analýzy a jednotlivé překážky přímo v mapě číslovat. Šlo by však například převést mapu na matematický popis jednotlivých překážek a z toho zjistit, které oblasti jsou souvislými překážkami, avšak tomu se zatím věnovat nebudeme, protože dosavadní detekce opuštění oblasti překážky pomocí kontroly okolí kroku trasovací funkce byla dostatečná. Navíc by takováto mapa měla násobně větší velikost. A bohužel jsme stále u mikročipů omezováni velikostí paměti.

Funkce začistění trasy se následně ukázala jako poměrně jednoduchý, ale zároveň dost silný nástroj. Když nedojde ke skutečně zvláštním situacím, dokáže trasu robota zkrátit o velký počet kroků. Algoritmus hledání trasy vstoupil do slepé uličky pouze

proto, že snažil projít přímou cestou k cíli. Ale jelikož je tato cesta slepá, vrací se nakonec zpět, a tak je zjednodušující funkce schopna toto slepé rameno trasy úplně vynechat.



Obrázek 3.8: Porovnání trasy vygenerované rekurzivní funkcí a trasy po začistění v Matlabu

Bohužel jako většina funkcí i kombinace hledání trasy a její začištění mají svá slabá místa, avšak i tak jejich spojením často získáme spíše kratší, přímější nebo celkově elegantnější trasu než jen ze samotné rekurzivní funkce. Teoreticky by se dalo přidat omezení, respektive podmínka, že bude vybrána kratší trasa i v porovnání před a po začisťující úpravě.

Při větvení vznikají i situace, kdy se rekurzivní funkce zasekne v samostatném cyklu, a tak trvá hledání trasy v Matlabu příliš dlouho. Z tohoto důvodu byla a ještě budou přidána dodatečná omezení, která jsou v prostředí Matlabu závislá jen na vzdálenostech, krocích, které může rekurzivní funkce udělat. Interně si tak rekurzivní funkce předává počet kroků, které prošla od posledního restartu vzdáleností. Restart uražené vzdálenosti je proveden v případě, že se funkce vydá po trase volným prostorem, v němž nejsou detekovány překážky ani značky, které si za sebou tvoří funkce sama, aby nechodila v přílišných kruzích, když se snaží jít po směrovém vektoru k cíli. U procesoru robota bude omezení i z hlediska procesorového času. Čas potřebný k nalezení trasy v procesoru robota bude ještě nutné samostatně ověřit, jelikož trasu nebude vykreslovat, měl by být násobně kratší než v Matlabovské simulaci.

Omezení na vzdálenosti při pokusech několikrát odhalilo různé cesty. Byla odhalena závislost nejen na počátečních podmínkách z hlediska polohy, ale i v závislosti na omezení trasy podél jednotlivých překážek. Některé požadované trasy, například i s nízkou vzdáleností, nejsou schopny dorazit do cíle, protože uvíznou ve smyčce, ale s ještě nižší vzdáleností trasu najdou. Nebo naopak sice dorazí do cíle, ale po delší

trase, než jakou by zvolil operátor podle vlastního úsudku. Nejspíše bude hledání trasy u robota implementováno v lehce pozměněné variantě, hledání trasy bude voláno s různými hodnotami volby omezení délky bludného pohybu od horního limitu po spodní, zároveň i s časovým omezením na dobu výpočtu trasy. Výsledná vypočtená trasa bude určena v závislosti na délce, tedy nejkratší možná, a až poté bude uplatněna funkce na začistění.

4 Aplikace simulované funkce

4.1 Mapa

Jedním z hlavních důvodů, proč byla funkce nejdříve vyvíjena v Matlabu, bylo, že v jiných prostředích je nutné nejdříve vymyslet a vytvořit prostor, ve kterém budeme mapu a průchod skrze ni zobrazovat. Zatímco v Matlabu se nám jednoduchým příkazem mapa vykreslí jako obrázek, v němž není problém přibližovat, ve většině jiných prostředích musíme tyto funkce nejdříve navrhnout. Ať už by byla funkce hledání trasy implementována kdekoliv, chceme ji operátorovi zobrazit, a jelikož je cílem, aby nemusel stahovat a instalovat aplikaci do chytrých zařízení či počítače, vydáme se směrem webového prohlížeče. Nová komunikační deska ESP32 podporuje vlastní webserver, na němž bude moci tento ovládací web distribuovat operátorovi přes prohlížeč. Webový prohlížeč je navíc téměř v každém chytrém zařízení a jediné, co bude operátor potřebovat, je najít si adresu robota v síti, do níž se připojí, a připojit se k jeho webserveru. ESP32 si snad ve své paměti ponese vše potřebné k obsluze webového serveru, jenž je přístupný přes připojení k Wi-Fi síti vybrané operátorem nebo přes vlastní přístupový bod, kde je odpojen od zbytku internetu. V obou případech distribuuje stejnou ovládací stránku s mapou a povelovými tlačítky.

Na vykreslování mapy byl vybrán prvek canvas, jenž je schopen zobrazit obrázky nebo i jednotlivé body či tvary, které programátor zadá. K vykreslení mapy se tedy z jednotlivých „pixelů“ vyskládá celková podoba prostředí. Operátorovi by se zároveň mohly hodit funkce jako zoom, posun v mapě, vybrání cíle trasy robota a určitě i ruční zadání překážky, pokud bude chtít dodat i překážku, kterou robot nedetekuje nebo jež ho má omezit v pohybu.

Každou z těchto funkcí je potřeba vytvořit samostatně, protože nic takového přímo implementováno není. Veškeré operace a eventy vstupů zařízení, na kterém webová stránka běží, je možné si nastudovat a podle toho je aplikovat a reagovat na ně nebo s možnostmi, jež se za poslední dobu dostávají do oběhu a mohou být velmi nápomocné, tyto funkce jen nechat vygenerovat nějakou z programovacích AI. Bylo by skoro proti vysokoškolskému bádání neotestovat, co tyto umělé inteligence zvládnou a jak moc jsou vlastně schopné. Podle všeho si chatGPT (verze 3.5) měl s tímto problémem snadno poradit a tak byl jeho směrem vznesen požadavek na funkce zoom a posouvání. Výstup, který vygeneroval byl, krásně strukturovanou a komentovanou funkcí, ale byla nedokonalá a vůbec nefungovala. Nejen že ani vzdáleně nekopírovala pohyb kurzoru tak, jak by si operátor představoval, ale ani jednoduchý

zoom nebyl ideálně pojat. Kostra funkcí byla zachována v podobě, kterou chatGPT vygeneroval, musela však stejně být dodána matematika, jež se postará o správné reakce, které jsou svázány se souřadným systémem canvasu.

Pro pochopení jednotlivých operací je třeba dodat způsob vykreslování mapy. Každý jednotlivý pixel mapy je vykreslován jako čtverec o velikosti $x * zoomfaktor$ a posunutý v závislosti na zoom faktoru a poloze v mapě, tak aby výsledek působil jako klasický obrázek.

Problémem bylo, že chatGPT není schopen chápat a přizpůsobit matematické operace potřebné k posunutí souřadných systémů, přestože mu byl tento návrh předán. A tak se při operaci posouvání mezery posouvaly takřka náhodně a při přibližování/oddalování se jen zvětšovaly pixely. To bylo samo o sobě jednoduché vyřešit – při posunu je nutné ke všem "pixelům" přičítat počet pixelů, o které se posune kurzor za dobu, kdy obrazem posouváme. Při přibližování a oddalování je nezbytné "pixely" ještě posouvat tak, aby se nepřekrývaly. Z dat, které nám předává event posunu kurzorem budeme brát souřadnice x a y a zároveň si uložíme pozici, v níž samotné posouvání započalo.

Po opravě této části bylo přistoupeno k funkci zvětšení a zmenšení, zde se jedná o podobnou hříčku matematiky, akorát původní představa umělé inteligence jen zvětšovala k nultému bodu obrázku (levý horní roh). Představa je však spíše taková, že bychom chtěli přibližovat na místo, kde je zrovna umístěn kurzor. Metod je více, ale snadné řešení je zanechat návrh stejný a do funkce přiblížení zanést i funkci posuvu. Tedy při přiblížení vůči bodu v mapě zvětšíme celý obrázek a posuneme jej tak, aby se původní bod, na němž spočíval kurzor, opět nacházel pod kurzorem. Jednoduchá úvaha a výsledkem je funkční přibližování i pohyb mapou.

Další funkcí bylo vykreslení cíle. Funkce posouvání a zoom jsou nezávislé na počáteční pozici kurzoru, v podstatě stačí, že došlo k chycení obrázku, a následný rozdíl souřadnic nám dává výsledek posuvu. Tak tomu však není u vykreslení bodu v canvasu. Souřadnice kurzoru, jež nám předá událost kliknutí, jsou vztaženy k jinému souřadnému systému, než je souřadný systém canvasu. Je tedy potřeba při kliknutí přepočíst souřadný systém a zároveň si bod cíle uložit nezávisle na tom, jak si operátor přiblížil mapu. Zároveň je nutné, aby byl cíl vždy jen jeden a aby bylo možné jej zrušit. Proto po prvním kliku povolíme jeho existenci a vykreslení, pokud dojde ke kliknutí druhému na stejné souřadnici, pak je cíl zrušen, pokud na pozici jiné, je přesunut na novou pozici, čímž udržíme logiku jednoho cíle. Teoreticky by bylo možné také přidat takzvané checkpointy nebo body, jimiž chceme, aby robot prošel, ale to zatím implementováno není.

Pro přepočtení souřadných systémů lze snadno aplikovat rovnice pro přepočet souřadných systémů, jež jsou využity i u prostorových transformací pozice robota. 4.1 U robota se používají i přepočty souřadných systémů posunutých nejen v ploše, ale i v prostoru, případně i souřadné systémy navzájem posunuté. K tomu lze využít takzvanou D-H (Denavit-Hartenbergovu) konvenci či úmluvu , která určuje pravidla pro přepočty mezi souřadnými systémy umístěnými v kloubech víceosých robotů. Z této konvence nám stačí využít logiku, jež se v ní využívá. Tato logika říká, že pro přesun mezi jednotlivými souřadnými systémy stačí provést dvě operace translace a dvě operace rotace, pokud jsou v jedné ose souřadného systému systémy svázány.

Pro 2D souřadný systém chodícího robota jde tato logika ještě více zjednodušit. Pro přechod mezi dvěma souřadnými systémy ve 2D by nám měla stačit i jen jedna translace 4.1 a jedna rotace 4.2. Největší zjednodušení je možné využít u přepočtu kliknutí myši uvnitř canvasu, zde stačí i jen jedna translace.

Větší problém by mohl být získat hodnoty těchto vzájemných posunů, s tím však vývojáři již počítali, a tak stačí projít seznam metod canvasu. Jednou z metod je *getBoundingClientRect*. Díky této metodě můžeme zjistit všechny hranice prvku canvas. Z těchto souřadnic je poté dopočten nulový bod canvasu, a tím i nulový bod souřadného systému, do nějž je potřeba provést posun souřadného systému kurzoru.

Dalším krokem bylo i rozšíření těchto interakcí na jiná zařízení, aby bylo možné robota obsluhovat i pomocí dotykových obrazovek. Ty mají odlišné rozhraní k předání události dotyku a posuvu. Jejich zkoušení mimo debugovací prostředí přímo v prohlížeči bude však možné až s přesunem webové stránky na ESP32 a pak přímo na nějakém zařízení s dotykovým displejem.

$$\begin{bmatrix} B_x \\ B_y \end{bmatrix} = \begin{bmatrix} A_X \\ A_Y \end{bmatrix} + \begin{bmatrix} P_X \\ P_Y \end{bmatrix} \tag{4.1}$$

$$R = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \tag{4.2}$$

4.2 Trasovací funkce

Po úvaze nad spojováním původního ovládání s novým bylo rozhodnuto o ponechání trasových výpočtů na straně operátora, tedy v počítači nebo chytrém zařízení. Nejenže zařízení sloužící jako ovládání robota bude mít asi vždy vyšší výkon, ale zároveň bude mít i větší možnosti, co se týká paměti. Navíc, jak bylo zjištěno někdy v této době vývoje, nebude možné použít STM32 na senzorické desce jako hlavní řídicí procesor, a tak by většina výpočtů zůstala na ESP32, které bude i tak hodně zatíženo komunikací mezi operátorem a robotem.

Funkce na hledání trasy byla přepsána do javascriptu a implementována do řídicího webového rozhraní. Vykreslení probíhá přímo do mapy. Samotná funkce ovšem musela být poupravena, protože javascript nepovoluje příliš hluboké vnoření, k němuž v případě původní rekurzivní funkce dochází, respektive tomu automaticky brání prohlížeč, jenž javascript interpretuje nezávisle na hardwaru. Úprava funkce se na tento problém zaměřila a přílišnému vnoření brání tak, že je-li možné pokračovat volně (po vektoru) k cíli, pohybuje se k cíli dál po přímce pomocí interní podfunkce, místo aby se vnořila. K dalšímu vnoření dojde pouze tehdy, obchází-li překážku.

Čas od času byly v simulační mapě nalezeny body, z nichž nebyla tato funkce schopna dosáhnout cíle ani opačné konfiguraci směru, od cíle do startu. Bylo tedy nutné na takové případy vymyslet řešení, kterým zajistíme dosažení cíle. Sledování každého kroku vnoření by bylo dost pracné a časově náročné, a tak bylo provedeno

několik pokusů, ze kterých vyplynulo, že občasné posunutí bodu byť jen o jediný krok jiným směrem zajistí změnu výsledku hledání trasy. Nejspíše zde dochází k tomu, čeho se dá u javascriptu nejvíce obávat, matematických nedostatků. Často se setkáváme s příklady, jež javascript nedokáže správně vyřešit. Jedním z nejznámějších příkladů je, že $0,0002 + 0,0001$ nevrátí výsledek $0,0003$, ale $0,00030000000000000003$. Obdobný problém může vést i k tomu, že se celé hledání trasy zasekne v jednom bodě, stačí však i jen lehce pozměnit cíl či start a jsme schopni problém s hledáním trasy obejít.

V případech, kdy startovací podmínky dané funkci nevrátí trasu, můžeme zavolat funkci s jinými parametry. Takové parametry mohou být třeba body mapy v menším měřítku, například každý desátý bod, a jestli je možné se do tohoto bodu dostat jak ze startu, tak z cíle. Porovnáním délek takto vzniklých tras můžeme najít tu nejkratší z nich, a tak se poměrně efektivně vyhnout tomuto problému. Během testování bylo navíc naprosto zřejmé, že vykreslování i výpočet v Matlabu jsou značně pomalejší než vykreslování a vypočtení trasy implementované v prohlížeči. Rychlost nalezení trasy je tak značně kratší. Zatímco při simulacích v Matlabu trvalo najít trasu ve složitějších mapách i několik desítek sekund, funkce ve webovém prohlížeči je schopna trasu najít a začistit během několika milisekund. Neměl by tedy být problém trasu přepočíst klidně i po každém kroku robota nebo kupříkladu pro každý desátý bod mapy.

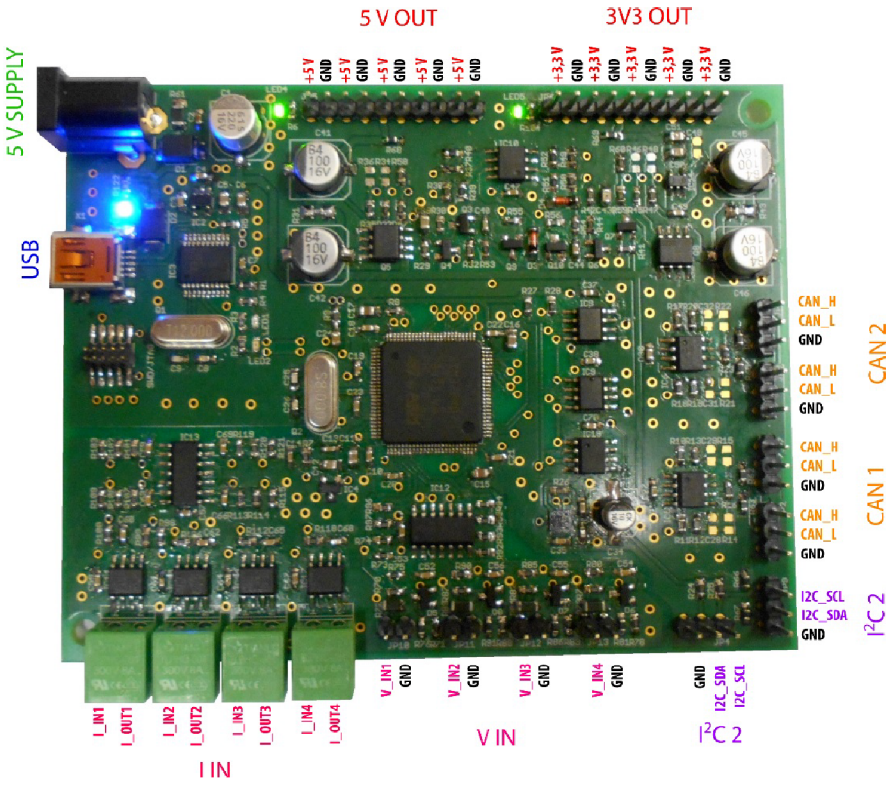
Struktura funkcí zůstala podobná. Pokud existuje robot a je mu zvolen cíl, pak se snaží trasovací funkce dostat po směrovém vektoru ze startu do cíle, jestliže narazí na překážku, zkouší ji obejít pomocí rekurze v sedmi dalších směrech vždy o 45° posunutých. Hledání trasy je ukončeno v případě nalezení již prvního úhlu vedoucího k cíli, proto by často docházelo k obcházení překážky jen z jedné strany, což může tvořit zbytečně dlouhé trasy, proto je zkoušen kladný a záporný směr otočení pro stejný úhel, než se přejde na další test o 45° většího úhlu.

Pokud se funkci podaří dorazit do cíle, je hledání ukončeno a přechází se do fáze začištění trasy, jehož integrace odpovídá funkci popsané výše 3.1.3. Pokud se funkci do cíle dorazit nepodaří, je zavoláno hledání trasy z cíle do startu. Jestliže se to nezdaří ani v tomto případě, zkouší se body zmenšené mapy, a pokud selže i toto hledání, pak jsou pravděpodobně robot a cíl rozděleny neprůchozí překážkou.

5 Integrace nových desek

5.1 Senzorická deska

V původním předpokladu prací, které proběhnou na robotu Hexor® II, se počítalo s integrací desky od pana Kredby. Tato deska, jak se po podrobnějším prozkoumání ukázalo, však nebyla připravena na to, aby se stala řídicí deskou robota. V plánu pana Kredby nejspíš bylo tuto desku použít pouze jako řídicí systém pro senzorickou část, tedy že se k ní připojí jeho systém ultrazvukového měření vzdálenosti a další podobné senzory a že bude komunikovat s řídicím systémem po sběrnici CAN. Ty byly k tomuto účelu připraveny rovnou dvě. Senzorická deska nedisponuje volně vyvedenými vstupně výstupními porty, až na pár pinů jsou všechny obsazeny nějakým integrovaným obvodem, tedy není možné jen změnit využití pinů a řídit jimi servomotory robota. Rozdíl byl i v původně použitých komunikacích, řízení robota v okamžiku započetí práce bylo postaveno na sběrnici UART, který je u nové desky použit na komunikaci přes USB. Není možné, aby přímo nahradila desku robota, protože nemůže ovládat jeho periferie ani ji nelze ihned zařadit mezi anténu a starou řídicí desku. Z aktuálních funkcí, jimiž deska disponuje, je možné využít primárně jen rozvod 3,3 V, 5 V a integrovaný akcelerometr. Sensory, které jsou pro řízení robota instalovány, nepotřebují ani přesné měření proudu, ani napětí, tedy téměř všechny hlavní funkce sensorické desky budou nevyužity.



Obrázek 5.1: Senzorická deska v podobě vytvořené a nastavené panem Kredbou [8]

Jediný UART, který byl na sensorické desce připraven, je připojen na převodník ke komunikaci přes USB, přestože je čip samotný schopen tuto funkci nabídnout i bez převodníku. Bylo potřeba upravit část desky, aby mohla proběhnout změna druhého I2C na UART, jednalo se o nejméně invazivní variantu změny, a v případě dalších úprav by tak nemělo být složité tuto změnu vzít zpět. Konkrétně byly odstraněny pull-down rezistory u obou konektorů druhého I2C a interně byly jeho piny přenastaveny na UART. Program pro původní UART komunikující přes USB byl doplněn tak, aby byl přenášen i na UART nový. Senzorické desce je možné posílat dotazy tak, jak si to asi původně pan Kredba představoval, ale jelikož je využit jen gyroskop a akcelerometr, není ani HUB připraven na příchod dat jiného typu. Pokud by se pokračovalo v pracích stejným směrem, bude třeba program HUBu rozšířit o rozpoznání a předání těchto dat do objektu ESP32 a doplnit původní počítačový program, upravit webserver ESP32 (nebo pouze jednu z variant) nebo vytvořit nový systém.

Na senzorické desce byla připravena komunikace přes UART, ale UART je sběrnice primárně stavěná pro komunikaci mezi dvěma účastníky. Pro případ, že chce komunikovat s více účastníky, nabízí se dvě řešení:

1. Můžeme zavést složitější protokol, který komunikaci mezi více účastníky dovolí, čímž se násobně zvětší množství přenesených dat, a to ještě za cenu složitosti celé komunikace a doby na zpracování, jelikož nejsou tyto protokoly

hardwarově integrovány, a tak by byla komunikace zbytečně zpomalena. Zároveň by do takového protokolu musela být zanesena i nějaká varianta hlídání a velení komunikace, jež by zajistila, že nikdy nebude vysílat více než jedno zařízení. Rozšíření UARTu pro více účastníků by tak způsobovalo skutečně nepříjemné komplikace. Protokol, který pan Kredba použil, byl tak z velké části zbytečný, protože přesto, že obsahuje adresu, na sběrnici mu stejně komunikují vždy jen dva účastníci.

2. Nebo použít pro každé dvě zařízení dva samostatné UART porty, tak jak je definovaný UART. Chceme-li komunikovat z jedné desky s více účastníky, musí deska disponovat stejným množstvím UART portů jako plánovaný počet účastníků.

Abychom mohli k robotu připojit ESP32, potřebujeme volný UART, avšak jediným volným na robotovi je ten, na němž je komunikace přes anténu. Pokud ji chceme zachovat, potřebujeme další UART port, kterých však není dostatečné množství, aby se desky daly pospojovat. K pospojování všech požadovaných subsystémových desek jsou zapotřebí čtyři UARTy. Po poradě s vedoucím této diplomové práce bylo rozhodnuto, že bude vytvořen jakýsi komunikační HUB, který dovolí spojit novou senzorickou desku, původní řídicí desku, novou komunikační jednotku (ESP32) a původní komunikační rozhraní ve formě antény. HUB bude disponovat dostatkem UART portů a bude doplněn o program schopný zprávy rozdistribuovat mezi jednotlivé desky tak, aby mohla být zanechána plná kompatibilita s původním řízením a aby bylo možné používat zároveň i nové řízení.

5.1.1 Parametry dílů použitých na senzorické desce

Hlavním řídicím čipem této desky je STM32, konkrétně model STM32F407VGT6, což je 32bitový mikroprocesor od firmy STMicroelectronics postavený na architektuře ARM Cortex M4. Disponuje 196 kB statické RAM, 1 MB flash paměti a je schopen pracovat na frekvenci 168 MHz. Má tři 12bitové A/D převodníky s až 24 kanály schopnými udělat až 7,2 MSPS. Dva D/A převodníky s rozlišením 12 bitů. Má až 84 programovatelných výstupů, kterým lze nastavit hardwarově integrované sběrnice, jako až 4× USART + 2× UART, 2× CAN, 3× I2C, 3× SPI, dokonce plno rychlostní USB 2.0 nebo 10/100 Ethernet. [14]

Napětí je možné měřit v rozsahu od 0 do 40 V, čehož je dosaženo nejdříve napěťovým děličem, který napětí sníží do rozsahu 0 až 5 V, dále je signál vyfiltrován a přes napěťový sledovač z jednoho operačního zesilovače a další dělič posouvající napětí do 3V logiky je přiveden na A/D převodník [8].

Pro přesné měření proudů instaloval pan Kredba integrované obvody ACS712, jež měří proud pomocí hallova jevu, čímž měření proudu galvanicky oddělí od samotného A/D převodníku, a které proud na výstupu vrací v napětí úměrném vstupnímu proudu. Pro lepší přesnost navíc ještě výstup zesílí pomocí invertujícího zesilovače,

vyfiltruje dolnopropustním filtrem a napěťovým děličem opět posune do 3V logiky. [8]

Aby autor zabránil ztrátě kalibračních dat pro senzory, osadil desku navíc dvěma samostatnými paměťmi typu EEPROM o velikosti 2 kB, mezi nimiž jsou rozdistribuována kalibrační data tak, aby i při ztrátě jednoho nebyla data ztracena. Díky tomu bylo možné i po přepisu programu do nového prostředí zachovat funkčnost měřicích vstupů.[8]

Paměti jsou společně s teploměrem (TCN75A) a s čipem LSM6DS0 instalovány na sběrnici I2C. Jedná se o MEMs čip, díky kterému by se měla zpřesnit odometrie, udává zrychlení ve třech osách a rotaci kolem nich. Je aktuálně jediným aktivně využitým čipem ze senzorické desky. [8]

Pro sběrnici CAN byly instalovány budiče MCP2551, odpovídající standardu ISO-11898, jež mění vysílací a přijímací vstup kontroléru na CANH a CANL sběrnice CAN, čímž zvyšují odolnost sběrnice proti rušení, a přizpůsobují tak kontrolér, aby se mohl připojit k médiu, které může mít i jinou napěťovou hladinu než 3V, ale klidně až ±40 V dlouhodobě a krátkodobě až ±250 V, čímž celkem značně převyšuje požadavky samotné normy.

5.1.2 Integrace do robota

Kvůli tomu, že původní program na desce byl přehrán/smazán, originální kód byl psán v jiné, již nekompatibilní verzi STMcubeIDE a také kvůli možným úpravám, bylo potřeba většinu kódu zrevidovat a vyčíst původní nastavení jak z programu, z diplomové práce pana Kredby, tak i z dokumentace jednotlivých čipů, které pan Kredba použil [8]. Některé části nebudou použity, ale to neznamená, že by měly být vynechány v programu, jenž na desce již původně byl. Sice nikde není zmíněno, že by pro tento systém bylo plánováno konkrétní využití. Ale v budoucnu může na tyto části opět navázat další student, případně z nich čerpat.

Oproti vlastní práci udělal ovšem pan Kredba několik změn, nejspíše však až po odevzdání diplomové práce, v níž se o nich nezmiňuje. Hlavní změnou je, že například data z akcelerometru podle kódu odesílá jako čistá data standardně odesílaná po sběrnici UART, tedy ve formě 8bitových znaků (char) s paritním bitem, ne jako zkrácenou verzi protokolu, který označoval jako HB12-C. Také způsob, jímž dešifruje data, by data v této formě mohl přebírat jako velmi problémová vzhledem k tomu, že v hlavičce čeká nejdříve na velikost dat. Proto by v podstatě vždy obdržel data vadná. V aktuální verzi není přímo implementován jiný způsob čtení dat, než že přijatá data převedeme zpět na data z akcelerometru, protože k jiné komunikaci s touto deskou nedochází. Místo zbytečně velkého komunikačního protokolu by bylo možné například použít startovací/ukončovací kombinaci symbolů, a nebo v případě přenosu dat do velikosti bufferu příjemce prostě jen dělit zprávy na základě mezery mezi jednotlivými vysíláními, tedy přímo funkcemi, jež podporuje hardware STM32 a tím šetřit výpočetní čas. I u následného dělení je možné použít funkce nebo datové typy, které snižují dobu zpracování signálu, jako například místo posunů použít datový typ Union a celý datový prostor přijaté zprávy přetypovat na třeba strukturu dat. [8]

Této metody bylo využito jak při rozklíčování dat posílaných přes anténu z počítače, tak u převodu dat z akcelerometru v lehce pozměněné formě. Primární důvod na využití této metody přišel z inspirace u PLC, kde vstupy, výstupy i paměť lze nahlížet jako jednotlivé bitové adresy a lze s nimi tak manipulovat i mezi jednotlivými na sobě nezávislými procesory (jednotlivými PLC), ale zároveň je možné stejné adresy interpretovat i jako datové struktury. Přímá inspirace přišla z použití DP/DP coupleru, což je komunikační prvek, jenž od sebe oddělí dvě části komunikační sítě Profibus, tedy je možné opět použít téměř celý adresní rozsah, a mezi sítěmi propouští data, která jsou určena jemu. [8]

Porovnání UARTu a Profibusu z hlediska OSI/ISO

Jelikož UART a Profibus patří do stejné kategorie sériových sběrnic, není nijak překvapivé, že mají mnoho podobností a že přímo svádějí ke zpětné inspiraci. UART stejně jako Profibus ve fyzické vrstvě používá dva vodiče. Začínají se lišit ve spojové vrstvě; zatímco UART má délku rámce typicky 9 bitů (z toho 8 bitů datových), Profibus má základní rámec délky 11 bitů (z toho 8 bitů datových).

UART a Profibus se mnohem více liší v síťové vrstvě. Z hlediska složitosti je UART o proti Profibusu velmi zjednodušený. V základní podobě UART neobsahuje žádnou informaci o cílové adrese, které jsou přenášená data cílena. Je tak určen primárně ke komunikaci PtP (point to point mezi dvěma prvky), zatímco Profibus je složitější a podporuje až 128 účastníků, a dokonce i víc než jeden režim určování volby směru komunikace, Master-Slave nebo token passing/token ring. Ničím takovým UART v hardwarové implementaci nedisponuje. O implementaci adresace u UARTu se snažil právě protokol HB12-C použitý panem Kredbou, který je dle jeho slov vyvíjený na TUL, ale jelikož se nepodařilo dohledat více, než zmiňuje pan Kredba, asi nebyl úplně výhodnou variantou. Vzhledem k tomu, že pro poslání povelu místo například 5 Bytů, kde 1 Byte by bylo uvození komunikace, 3 Byty povel a nakonec 1 Byte ukončení zprávy, používá paket o velikosti 12 Bytů, je jeho využití téměř zbytečné. Sice v případě odesílání dat zvětší objem přenosu jen o 12 Bytů a dovoluje adresaci, ale zato násobně zpomalí komunikaci vzhledem ke své složitosti a stejně na sběrnici jsou jen dva účastníci, adresace tak není nutná. Ačkoliv disponoval dobrým zabezpečením proti chybě v přenosu, vlastně jeho využití nebylo nutné. V rozšířené části práce tedy využit není, je využit pouze v části, kde byl vyžadován deskou robota k přímé komunikaci.

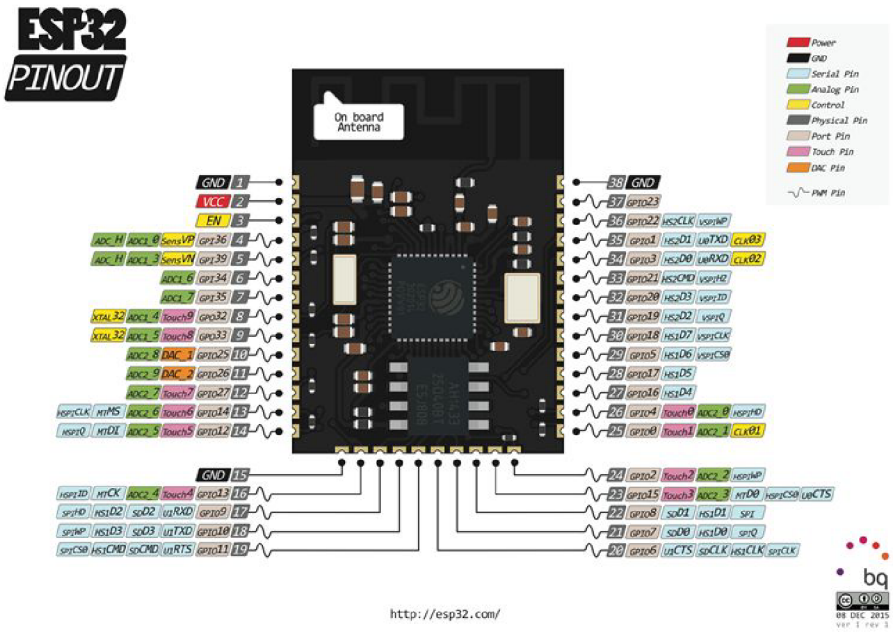
V aplikační vrstvě se u Profibusu mluví o třech variantách DP (Decentralyzed periphery), ta využívá metodu master-slave, což značně urychluje komunikaci, FMS (Fieldbus Message Specification), který podporuje práci s daty, ale zároveň je pomalejší, a posledním je PA (Process Automation), který je vhodný do výbušných prostředí, ale zároveň je omezen na rychlost 31,25 kbit/s. Oproti tomu je UART schopen komunikovat v rychlostech laditelných ve větším rozsahu, avšak pomaleji. Může pracovat do rychlostí okolo 92 kB/s. U robota je nastavena rychlost pro UART v komunikaci s anténou na 9 600 baudů, což přibližně odpovídá rychlosti 1 kB/s. Komunikace mezi spojovací deskou a ESP32 probíhá v rychlejším režimu, aby se stačil zvolený interface poslat v rozumném časovém intervalu. ESP32 však

maximální rychlost UARTu omezuje na rychlosti do 400 000 baudů, zatím byla z hlediska testované stability využita rychlost 250 000 baudů. Na komunikaci se senzorickou deskou byla použita rychlost 115 200 baudů. Tyto rychlosti se mohou kdykoliv změnit v závislosti na tom, zda bude potřeba přenášet větší množství dat.

5.2 ESP32

5.2.1 Parametry použitého ESP32

Konkrétním modelem použitého ESP je ESP32-WROOM-32D, od společnosti Espressif. Jedná se o modul obsahující 40MHz krystal, který je možné využít jako zdroj hodinového signálu pro CPU v čisté podobě nebo ještě interně vydělit až na frekvenci 240 MHz. Je vybaven až 4 MB flash paměti připojenými na SPI k CPU. Obsahuje hardwarové i softwarové sběrnice 3×UART, 4×SPI, 2×I2C, 2×I2S, TWAI (compatibilní s ISO11898-1 CAN 2.0), možnost přímého připojení SD karty, převodníky ADC (až 12 bitů a 18 kanálů) i DAC (8 bitů a 2 kanály), až 16 kanálů PWM generátorů pro ovládání LED i podporu PWM pro motory a také až 34 programovatelných vstupně výstupních pinů, z nichž šest je interně využito k připojení paměti a dalších šest je možné využít jen ve vstupní konfiguraci, a i další nezmíněné funkce a vlastnosti. Ale jeho hlavní výhodou oproti konkurenčním mikroprocesorům je, že je schopen komunikovat přes Bluetooth (v protokolech 4.2 BR/EDR nebo i LE) a Wi-Fi (802.11 b/g/n), které mají vlastní oscilátory, a že je možné ho programovat v prostředí ArduinoIDE, jež je velmi rozšířené a jeho adaptace jazyka c/c++ je snadno uchopitelná. Zároveň je možné ho programovat i v čistém jazyce c v jiných prostředích nebo ve vývojovém prostředí od výrobce, avšak v tomto prostředí není tak veliká komunita vývojářů, kteří by přispívaly svými výtvory, a tak je mnohem složitější v případě řešení obtíží. Díky komunitě se již objevily i možnosti programování v jiných vyšších jazycích, jako Python, Java a jiné. [4]



Obrázek 5.2: ESP32 rozložení pinů na základním modulu [2]

Pro prostředí arduinoIDE je navíc velké množství hotových knihoven a příkladů, z nichž je možné brát inspiraci pro pochopení některých vlastností modulu. Velmi dobrým příkladem takové knihovny je třeba knihovna AsyncWebServer od me-no-dev, která pomocí svých přehledných funkcí značně zjednodušuje práci s distribucí a obsluhou webserveru. Podporuje například možnost přímého načtení souborů webu z paměti mikroprocesoru, tudíž je možné vytvořit si webovou stránku offline a následně ji uložit do paměti a distribuovat skrze interní webserver. Oproti tomu obsluha v podobě HAL funkcí přímo od vývojářů je složitá a pro obsluhu webové stránky je zapotřebí velmi komplexní znalost všech potřebných protokolů a datových výměn, aby bylo možné obsloužit klienta připojeného k serveru.

Oproti arduinoIDE však využití prostředí a HAL funkcí od vývojářů dovoluje úplný přístup ke všem funkcím procesoru, respektive k oběma jádrům zvlášť, a tím dovoluje rozdělit práci mezi jádra tak, jak si vývojář řekne. To je v některých částech výhodou, avšak programování se tím stane složitější, protože by bylo potřeba správně časovat jednotlivé instrukce a časování synchronizace jednotlivých jader, o což se v případě využití ArduinoIDE stará FreeRTOS, a sice nedává takovou svobodu, ale usnadní práci a zrychlí vývojovou a debugovací fázi. Tím však ve stejnou chvíli svým způsobem lehce debugovací fázi zhorší, protože nepodporuje J-TAG a krokování programu, k debugu je tak nutné využít jen sériový monitor.

5.2.2 Program ESP32

Vzhledem k jednoduchosti a množství možných komunikací, které podporuje, bylo ESP jasnou volbou pro komunikaci mezi počítačem a robotem. Jelikož je vhodné mít možnost ovládat robota bez nutnosti vozit s sebou počítač, podpora webserveru

a Wi-Fi je k tomu perfektní. Nebude nutné instalovat žádnou aplikaci do chytrého zařízení a vše by měla zvládnout webová stránka, ve které bude integrováno ovládání. Přímo na ESP32 běží komunikace se spojovací deskou, jež si postupně vyměňuje datovou strukturu řízení robota.

Celá struktura má velikost asi 1,5 kB a přes UART se oběma směry vymění 10× za vteřinu. Sice je tato datová struktura větší než původní krátké zprávy, pomocí nichž komunikuje původní ovládání, ale díky rychlejšímu nastavení komunikace je možné komunikovat poměrně rychle, a navíc máme pořád všechna data pohromadě, na rozdíl od původní komunikace, která musela každá jednotlivá data nejdříve vyžádat.

Na straně ESP32 si tedy z dat, která přijdou, program vyseparuje ta, jež potřebuje k řízení, rozhodování, diagnostice atd. A v odpověď zapíše data upravená operátorem jako změny v řízení a odešle zase zpět, kde se rozhodne, zda mohou být provedeny operace, o něž ESP32 zažádalo.

Protože buffer pro UART na straně STM32 je větší a dokáže pojmout vlastně veškerá data celé struktury naráz, není potřeba data ničím obohacovat, jdou odesílat jako čisté 8bitové znaky rozšířené o paritní bit. V opačném směru, zpráva od STM32 do ESP32, se buffer musí naplnit sedmkrát, aby obsáhl veškerá data, začátek a konec je tedy obohacen o dva znaky. Na začátek zprávy bylo přidáno "?s" a na konec "!x", mezi prvním a druhým znakem obou sekvencí v ASCII přepočítané na decimální tvar je mezera minimálně 50 znaků a rozdíl v binárním tvaru je také relativně veliký (? - 00111111, s - 01110011, ! - 00100001, x - 01111000).

Tyto znaky nejsou tak často používané, nebo spíše jejich kombinace v přímém sledu, v případě jiné zprávy je snad jejich náhodná kombinace dostatečně nepravděpodobná. Z hlediska dat jsou vzdálené natolik, abychom se takové kombinaci vyhnuli. Obzvláště když víme, že začátek zprávy bude vždy v rozšířeném bufferu, do něhož se data skládají vždy od začátku, protože po nalezení ukončovací kombinace se kontroluje velikost a následně se data opět načítají od začátku bufferu.

Část dat je využita přímo v ESP32, a tedy předána webserveru, kde jsou následně zobrazena operátorovi. Ne všechna data se zobrazují přímo, některá, jako například které zařízení je řídícím operátorem, se zatím nezobrazují, ale jsou využita jen v rozhodovací logice. V případě, že ESP32 není aktuálním operátorem, se data jen nepředají dále v řetězci komunikace, a tak se podle nich robot nemůže řídit. Část dat navíc většinu času vůbec nemusí být využita. Například data o plánovaných krocích v manuálním módu a další podobná data.

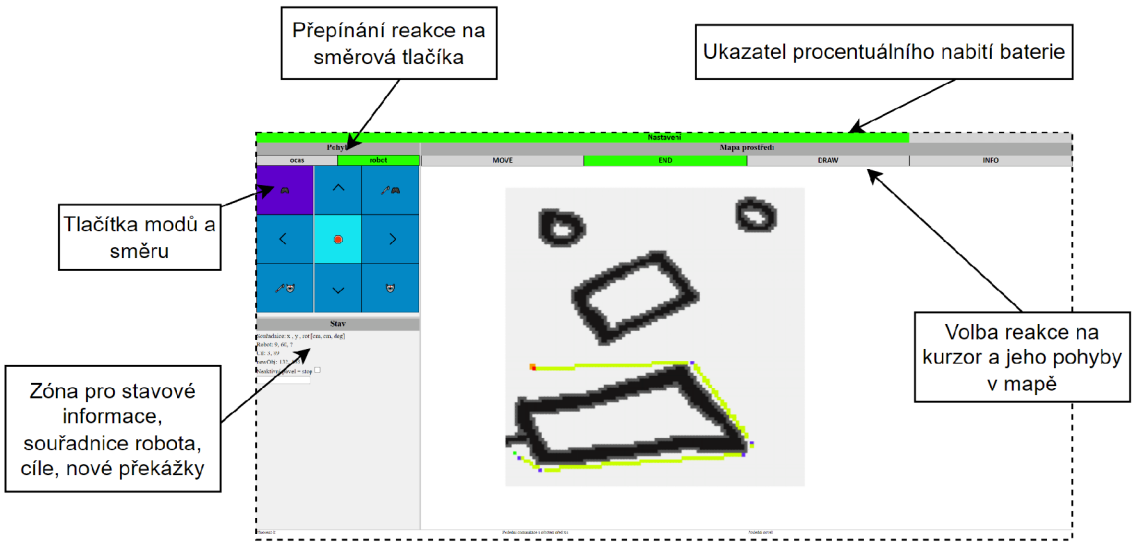
5.2.3 Webserver

V ESP32 probíhá obsluha dat přijatých z webového serveru, ale ten reaguje převážně jen na povely operátora a několik časových eventů volání, které například aktualizují polohu robota. Operátor ovládá pohyb robota a jeho ocasu pomocí výběrových tlačítek nebo pomocí klávesnice. Byla přidána i funkce RTZ (return to zero) pro ovládání. Nebo pomocí již popsané mapy, kde je možné přidávat překážky, volit cíl trasy nebo jen sledovat pohyb robota v případě, že operátor připojený přes ESP32 není hlavním operátorem robota.

Ve vrchní části stránky je barevný pruh zobrazující procentuální nabití baterie robota. V tomtu pruhu se nachází ještě vysouvací tlačítka s nastavením pro připojení k Wi-Fi a s nastavením přiřazení kláves. Přiřadit by měly jít všechny znaky na klávesnici a přiřazují se k funkcím:

- přepínání mezi ovládáním pohybu ocasu a robota,
- přepínání mezi módy robota (manuální, asistovaný manuální, asistovaný automat, plně autonomní režim, případně i prezentační režim),
- nastavení tlačítek směru.

Jelikož původní ovládání má přesně dané struktury, tedy způsob řízení úplně měnit nelze, režimy jsou přepočítány na kroky provedené většinou v manuálním režimu a data ze senzorů jsou vyčítána v závislosti na vybraném režimu. Na webserveru se provede přepočet na pohyby, které by musel udělat operátor, a postupně se posílá sekvence kroků. Pro zobrazení trasy volané z původní aplikace byl zachován původní formát cesty, ale pouze pro její zobrazení. Pro prezentační režim byla zachována zpráva dovolující se do tohoto režimu přepnout, ale jinak není tento režim do ovládání z ESP32 implementován. Povely jsou předávány přes ESP32, kde se z instrukce mění na zvolenou strukturu, jež je popsána v příloze A.2, jež se předá do HUBu, jenž je opět převede na jednotlivé povely, které již odpovídají struktuře původních povelů robota.



Obrázek 5.3: Ukázka webové stránky pro ovládání robota (offline verze)

V podstatě se jedná o jednoduché řízení, které nejdříve otočí robota k následujícímu výraznému bodu trasy a následně se robot pohybuje rovně, až dokud nedorazí do okolí bodu. Informaci o poloze dostává z odometrie i z gyroskopu a akcelerometru.

Prvním krokem je srovnat natočení souřadného systému akcelerometru a robota v zobrazené mapě. Poté se z vektoru mezi bodem robota a výrazným bodem vybere

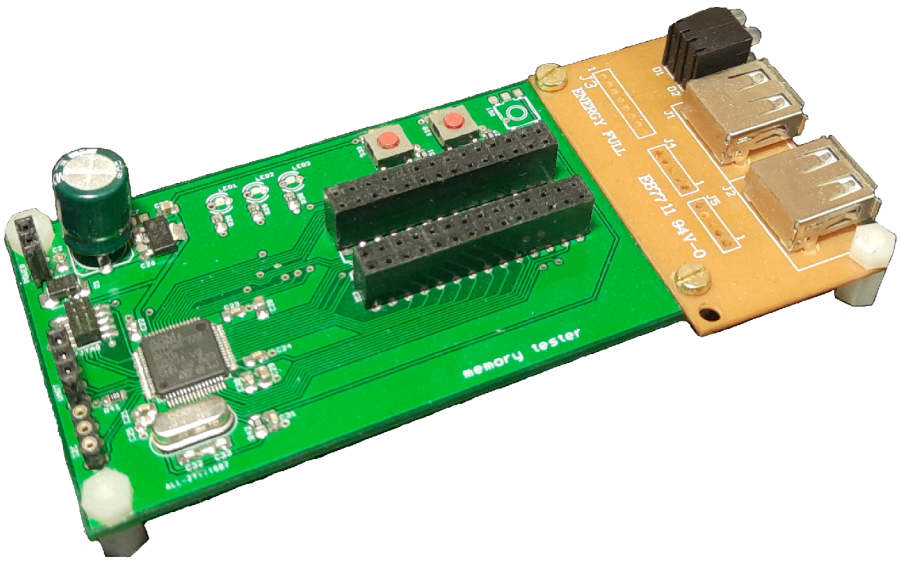
kratší směr k otočení. Pro přesně čelem vzad se vybírá otočení vpravo. Když se regulace směru dostane do dostatečně malé odchylky, vyrazí robot směrem vpřed a jde tak dlouho dokud nedorazí podle odometrie do zvoleného bodu v mapě nebo dokud nenarazí na překážku, která by ho nutila změnit směr. Robot je v mapě reprezentován jedním bodem, pro určení natočení má přední stranu bodu přebarvenou, aby bylo operátorovi zřejmé, kterým směrem je v mapě vpřed. Později by bylo možné také udělat také variantu pevného robota a rotující mapy.

5.3 HUB

Jak již bylo zmíněno, k robotu byla přidána nová deska, jež má za úkol spojit dohromady všechny nové moduly. Jelikož je vybavena opravdu výkonným čipem, bylo přistoupeno k variantě přepočtu předávaných dat na zprávy odpovídající původní struktuře zpráv mezi počítačem a robotem právě zde, a ne v ESP32.

5.3.1 Parametry čipu na HUBu

Program pro desku v práci přezdívanou jako HUB je psaný ve stejném prostředí jako program pro senzorickou desku, STM32CubeIDE. Konkrétní využitý čip je STM32F722RET6, což je 32bitový mikroprocesor od firmy STMicroelectronics postavený na architektuře ARM Cortex M7. K dispozici nabízí 256 kB statické RAM + 4 kB pro nízkospotřebové módy, 512 kB flash paměti s ochranou proti vyčtení. Má tři 12bitové A/D převodníky s až 24 kanály schopnými udělat až 7,2 MSPS. Dva D/A převodníky s rozlišením 12 bitů. Má až 48 programovatelných výstupů, kterým lze nastavit hardwarově integrované sběrnice, jako až 4× USART + 2× UART, 1× CAN, 3× I2C, 3× SPI a plno rychlostní USB 2.0. [15]

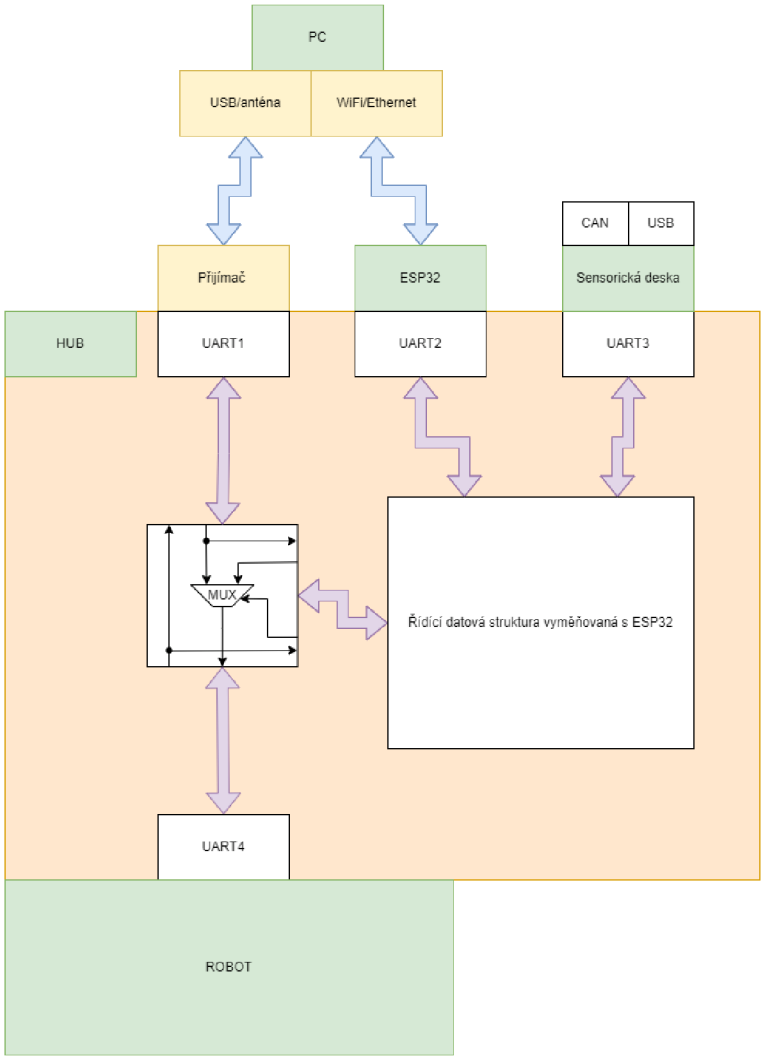


Obrázek 5.4: Deska přetvořená na robotův komunikační HUB modul

5.3.2 Popis programu

Podle toho, který komunikační kanál je zvolen jako hlavní operátor, je interně rozhodnuto, zda budou zprávy z antény krom zápisu do datové struktury i propisovány na výstup, nebo zda budou na výstup posílána data z datové struktury přepočítaná na zprávy odpovídající původní struktuře v časových intervalech 100 ms. Pokud nejsou přijímány žádné povely a hlavním operátorem je komunikace z ESP32, pak dochází k dotazům na hodnoty na senzorech robota. K dotazům na hodnoty senzorů robota dochází i v případech, kdy povely posílány jsou, ale už dlouho nedošlo k obnovení hodnot ze senzorů ve struktuře dat robota.

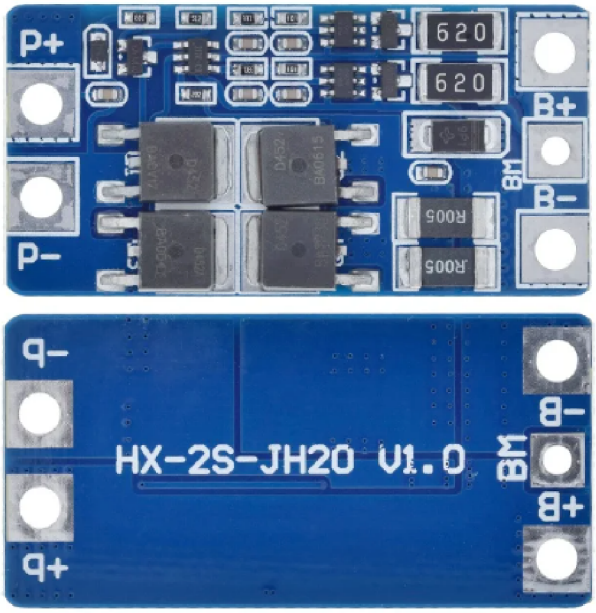
Deska využitá jako HUB byla původně navržena k testování paměťového čipu, ale byla předem připravena na to, že se může použít i k jiným účelům, tudíž většina jejích pinů byla vyvedena na samostatné pady, které je možné osadit nebo přímo k nim připájet další rozšíření. Je vybavena rozšiřovacím modulem pro připojení USB-A konektoru, jenž zatím nemá žádnou funkcí, avšak bylo by ho možné opatřit flash diskem a ukládat na něj logovací soubor nebo třeba mapy. Avšak nakonec k tomuto využití zatím nedojde, protože nebyl implementován algoritmus, který by dovedl rozpoznat přesnou polohu ve zvolené mapě, a tak je téměř zbytečné si mapu ukládat, protože pokud bychom v robotu mapu načetli, stejně se v ní nezorientuje. Jediným využitím by bylo například prostorové mapování pro využití k dalšímu zpracování nebo pro jiného robota. Ale to nebude součástí této práce.



Obrázek 5.5: Reprezentace logiky komunikace mezi deskami

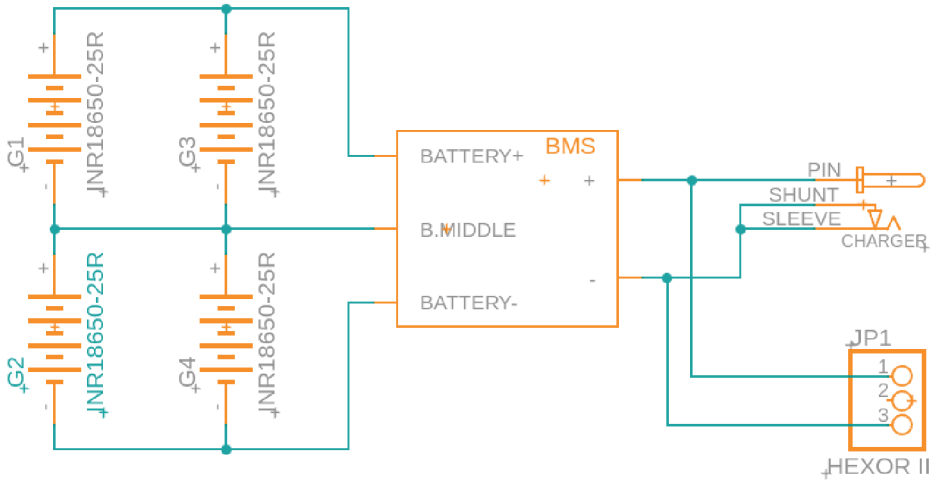
6 Baterie

Součástí zadání bylo také aktualizovat power managment robota. Jelikož se tomu minulý vlastník věnoval relativně hodně, dalo by se říci, že není moc míst, kde by šlo příliš zlepšovat. Alespoň z hlediska průzkumu, který kolem možných voleb udělal. Jeho řešení obsahovalo nabíjecí modul, který je schopen nabíjet baterii proudem až 2,5 A, a dokonce i balancovat hodnotu článků baterie proudem až 160 mA. Jelikož se od doby výroby jeho vlastního řešení trh s bateriemi i s příslušenstvím posunul značně vpřed, byly nakoupeny nové baterie podobného typu a kapacity, jež vybral i pan Kredba. V tomto výběru udělal již on dostatečný průzkum, a přestože velikostí a typů baterií přibylo, stále jsou jedny z nejlepších v poměru váha a energetická kapacita Li-Ion baterie. Protože robot nepotřebuje příliš velký dlouhodobý odběr, není potřeba v tomto směru nic měnit. Jak pan Kredba sám poznamenal, životnost těchto baterií se pohybuje od 5 do 10 let, často v závislosti na tom, v jakém stavu jsou skladovány a jak je o ně pravidelně pečováno. U robota, který většinu času tráví v krabici, není předpoklad denní péče. Při nepravidelném použití už baterie mohli ztratit velké množství své původní kapacity. S vývojem trhu přišly i malé moduly BMS, jež jsou schopny také balancovat a navíc jsou násobně menší, čímž se uvolní místo pro nové desky. Zároveň i k nabíjení už přibylo více typů zdrojů, respektive nabíjecích systémů, které velmi pravděpodobně nabíjecí modul vyrobený již dříve předčí – minimálně tím, že jsou schopny dodat i více než původních 2,5 A, tedy by měly být méně zatíženy a tím i efektivnější.



Obrázek 6.1: Vybraný BMS modul rozměr $47,5 \times 23,5$ cm

Konkrétně byly opět vybrány baterie typu Li-Ion, primárně kvůli ceně Li-Pol baterií. Tento typ robota nemá velký odběr proudu, takže nemá důvod Li-Pol baterie protlačovat. Cena Li-Pol baterií proti Li-Ion bateriím je pro stejnou kapacitu minimálně dvojnásobná, záleží na výrobci a následně na maximálním odběrovém proudu. Kapacita baterie by teď měla být asi 5000 mAh, mělo tedy by přibýt asi 2000 mAh. Hodnotou 5000 mAh si v praxi nemůžeme být zcela jisti, protože nákup všech nových dílů proběhl z asijského giganta Aliexpressu, takže kvalita a původ zboží nejsou vždy zaručeny. Typ nových článků je INR18650-25R od společnosti Samsung, napěťový rozsah jednoho článku je od očekávaných 2,5 V do 4,2 V a maximální stále odebíraný proud je 4C, tedy 20 A. BMS modul, který byl vybrán, podporuje dlouhodobý odběr 10 A, kromě funkce balancování poskytuje i ochranu proti přebití a tepelnou ochranu. K němu byl vybrán ještě nabíjecí zdroj kompatibilní s původním připojením na robotu.



Obrázek 6.2: Zapojení BMS a baterie

Protože bateriové články jsou čtyři, ale napětí, kterému jsou systémy robota přizpůsobeny, je do 9 V, bude vhodné držet se stejné nebo alespoň podobné napěťové hladiny. S výběrem stejných typů baterií to znamená, že bude muset zůstat i stejný způsob jejich pospojování, tedy sério-paralelní zapojení článků. BMS moduly jsou stavěny pro články zapojené sériově od dvou do asi 6 článků. Kvůli napětí byl vybrán modul pro dva články. Pokud chceme dva články sériově a dva paralelně, tak ony dva paralelní se zapojí jako jeden sériový a BMS s ním pracuje jako s článkem s větší kapacitou. Paralelní zapojení článků způsobuje samovolné vyrovnání napětí.

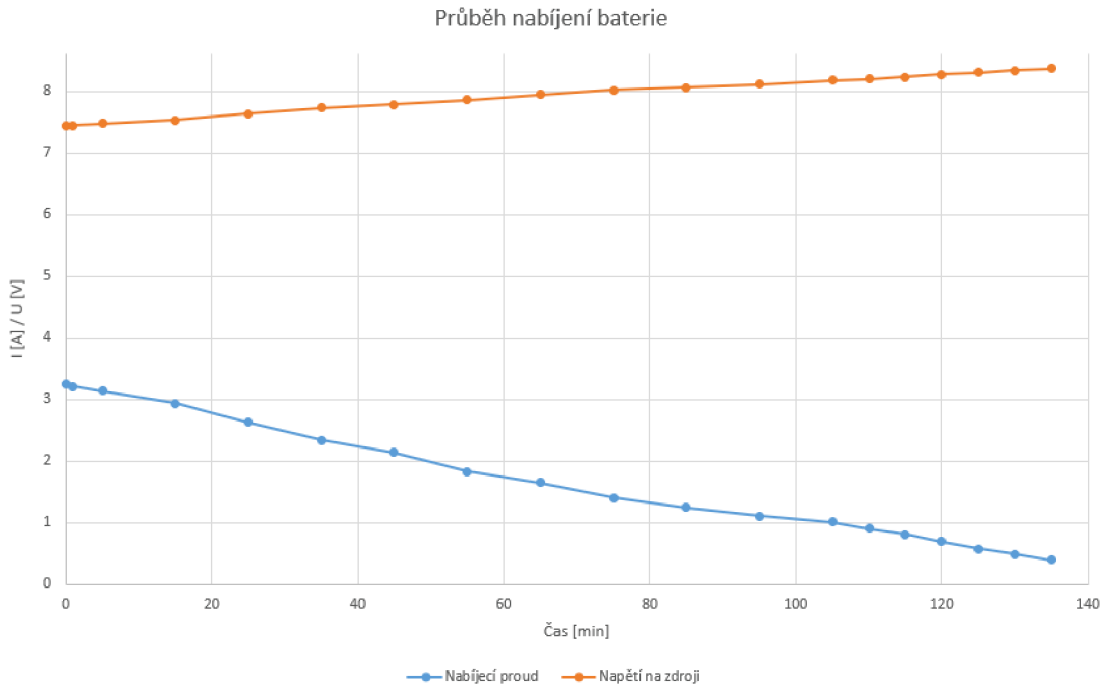
6.1 Měření nabíjecího zdroje

Podle předpokladu alespoň jeden ze zakoupených nových dílů fungoval jinak, než bylo očekáváno. Prodejce udává jak v popisu zdroje, tak i na těle samotného zdroje, že je schopen dodat 5 A. Podle provedeného měření se této hodnotě však ani nepřiblíží. Podmínky sice nebyly takové, aby od zdroje takto veliký proud vyžadovaly, přesto byl očekáván proud vyšší. Baterie byla sice vybitá, avšak ne až na úplně minimální hodnotu, a tak je možné, že by zdroj byl schopen tento proud dodat, ale například jen opravdu úplně vybité baterii a nejspíše pouze po krátkou dobu dobíjení. Zdroj sice nedosáhl deklarovaného maxima proudu, to však není zcela na závadu, pomalejší nabíjení je pro baterie obecně vhodnější. Ideální maximum nabíjecího proudu se pohybuje okolo 1,5 C ($1,5\times$ kapacita článku), pro baterie je však lepší nabíjení proudem okolo 0,5 C ($0,5\times$ kapacita článku). Pro použité články se tedy pohybujeme v rozmezí od 1,25 A do 3,75 A, což je proud na dosažený začátku nabíjení.

O úplně ukázkový nabíjecí zdroj se však nejedná. Ideálně by nabíjení Li-Ion baterií mělo být CCCV (3/4 času s konstantním proudem a poslední 1/4 času s konstantním napětím). Ideálním zdrojem pro tento typ baterií je například laboratorní zdroj, jemuž je na začátku nastaveno cílové napětí a omezení proudu na vhodnou

hodnotu. Po připojení by se tak nabíjení spustilo s omezeným proudem, a dokud se napětí baterie nepřiblíží napětí nastavenému na zdroji, je napětí automaticky snižováno v závislosti na proudu. V poslední fázi se začne proud snižovat a nabíjení se ukončí při poklesu proudu pod zvolenou mez (většinou 100 mA).

Měření proběhlo v domácích podmínkách na nekalibrovaných měřících přístrojích, jež spadají do kategorie hobby, tudíž měření nemusí být úplně přesné, avšak dává nám alespoň představu o tom, jak samotné nabíjení probíhá. Konkrétně se jednalo o Digitální multimetr PDM-250-a2 použitý jako ampérmetr nastavený v rozsahu 10 A, jenž má v tomto režimu přesnost 10 mA $\pm$ (3.0 % + 7). A jako voltmetr byl použit TRMS digitální clamp metr PZM 2 A2 nastavený na rozsah 20 V s přesností 0,01 V $\pm$ (0.5 % + 5). Pro kontrolu byl před zdroj zapojen měřič spotřeby, z nějž lze na displeji vyčítat síťové napětí, odebíraný proud, výkon a účiník. Jelikož měřák je staršího data výroby a manuál se již nepodařilo dohledat, výsledkům se nedá plně důvěřovat a ani přesnost nemá nijak velikou již na displeji, u výkonu zaokrouhluje na celé watty a u proudu na jedno desetinné místo, tedy je jeho výstup opravdu velmi orientační. Z toho důvodu nejsou jeho data plně vyhovující a závěry, jež z nich vyplývají nemusejí být úplně přesné. Avšak alespoň orientačně bylo možné určit, že účinnost nabíjecího zdroje se pohybovala od 70 % do 90 %, 85 % bylo dosaženo zhruba v polovině nabíjení (po 65 minutách).



Obrázek 6.3: Průběh nabíjení baterie

7 Montáž na robota

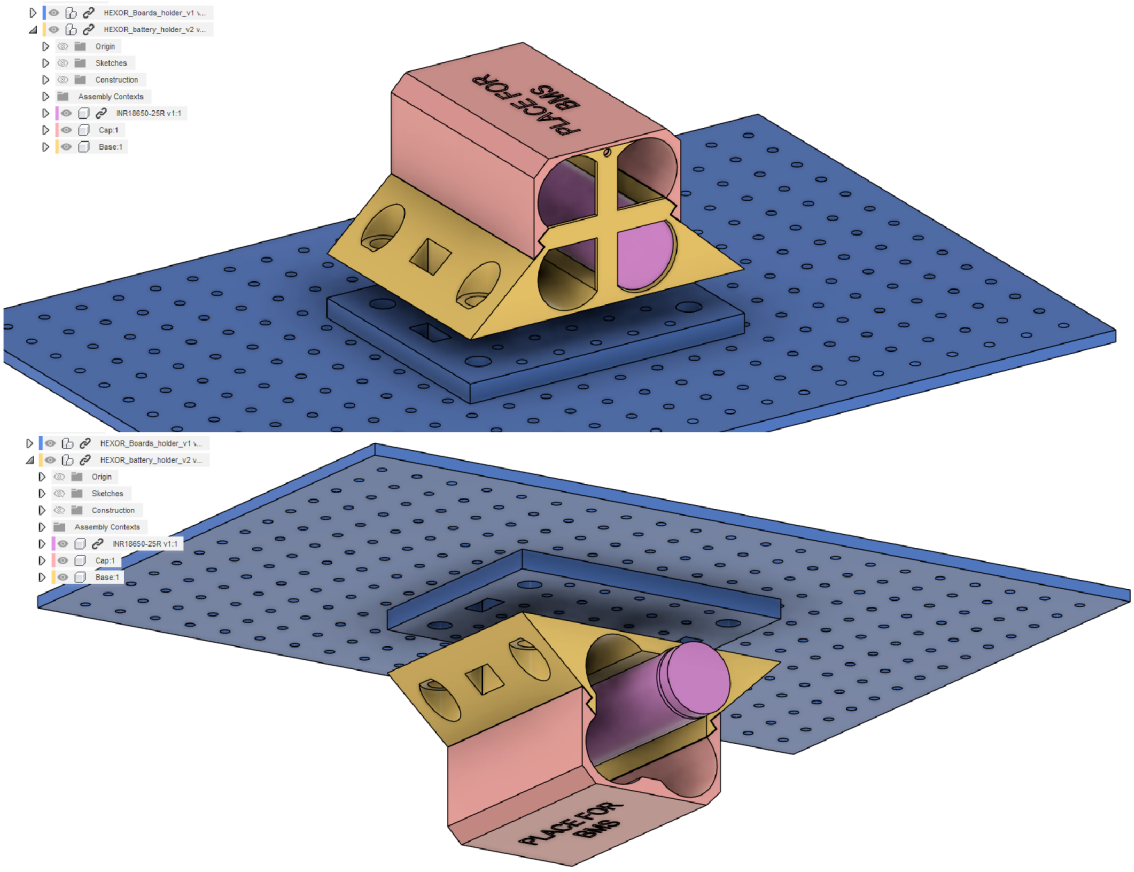
Montování nových desek na robota skýtá několik problémů. Kostra robota má hodně volného místa mimo jiné díky odebrání nabíjecí desky, avšak i ta byla uchycena pouze pomocí stahovacích pásků, baterie také pomocí elektrikářské pásky. Proto by bylo vhodné nějaké elegantnější řešení. Pomocí 3D tisku budou vytvořena a na robotu instalována lepší místa na uchycení jednotlivých modulů nebo případných dalších nových dílů do budoucna.

7.1 3D tisk

Variant 3D tisku za poslední roky přibývá – tisk z různých materiálů, různými metodami, s různou cenou tisku. Nejdostupnější variantou 3D tisku je stále tisk metodou FDM, tedy metodou modelování tištěného dílu pomocí přidávání materiálu taveným vláknem. I metody FDM se následně dělí podle materiálů nebo principu přidávání materiálu. V rámci dílen na TULab je možné využít povětšinou tiskáren od české společnosti Průša, případně tiskárnu od společnosti Creality s větším formátem tiskové plochy. Tyto tiskárny tisknou většinou z pevných plastových strun (typu ABS, PLA, PETG a dalších podobných), povětšinou jednou extruderovou hlavou i tryskou, ale je možnost tisknout i z více materiálů pomocí automatických výměn materiálu. Další možností je využít tiskárnu s technologií tisku z tekutých plastů (pryskyřic/resinů) vytvrzovaných UV světlem. Jelikož byla možnost využít i tiskárnu vlastní, byla využita, přesto mohl proběhnout výběr typu mezi tiskem z pryskyřice s vysokou přesností na MSLA tiskárně a tiskem z PLA, TPU nebo PETG na tiskárně s menší přesností. U dílů, které bude potřeba vyrobit, je stále hlavním požadavkem cena a jednoduchost, jelikož neponesou nic extrémně těžkého ani nemusí být extrémně přesné, ani nebudou příliš velké. Vybranou metodou díky dostupnosti materiálů a jejich ceny, vzhledem k požadavkům na vytvořený díl, byla nakonec standardní FDM tiskárna Ender-3 a materiál PLA.

Nakonec byly vymodelovány a vytištěny nosiče a držáky na baterii a nové moduly a desky. Jako první došlo na modelování držáku baterie, protože by mohlo v budoucnu opět dojít na její výměnu, třeba i za stejný typ, tedy za baterii typu 18650. Byla vymyšlena a nakreslena dvě pouzdra, v nichž lze jednotlivé články vyměnit za nové bez nutnosti rozbalování izolepy, zároveň by snad mohl nový držák působit lépe než dvěma stahovacími pásky přichycená baterie k tělu robota. I když možnost přichycení pomocí stahovacích pásků byla do modelu také zakomponována, avšak mezera pro pásek může i nemusí být využita. V první iteraci návrhu byla

pouzdra dvě, pro první dva a druhé dva samostatné články, paralelní spojení mělo proběhnout až pomocí drátů. Po jejich umístění na tělo robota, však bylo zjištěno, že toto není vhodná varianta. Bylo totiž velmi pravděpodobné, že by se mohl kladný pól baterie náhodně dotknout kostry robota, jež je spojena s nulovým potenciálem, takže by vznikal zkrat. Proto byl návrh předělán na jedno pouzdro ze dvou oddělitelných částí. První část, základna, bude pevně spojena s tělem robota, druhá část je kryt, který drží druhou dvojici článků a modul BMS, čímž je zároveň dodatečně fixován k baterii.



Obrázek 7.1: 3D tištěné držáky na baterii a nové desky

Nejdříve byl záměr nový držák pomocí podložek a vytvořených děr přichytit přímo proti novým nebo starým dírám v kostře robota, avšak následně se ukázalo být lepší vytvořit protikus, který by zároveň mohl sloužit i jako nová část kostry, proti které by šly přichytit nové moduly a desky. Důvodem, proč něco takového tisknout, je hlavně rychlost v prototypování. Tento protikus by byl pevnější například ve formě z hliníku nebo karbonu, avšak jejich tvorba by trvala násobně déle. Navíc pokud by došlo ke zjištění, že toto řešení není dostatečně vyhovující, bylo by také dražší a opět zdlouhavější takový díl nechat vyrobit znovu.

8 Diskuze a možná vylepšení

Při tvorbě diplomové práce nabytými znalostmi, jež autor nemohl na robotu aplikovat, je samotné řízení motorů a jejich regulace. Původní řešení nepodporuje pozicování nohou, a tak není možné otáčet robotem při chůzi vpřed. Dalším příkladem nevyužitých, avšak nabytých zkušeností, je využívání zpětných vazeb z odometrie, jsou stále v původní formě od předchozího tvůrce. Dále nemohlo být plně využito nic, co souvisí s předáváním řídicích signálů až robotovi, kde bylo v plánu aplikovat metody používané například u (FPV) dronů, například protokol ELRS a jemu podobné.

Při pracích provedených na robotu a převážně při navazování na původní program a hardware se autor práce poučil, jak velmi mohou následným majitelům robota usnadnit práci dobrá dokumentace a struktura a komentáře v programu. Mnoho informací se i na internetu může ztratit, a tak by bylo dobré většinu zdrojů ukládat společně s prací na přiložené CD či jiný nosič, aby byly zachovány i v případě, že domény, z nichž je citováno, v budoucnu nebudou dostupné. Taktéž díky práci vychází najevo, jak výhodné je použití univerzálních modulů, které dovolují větší rozsah změn, a že je možné je využít i v úplně jiné formě, než jaká byla původně zamýšlená. Dobrým příkladem jsou právě deska HUBu a deska Senzorická. HUB byl navržen tak, že není problém změnit jeho účel, zatímco senzorická deska byla navržena naprosto specificky na konkrétní účel, z nějž se velmi špatně předělává pro jinou než zamýšlenou aplikaci.

Pro budoucí práce na robotovi tak zbývá hodně prostoru nejen v místech, kterým se tato práce musela vyhnout, ale dost možná i v místech, jimž se věnovala. S novou řídicí deskou by bylo možné využít potenciál senzorů, které jsou již zabudovány, i senzorů, jež by mohly být připojeny k nové desce, která zůstává převážně nevyužita. V ideálním případě všechno dohromady. Konstrukce robota je robustní, a tak by například mohl být doplněn o lidar. Nebo by jím mohl být dokonce nahrazen ultrazvukový senzor, jenž je umístěn na téměř nejhorším možném místě, na velmi nestabilní špici ocasu, na které dochází k největším akceleracím, a tím je měření vzdálenosti jistě negativně ovlivněno.

9 Závěr

Jako zhodnocení práce z pohledu autora by se dalo říci, že autor se poučil a naučil se mnoho nových věcí v různých odvětví robotiky, avšak dost možná je neaplikoval tak, jak by si sám představoval vzhledem k tomu, že jejich aplikace musela být přizpůsobena původnímu řídicímu programu. Nemohly být adaptovány žádné nové pohyby robota, takže ani způsob chůze, ani způsob prohlížení okolí. Tedy ani výsledky nedosahují formy, k níž bylo směřováno na počátku práce. S nemožností změn v této oblasti se zároveň zhoršují možnosti, jimiž lze robota řídit.

V průběhu tvorby této práce se vyskytla řada problémů. Prvním byla nekompatibilita senzorické desky se stávajícími systémy robota. Tato senzorická deska měla původně nahradit desku řídicí. Na tuto nekompatibilitu se přišlo až v době, kdy už proběhla celkem rozsáhlá rešerše na způsoby mapování, trasování a dalších částí řízení samostatného pohybu robota v prostoru. Většina této práce musela být kvůli nekompatibilitě zahozena, protože nešla do řízení promítnout. Části informací, které byly získány v rámci rešerše, samostudia i výuky, byly využity při vývoji webové aplikace pro ovládání robota, avšak některé musely být rovnou se zjištěním nekompatibility smazány.

Dalším problémem byla nesourodost některých informací popsaných v pracích, které se robotu věnovaly, a skutečnosti. Bylo tedy potřeba dost času trávit pročítáním původních programů, v nichž některé části nemusely být úplně kompletní. Stejně tak byl i problém s dohledáním některých starších dokumentací, například dokumentace k robotu je dostupná jen částečně a popis řídicí desky robota vůbec nebyl nalezen. I vzhledem ke změnám, které byly během let na robotu provedeny, by bylo vhodné tyto informace uchovávat v předchozích pracích, jež se robotu věnovaly, avšak k tomu nedochází, ale ne vždy je to vůbec možné.

Zároveň skutečnost, že komunikace probíhá trochu jinak, než jak tvůrce popisoval, způsobovala, že vývoj nových funkcí byl velmi zpomalován nutností pročítat program jak ovládací aplikace, tak i samotného robota. Sice ne přímo v rámci zadání diplomové práce, ale po domluvě s vedoucím, proběhla dohoda o nevytváření úplně nové desky pro řízení. Nakonec nedošlo na návrh nové řídicí desky s čipem z řady obsažené v zadání, protože měla být snaha využít senzorickou desku, o které až při snaze začít nastavovat její hardware vyšlo najevo, že její využití jako řídicí desky nebude možné, a tím bylo rozhodnuto, že zůstane původní řídicí deska a čip ze zadání, jenž má být součástí řízení, bude na desce HUBu. Společně s tím muselo proběhnout přepsání původního programu tak, aby mohl být opět nahrán do senzorické desky. Nestačilo jen vzít původní program a nahrát ho v původní formě, protože jak bylo popsáno, nebyl kompatibilní s vývojovým prostředím. V návaznosti na to musely

být přizpůsobeny a doladěny některé další funkce a musela být přidána další, pro robota nová deska (HUB). Pro tu musel být vymyšlen systém předávání zpráv mezi separátními UARTy. Na dvou ze čtyř UARTů jsou připojeny komunikace, jež obě můžou předávat povely. Musel být vymyšlen systém, v němž bude nejdříve rozhodnuto o tom, který UART, respektive které zařízení, bude tím, jehož povely robot uposlechne. Vzhledem k zachování původní řídicí desky robota bylo upřednostněno původní ovládání, u něhož popis a reálná implementace místy nekoresponduje. Bylo nutné zkoumat všechny zdroje, a oproti původnímu plánu tak získalo vyšší prioritu zachování původního ovládání. Výsledkem tedy je, že hlavní část ve formě vývoje nového řízení je funkční asi z poloviny.

Hlavní úkol ve formě nového řídicího programu byl splněn spíše oklikou, protože nebylo vyvinuto zcela kompletní nové řízení. Bylo vytvořeno nové ovládání a zároveň byla zachována kompatibilita s původním ovládáním, včetně snahy o podobnosti z hlediska prostředí, aby pro operátory zvyklé na původní ovládání bylo snazší přejít do nového prostředí. Přesto však některé funkce nebyly dostatečně ozkoušeny a některé ještě nebyly implementovány, protože popis některých funkcí nebyl nalezen a jejich integrace v programu nebyly pochopeny (například prezentační režim). Proto je kupříkladu automatický režim implementován jako převod na manuální režim, který však nepodporuje některé zpětné vazby. Zbylá část zadání, vytvoření řídicího programu za pomoci čipů STM32, je implementována v podobě HUBu, který zajišťuje výměnu dat a tvorbu povelů.

Bod o aktualizaci baterie byl splněn. Po průzkumu v odvětví baterií a porovnání s výsledky předchozích písemných prací i fyzického řešení vyšlo najevo, že moc velké změny nejsou zapotřebí. Byla pořízena nová elektronika, která by teoreticky měla být lepší než ta, jež byla instalována doposud. K nákupu místo vývoje vlastního řešení bylo přistoupeno jak vzhledem k rychlosti, tak vzhledem k cenám, které trh nabízí a k nimž by se vývoj, tedy návrh a testování nové desky, nemohl přiblížit. A také stále platilo pravidlo, že bude snaha nevytvářet nové desky.

Všechny části, které byly během této práce z robota sundány, budou spolu s ním stále dostupné. Přestože byly z robota demontovány, nestalo se tak pro nefunkčnost, ale jen jako náhrada či aktualizace. Tím tedy zůstává možnost například provést testy porovnání kvalit nových dílů z čínského tržiště proti dílům, které byly instalovány dříve. Zatím totiž byly porovnány maximálně jejich parametry od výrobců, podle nichž vychází lépe díly nové.

Použitá literatura

[1] ĎAĎO, Stanislav a Marcel KREIDL. *Senzory a měřicí obvody*. 1. vyd. Praha: České vysoké učení technické, 1996. ISBN 80-01-01500-9.

[2] *ESP32 pinout* [online]. [cit. 2024-05-08]. Dostupné z: esp32.com.

[3] *ESP32-S2 Datasheet*. 1. vydání. Espressif systems, 2020. Dostupné také z: https://www.espressif.com/sites/default/files/documentation/esp32-s2_datasheet_en.pdf.

[4] *ESP32WROOM32D & ESP32WROOM32U: Datasheet*. v2.4. vyd. Espressif systems, 2023. Dostupné také z: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.

[5] *ESPAsyncWebServer* [online]. Bulharsko: Espressif Systems, 2020 [cit. 2020-08-23]. Dostupné z: https://github.com/me-no-dev/ESPAsyncWebServer.

[6] CHATILA, R. a J. LAUMOND. Position referencing and consistent world modeling for mobile robots. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. 1985, sv. 2, s. 138–145. Dostupné z DOI: 10.1109/ROBOT.1985.1087373.

[7] KOYANAGI, Fernando. *ESP32: Internal Details and Pinout* [online]. Brasil: Autodesk, 2020 [cit. 2020-08-26]. Dostupné z: https://www.instructables.com/id/ESP32-Internal-Details-and-Pinout/.

[8] KREDBA, Jan. *Řídicí systém senzorického subsystému mobilních robotů* [online]. Liberec, 2017 [cit. 2024-05-08]. Dostupné z: https://dspace.tul.cz/handle/15240/23511. diplomová práce. Technická univerzita v Liberci.

[9] LOZANO-PEREZ. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*. 1983, roč. C-32, č. 2, s. 108–120. Dostupné z DOI: 10.1109/TC.1983.1676196.

[10] MAGID, Evgeni, Roman LAVRENOV a Ilya AFANASYEV. Voronoi-based trajectory optimization for UGV path planning. In: *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*. 2017, s. 383–387. Dostupné z DOI: 10.1109/ICMSC.2017.7959506.

[11] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. 1. vydání. Praha: BEN - technická literatura, 2005. ISBN 80-7300-141-1.

[12] *Robot Motion Planning and Control*. 1. vyd. Springer Berlin, Heidelberg, 13 November 1997. ISBN 978-3-540-76219-5.

[13] SAJKOWSKI, M., T. STENZEL a B. GRZESIK. Walking robot HEXOR® II - a versatile platform for engineering education. In: *2008 13th International Power Electronics and Motion Control Conference*. 2008, s. 956–960. Dostupné z DOI: 10.1109/EPEPEMC.2008.4635391.

[14] *St.com - STM32F407VG* [online]. 2024. [cit. 2024-04-26]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32f407vg.html.

[15] *St.com - STM32F722RE* [online]. 2024. [cit. 2024-04-26]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32f722re.html.

A Přílohy

A.1 Obsah vloženého balíku do IS/STAG TUL

- Text práce
- Obrazová dokumentace
- Zdrojové kódy
- 3D modely

A.2 Popis datové struktury

Název	Data typ	Velikost ve struktuře v paměti	Velikost v bytech podle datového typu	Poznámka	Interpretace paměti			
					1B	1B	1B	1B
mode	uint8_t	1	1	Aktivní mód robota				
operátor	uint8_t	1	1	Priorita ovládání				
changeInPosition	boolean	1	1	Aktivní, pokud má docházet k pohybu				
changeInRecData	boolean	1	1	UART přijal změnu v datech (zatím nevyužito)				
mainOperatorActive	boolean	1	1	Zvolený hlavní operátor je aktivní				
changeToOtherOperatorInputs	boolean	1	1	Operátor předal řízení				
mainOperatorTimeout	boolean	2	1	Vypršela doba reaktivity operátora (může být spojeno s předchozím)				
timeToUnlockOperatorInput	Integer	4	4	Když klesne na nulu, může být provedena výměna operátora (v milisekundách)				
move	uint8_t	1	1	Směr pohybu (pro povel pohybu)				
moveTait	uint8_t	1	1	Směr pohybu ocasu				
numberOfScanAngles	uint8_t	2	1	Počet úhlů (180/numberOfScanAngles), ve kterých provede ultrazvukový senzor měření				
mag[3]	float	12	12	-3,14 až 3,14 úhel otočení okolo jednotlivých os (z, y, x)	3x4			
magDeg[3]	float	12	12	Úhly otočení okolo jednotlivých os (z, y, x) ve stupních	3x4			
position[3]	uint16_t	8	6	x, y, rotace pozice v prostoru od středu mapy (základ - [50,50])	3x2			
setSpeed	integer	4	4	Rychlost v procentech				
destPoint[3]	integer	12	12	Poloha nastaveného cíle	3x4			
recalcPath	boolean	4	1	Žádost o přepočítání trasy (nevyužito díky velkému výkonu výpočtu trasy v PC)				
path[300]	integer	1200	1200	Střídání x, y složky bodů trasy (délka = 150 bodů, může být i delší, ale pro šetření místem zatím omezeno)	300x4			
pathPointsOfInterest[50]	integer	200	200	Body vyhodnocené jako důležité. Většinou body, ve kterých dochází ke změně směru (rohy překážek)	50x4			
vectorToNextPoint[3]	integer	12	12	Vektor k následujícímu důležitému bodu	3x4			
vectorToGoal[3]	integer	12	12	Vektor k cílovému bodu	3x4			
obstacleMeasurementCount	uint8_t	4	1	Počet měření v závislosti na numberOfScanAngles				
obstacleDistance[32]	uint8_t	32	32	Pole 180/numberOfScanAngles => úhel + vzdálenost v mapě	32x1			
obstaclePositionAngle[3]	integer	12	12	Pozice detekované překážky v prostoru (úhel)	3x4			
obstaclePositionDistance	integer	4	4	Vzdálenost překážky z předchozího parametru				
batteryState	uint8_t	2	1	Stav baterie v procentech				
batteryVoltage	uint16_t	2	2	Napětí baterie				
batteryCurrent	uint16_t	2	2	Proud odebraný z baterie				
sensors	uint16_t	2	2	Taktilní senzory (tykadla), IR senzory jednotlivé bity				
tailAngleZ	integer	4	4	Úhel natočení ocasu kolem svislé osy (z, yaw)				
tailAngleY	integer	4	4	Úhel natočení ocasu kolem vodorovné osy (y, pitch)				
CELKOVÁ VELIKOST ODESÍLANÉ STRUKTURY	bot_struct_t	**1560**	1549					

paměť do jednoho 32b celku
doplněk do 32b (nevyužitá paměť)
více než jeden 32b celek (pole > 32b)