ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra řídicí techniky K-13135

Využití platformy mBot pro návrh robotického podvozku

Using the mBot platform for robotic chassis design

Bakalářská práce

Studijní program: Kybernetika a robotika

Vedoucí práce: Ing. Jan Havlík, Ph.D.

Lukáš Daněk

Praha 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Daněk** Jméno: **Lukáš** Osobní číslo: **474504**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití platformy mBot pro návrh robotického podvozku

Název bakalářské práce anglicky:

Using the mBot platform for robotic chassis design

Pokyny pro vypracování:

1) Seznamte se s problematikou ovládání robotických podvozků a robotickou platformou mBot. Robotická platforma mBot je vybavena jednoduchým podvozkem a řídicí elektronikou na bázi Arduino Uno s procesorem ATmega 328.
2) Navrhněte úpravy a nutné dovybavení robotické platformy mBot, které budou umožňovat rychlý a přesný pohyb v prostoru na hladkém zpevněném povrchu. Návrh sestavy má vycházet z dobře dostupných součástí a umožnit pozdější rozšíření o další senzory a aktuátory.
3) Realizujte navržené úpravy.
4) Proveďte experimentální ověření funkčnosti realizovaného robotického podvozku.

Seznam doporučené literatury:

[1] mBot documentation. Online: https://www.makeblock.com/support/ps-mbot [26. 1. 2020].
[2] Trobaugh, J. J.: Winning Design! LEGO MINDSTORMS NXT Design Patterns for Fun and Competition. Apress, Berkeley, CA, 2010. ISBN 978-1-4302-2965-0
[3] Pérula-Martínez, R., García-Haro, J.M., Balaguer, C. et al.: Developing Educational Printable Robots to Motivate University Students Using Open Source Technologies. J Intell Robot Syst (2016) 81: 25

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Havlík, Ph.D., katedra teorie obvodů FEL

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

Datum zadání bakalářské práce: **03.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

Ing. Jan Havlík, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta

Prohlášení

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

V Praze dne 21. května 2020

Poděkování

Chtěl bych poděkovat svému vedoucímu, panu Ing. Janu Havlíkovi, Ph.D., za odborné vedení mé práce.

Abstrakt

Cílem této práce bylo navrhnout a realizovat úpravy jednoduché robotické stavebnice mBot, které umožní přesný a rychlý pohyb na souřadnice na hladkém povrchu. Bylo nutné vybrat vhodnou metodu určování pozice robotu a vhodný způsob komunikace mezi senzory a řídicí deskou mBotu. Důležitými požadavky na výsledné řešení byly jak dobrá dostupnost a nízká cena použitých senzorů, tak i nízká výpočetní náročnost určení polohy. Klíčové bylo, aby vznikl modul, který bude možné snadno připojit k robotu mBot pomocí jeho standardních součástí.

Vzhledem k volbě odometrie za použití optických senzorů a komunikace po I2C sběrnici se tyto požadavky podařilo splnit. Výsledné řešení dosahuje požadované přesnosti, neboť relativní chyby vztažené vůči ujeté vzdálenosti jsou v řádu malých jednotek procent. Navržený modul je možné bez mechanických úprav připojit k libovolné stavebnici mBot. Použitý software umožňuje jednoduchou úpravu parametrů výpočtů pro určení polohy a regulačních konstant použitého regulátoru. Výsledný modul bude využíván na fakultě pro výukové a prezentační účely.

Klíčová slova

pohyb, souřadnice, robot, odometrie, optický senzor, Arduino, mBot

Abstract

The main goal of this work was to develop upgrades for simple robot kit mBot, which will allow this robot fast and precise movement to coordinates on a smooth surface. It was necessary to choose a suitable method for robot positioning and a suitable way of communication between sensors and mBot's control unit. Important requirements for the final solution were good availability and low price of the used sensors, as well as low computational difficulty. The main idea was to develop a module, which can be easily connected to the mBot robot using only it's standard components.

Because of the chosen method for robot positioning (i.e. odometry based on measurements from optical sensors) and communication via I2C bus, all the requirements were met. Solution presented in this work achieves required accuracy, which means that relative errors related to the distance traveled are equal to small units of percents. The developed module can be mounted to any mBot robot without any mechanical changes. Software of the module allows easy adjustment of parameters for positioning calculations and regulator's constants. Final module will be used at the CTU FEE for educational and presentation purposes.

Keywords

movement, coordinates, robot, odometry, optical sensor, Arduino, mBot

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Problematika pohybu robotu v prostoru je i přes velký rozmach techniky v současné době stále aktuální. Nejen v Evropě existuje mnoho robotických soutěží, ve kterých se bez spolehlivého robotického podvozku schopného přesného a rychlého pohybu nelze obejít. Tyto soutěže jsou často zaměřené na řešení komplexních úkolů typu průzkum oblasti a lokalizace zájmových objektů. Pro určování pozice robotu je často využíváno více druhů senzorů, např. enkodéry, LIDAR, GPS, ultrazvukové senzory nebo optické senzory. Kompletní robotické podvozky jsou rychlé, přesné, mechanicky odolné, ale také drahé a úzce specializované.

S rozvojem robotiky a jejím postupným pronikáním do každodenního života je důležité, aby se s touto problematikou mohla seznámit i neodborná veřejnost. K tomuto účelu slouží robotické stavebnice, jako jsou Lego Mindstorms či Fischertechnik. Tyto stavebnice jsou však stále poměrně drahé, možnosti jejich dalšího rozšíření bývají omezené a programování není většinou možné jinak než pomocí výrobcem definovaných bloků. Již několik let jsou velmi oblíbené open-source platformy Arduino a Raspberry Pi. Existuje řada projektů, které tyto platformy využívají pro projektovou výuku. Hlavním účelem tohoto způsobu výuky je nejen motivovat studenty přicházet s novými nápady na řešení různých problémů z oblasti robotiky, ale také jim dát možnost tyto nápady realizovat.

Jeden z projektů využívajících platformu Arduino je např. Developing Educational Printable Robots to Motivate University Students Using Open Source Technologies [1]. V rámci tohoto projektu byl vytvořen modulární, cenově dostupný robot, jehož součásti lze tisknout na 3D tiskárně. Zároveň s tímto novým výukovým robotem byly provedeny experimenty zkoumající možnosti projektové výuky. Jako více teoretický projekt lze potom doporučit knihu Winning Design! LEGO MINDSTORMS NXT Design Patterns for Fun and Competition [2]. Její autor s pomocí stavebnice Lego Mindstorms popisuje různé návrhy šasi robotu a radí, jak navrhovat snadno připojitelné moduly nebo jak řešit problémy spojené s navigací robotu a detekcí překážek.

Jedním z modulárních robotů založených na platformě Arduino je také mBot společnosti Makeblock [3]. Tento robot standardně nedisponuje žádnými senzory, které by mu umožnily provádět odometrii, a není proto schopen řízeného pohybu na souřadnice. Cílem této práce bylo proto navrhnout a realizovat snadno připojitelný modul, který odometrii a pohyb na souřadnice robotu mBot umožní. Motivací byla nejen možnost využít získané zkušeností z účastí v robotických soutěžích, ale také fakt, že výsledné řešení bude v případě vhodného provedení využito k výukovým a prezentačním účelům na fakultě. Současní studenti nebo zájemci o studium si tak v ideálním případě budou moci snadno vyzkoušet nejen samotné připojení modulu a jeho použití, ale také např. vliv regulačních konstant na pohyb robotu.

Kapitola 1

Platforma mBot

Platforma mBot je produktem společnosti Makeblock, využívá prvky open-source platformy Arduino. Její celý název mBot, Educational Robot Kit napovídá, že se jedná o výukovou stavebnici určenou spíše pro lidi s méně zkušenostmi z oblasti robotiky, zejména pro děti. Základní, nejjednodušší model stavebnice mBot lze v současnosti sehnat v českých obchodech přibližně za cenu 2500 Kč, nicméně konkrétní model mBot V1.1 byl poskytnut vedoucím práce.

Spolu s hardwarovými díly jsou v balení také návod pro sestavení [4] a krátký průvodce základním softwarem [5]. Sestavení robotu a otestování implementovaného softwaru zabralo jen několik desítek minut. Objevil se drobný problém s mechanicky poškozeným motorem, který však bylo možné snadno vyměnit za funkční motor z jiné stavebnice. Podrobně popsaný je mBot V1.1 např. v článku [6], proto je níže uveden jen základní přehled hardware a software a informace, které jsou důležité pro tuto práci.

1.1 Mechanické součásti a hardware

Základní balení stavebnice mBot V1.1 obsahuje zejména tyto mechanické části robotu:

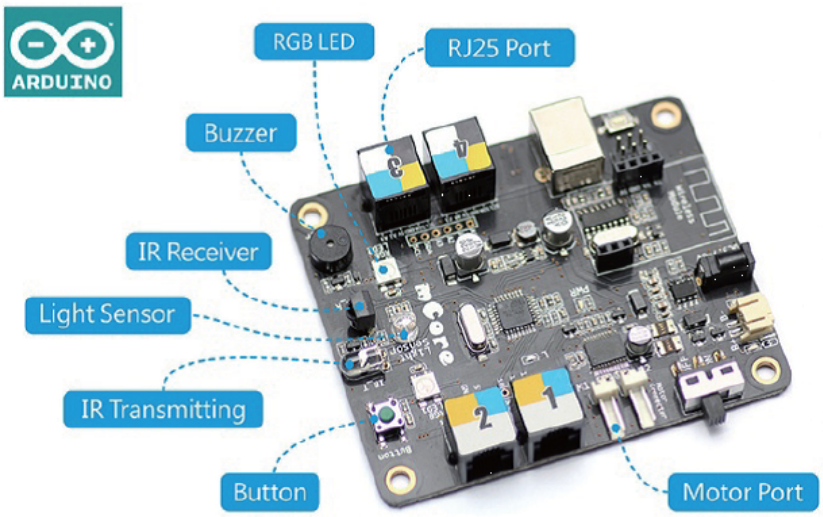
- hliníkovou konstrukci o rozměrech 165 x 90 x 30 mm,
- dva stejnosměrné motory s úhlovou převodovkou,
- dvě plastová kola o průměru 65 mm s gumovými kroužky pro zlepšení adheze,
- jedno plastové kolo, které je v podélném směru otočné a v příčném směru kluzné,
- čtyři hliníkové sloupky pro uchycení řídicí desky mCore,
- držák pro připojení čtyř bateriových článků velikosti AA.

Co se týká elektroniky, v balení se nachází:

- hlavní řídicí deska mCore navržená na základě desky Arduino UNO,
- modul pro bezdrátovou komunikaci prostřednictvím Bluetooth,
- ultrazvukový snímač vzdálenosti a infračervený senzor pro sledování čáry,
- infračervený dálkový ovladač.

Pro připojení motorů k desce mCore slouží konektory JST a pro připojení mCore k počítači je použit konektor USB (na straně počítače typu A a na straně mCore typu B). Motory robotu nejsou osazeny žádnými senzory, a řídicí deska proto nemá zpětnou vazbu ohledně jejich pohybu. Pro připojení dalších elektronických modulů (ultrazvukový snímač, infračervený senzor) je deska mCore vybavena čtyřmi konektory RJ25 se šesti piny. Základní konfiguraci robotu lze doplnit dvěma dalšími moduly připojenými přes zbývající konektory RJ25.

Dále je na desce vyvedeno šest pinů jako header pro připojení rozhraní SPI. Toto rozhraní bylo použito při návrhu prototypu vyvíjeného modulu. Pro finální provedení bylo nutné použít I2C sběrnici a konektor RJ25, aby bylo možné výsledný modul snadno připojit k libovolné desce mCore.



Obrázek 1.1: Řídicí deska mCore [6]

1.2 Software

Řídicí deska mCore má z výroby nahraný firmware realizující komunikaci s mobilním telefonem přes rozhraní Bluetooth. Z mobilního telefonu je po instalaci příslušné aplikace možné ovládat základní součásti a funkce robotu jako např. bzučák nebo RGB LED diody, aktivovat sledování čáry nebo řídit robota pomocí virtuálního joysticku.

Programování desky mCore je realizováno buď v prostředí mBlock (grafické programování vycházející z jazyka Scratch), nebo v prostředí Arduino IDE založeném na jazyku C. Pro účely této práce byla zvolena druhá varianta s použitím oficiální knihovny k robotu mBot [7]. Po navázání komunikace mezi deskou mCore a počítačem byla otestována funkčnost základních příkazů implementovaných v této knihovně. V novém programu byly využity standardní knihovny jazyka C a knihovny Arduino, jako je např. knihovna pro komunikaci přes SPI.

Kapitola 2

Výběr vhodné metody určení polohy

V praxi se používá několik základních principů určení polohy robotů:

- odometrie spočívající v měření ujeté vzdálenosti a změny směru pohybu,
- triangulace založená na měření azimutu k tzv. majákům (aktivním nebo pasivním orientačním bodům), použitá i v GPS a podobných systémech,
- pokročilé metody využívající skenování okolního prostoru (kamery, LIDAR, ultrazvuk),
- kombinace využívající výhody výše uvedených principů.

Odometrie se od ostatních metod liší tím, že závisí výhradně na existenci a vlastnostech povrchu, po kterém se robot pohybuje. Její výhodou je nezávislost na ostatních vlastnostech okolního prostředí (např. překážky, stěny, rozmístění majáků) a jejich časové proměnlivosti. Nevýhodou je pak sumace chyby měření v závislosti na ujeté vzdálenosti. Metoda je zcela závislá na nepřetržitém kontaktu s povrchem a poloha robotu je vždy relativní vůči počátku, který musí být definován externě.

Ostatní metody se vyznačují časově nezávislou maximální chybou, a tedy dobrou opakovatelností určení polohy. Jejich zřejmou nevýhodou je však nutnost využití vlastností okolního prostředí nebo komunikovat s externím zařízením. Pozici robotu musí být vždy možné určit jednoznačně pouze z informací z použitých senzorů (např. skenování okolního prostoru nelze použít v příliš homogenním prostředí).

2.1 Odometrie

Nejčastějším způsobem realizace odometrie je přepočet úhlu otočení kol robotu na ujetou vzdálenost a změnu směru pohybu. Nejjednodušší a nejdostupnější senzory jsou enkodéry umístěné přímo na hřídelích motorů. Toto řešení je technicky jednoduché, ale např. při protočení kol dojde ke znehodnocení informace o poloze. Proto je vhodnější umístit enkodéry na oddělená, volně se odvalující kola. Takové řešení je však konstrukčně složité, protože dotyk podvozku s povrchem ve více než třech bodech vyžaduje volné zavěšení kol umožňující kopírovat i drobné nerovnosti. Přesnost tohoto řešení je potom závislá na precizním mechanickém provedení.

2.2 Triangulace

Triangulace využívá aktivní nebo pasivní orientační body umístěné na předem definovaných souřadnicích. V průmyslových prostředích jsou často používané malé rádiové nebo ultrazvukové vysílače. Robot pak musí být vybaven odpovídajícím přijímačem s možností určení směru dopadu signálu. V případě pasivních orientačních bodů, např. optických značek, je nutné použít kameru a rozpoznávání obrazu, což výrazně zvyšuje výpočetní náročnost metody. Příkladem triangulace s aktivními majáky jsou geolokační systémy (GPS, Galileo, Glonas), které dosahují s dostupnými přijímači přesnosti v řádu jednotek metrů.

2.3 Skenování okolního prostoru

Pro skenování prostoru je opět možné použít kamery. Rozpoznávání obrazu je potom však ještě náročnější než v případě triangulace, neboť nestačí detekovat pouze předem známé objekty. Alternativou je LIDAR, který využívá odrazu laserových paprsků od překážek (princip Time-of-Flight). Úlohu rozpoznávání obrazu lze v určitých případech zjednodušit skenováním prostředí pouze v jedné horizontální rovině. To je obvyklé použití jednoduchých LIDAR senzorů v malých robotech. Dnes jsou již takové senzory poměrně dostupné, ale jejich cena je vyšší než např. cena jednoduchých kamer. Jako alternativu je možné použít ultrazvukové senzory, které mají však nevýhodu v malé rychlosti šíření zvuku a nemožnosti přesného určení směru, ze kterého přichází odražený zvuk. Z tohoto důvodu se používají spíše na detekci blízkých překážek.

2.4 Výběr vhodné metody

Dřívější účast na několika robotických soutěžích poskytla cenné zkušenosti s určováním polohy robotů na hladkém povrchu s přesností na jednotky centimetrů. V těchto projektech nebylo možné spoléhat na vlastní infrastrukturu (majáky) ani zkoumat vlastnosti okolního prostředí, které nebylo předem známé. Zároveň nešlo použít GPS, protože její přesnost není dostatečná. Nejefektivnějším řešením proto byla odometrie využívající enkodéry umístěné na volně se odvalujících kolech.

Při výběru způsobu určení polohy v případě této práce neplatila výše uvedená omezení, a bylo tak možné uvažovat o výběru libovolné metody. Omezení byla dána nízkým výpočetním výkonem CPU mBotu a snahou, aby výsledné řešení nebylo příliš komplikované a drahé. Proto byly z výběru vyřazeny skenování okolního prostoru pomocí kamer a kombinace více metod. Protože mBot je konstruován pro pohyb po hladkém povrchu, typicky ve vnitřních prostorech, hlavními kandidáty zůstaly odometrie a LIDAR. Byl brán v úvahu historický vývoj počítačových myší, u kterých bylo mechanické snímání pomocí enkodéru nahrazeno optickým skenováním povrchu. Na následujících stranách je popsáno několik podstatných výhod a nevýhod enkodérů, optických senzorů a LIDAR senzorů.

2.4.1 Enkodéry

Princip enkodérů je poměrně známý a není nutné ho v této práci podrobně popisovat. Určování polohy obvykle spočívá v čítání pulzů generovaných přerušováním mechanických kontaktů nebo infračerveného paprsku. Signál enkodéru je nejčastěji zpracováván v kvadraturním dekodéru, což umožňuje určit rychlost i směr rotace.

Výhody:

- dobrá dostupnost a nízká cena,
- často zabudované přímo v motorech (není případ mBotu),
- velmi nízká výpočetní náročnost (kvadraturní dekodér je zabudován přímo v některých mikrokontrolérech).

Nevýhody:

- znehodnocení informace o poloze při prokluzu kol v případě umístění na hnanou osu,
- komplikovaná mechanická konstrukce náročná na preciznost provedení v případě použití nezávislého zavěšení snímacích kol.

2.4.2 Optický senzor

Optický senzor polohy pracuje na jednoduchém principu snímání povrchu v zorném poli a vyhodnocování posunu obrazu mezi jednotlivými snímky. Tento typ senzoru je velmi dobře znám z počítačových myší, kde je využito LED diody (viditelné, infračervené nebo laserové světlo). Určení změn polohy v pravoúhlém souřadnicovém systému je realizováno v senzoru samotném, proto je jeho použití snadné a nenáročné na výpočetní výkon. Senzor je navíc možné velmi dobře ochránit před vnějšími vlivy (nečistoty, mechanické poškození, ...).

Výhody:

- nízká cena a dobrá dostupnost při použití masově vyráběných senzorů z myší,
- rychlé a přesné určení polohy s nízkou výpočetní náročností,
- jednoduchá mechanická konstrukce a snadná zástavba na podvozek robotu,
- snímání pohybu ve dvou kolmých směrech → pro robot stačí použít jen jeden senzor (na rozdíl od enkodérů, kde jsou nutné alespoň dva),

Nevýhody:

- při použití kompletního senzoru včetně optiky z počítačové myši nutnost snímat povrch z velmi malé vzdálenosti (jednotky milimetrů),
- vyšší cena a horší dostupnost specializovaných senzorů pro průmyslové použití (snímací vzdálenost 3 - 5 cm),
- nespolehlivé snímání na nevhodných površích (např. členité mimo rozsah zaostření optiky nebo příliš lesklé).

2.4.3 LIDAR senzor

LIDAR je poměrně moderní způsob detekce objektů a měření vzdálenosti k nim. Jedná se o použití principu Time-Of-Flight, kdy je měřen čas od vyslání laserového pulzu do jeho přijetí na stejném místě. Na základě této doby je potom vypočítána vzdálenost k objektu, od kterého se paprsek odrazil. LIDAR je v současnosti také hojně využíván v automobilovém průmyslu, především k detekci překážek.

Existují projekty, které LIDAR používají pro určení polohy robotu, a to i přímo na elektrotechnické fakultě ČVUT. Robot si s pomocí otáčejícího se senzoru vytváří mapu okolního prostředí a na základě výskytu specifických objektů určuje svou pozici. Je klíčové, aby robot disponoval dostatečným výpočetním výkonem.

Výhody:

- při dostatečném výpočetním výkonu rychlé vytvoření přesného modelu okolního prostředí,

Nevýhody:

- velmi vysoká cena v případě průmyslových senzorů schopných skenování 360° okolí,
- nutnost mechanické konstrukce zajišťující rotaci levnějších dálkoměrů,
- výpočetně náročné určení polohy.

Cílem této práce bylo vytvořit modul, který bude levný, snadno dostupný a příliš nenaruší jednoduchost celé stavebnice mBot. LIDAR nesplňuje žádný z těchto požadavků, proto byl z výběru vyřazen. Z důvodu komplikované mechanické konstrukce enkodérů byl proto nakonec zvolen optický senzor. Dalším argumentem pro tuto volbu byl fakt, že snímání polohy počítačové myši je snímání pohybu malého robotu na hladkém povrchu velmi podobné. Navíc i běžně dostupné optické senzory jsou velmi přesné (rozlišení posunu v řádech milimetrů) a určení polohy je rychlé. Kvalita snímání rozdílných povrchů se sice s různě drahými senzory liší, avšak i současné levné počítačové myši fungují dobře např. na hladkém stole.

Kapitola 3

Optický senzor polohy

3.1 Výběr vhodného senzoru

Při výběru konkrétního optického senzoru byly brány v úvahu dvě možnosti zmíněné v předchozí kapitole. První z nich bylo pořízení profesionálního optického senzoru vyráběného např. pro robotické vysavače a dodávaného s patřičnou optikou, díky které může snímat povrch z výšky přibližně 3 - 5 cm. Druhou možností bylo koupit levnou počítačovou myš a použít její senzor, který měl dle předpokladů pro dané účely svou přesností i rychlostí dostačovat.

Zvolený postup práce vedl na sestavení ověřovacího vzorku na univerzální desce plošného spoje s využitím komunikace přes SPI, aby bylo možné nejdříve ověřit, zda je vybraný senzor dostatečně přesný a spolehlivý. Oba typy zmíněných senzorů umožňují komunikaci přes SPI, ale vzhledem k požadavku na použití dostupných a levných senzorů byl zvolen optický senzor z počítačové myši.

Zajímavý článek o tomto typu optického senzoru je například Mouse Cam [8]. Jeho autor popisuje, jak je možné získat obraz ze senzoru Agilent ADNS-2610, který byl běžným senzorem v mnoha počítačových myších přibližně před deseti lety. Tento senzor však dle katalogového listu [9] zvládá rychlost pohybu maximálně 12 ips (cca 30 cm/s) a rozlišení 400 cpi. Navíc je v současné době již téměř nedostupný. Proto byl zvolen senzor PMW3325 od firmy PixArt, který je zabudován v současné herní myši prodávané přibližně za 250 Kč. Bylo očekáváno, že jeho výhodou bude kromě rychlosti 100 ips (cca 254 cm/s) a rozlišení 5000 cpi zejména spolehlivá funkce na různých površích. Jeho základní specifikace jsou dostupné na webu výrobce [10] a v katalogovém listu [11] (stručný přehled je v příloze B). Dalším krokem byla analýza komunikace mezi senzorem a procesorem zabudovaným v myši.

3.2 Komunikace se senzorem přes SPI

Pro analýzu komunikace byl použit logický analyzátor, který při vhodném nastavení umožňoval textové zobrazení hodnot přenášených bajtů. Skutečná komunikace byla porovnána s popisem v katalogových listech [11] a [12] (senzor PMW3310 je podobný senzoru PMW3325, ale jeho katalogový list je podrobnější). Tím byly identifikovány sekvence zpráv odpovídající navázání komunikace po zapnutí napájení a vyčtení změn polohy. V pozorované komunikaci se však vyskytovaly další zprávy, jejichž význam nebylo možné určit. Bylo proto předpokládáno, že souvisejí např. s identifikací zařízení nebo komunikací

s ovladačem v PC. Z tohoto důvodu byly v implementaci kódu pro tuto práci ignorovány a byla sestavena následující komunikační sekvence:

1. při zapnutí robotu je provedena tzv. power-up sekvence: čekání 50 ms a zápis hodnoty 0x5A do registru POWER_UP_RESET (0x3A),
2. následuje čekání 50 ms a zápis hodnoty 0x39 do registru 0x18,
3. po dalších 50 ms proběhne čtení z registru PRODUCT_ID (0x00),
4. čekání 250 ms, aby čidlo mělo dostatek času začít pracovat (experimetálně zjištěná hodnota),
5. nyní je umožněna běžná komunikace s čidlem; v případě této práce opakované čtení z registru BURST_MOTION_READ (0x16).

V registru BURST_MOTION_READ je 6 bytů obsahujících poslední informace o změně souřadnic od minulého vyčtení, a to v následujícím formátu (po bytech zleva):

1. MOTION: nejvyšší bit (7) indikuje, zda došlo k pohybu, bit 5 indikuje, zda došlo k přílišnému oddálení senzoru od snímaného povrchu (a tím ke znehodnocení dat),
2. DELTA_X_L: nižší byte změny v souřadnici x,
3. DELTA_X_H:, vyšší byte změny v souřadnici x,
4. DELTA_Y_L: nižší byte změny v souřadnici y,
5. DELTA_Y_H:, vyšší byte změny v souřadnici y,
6. SQUAL: informace o tom, kolik je v obrazu validních dat (Surface quality).

3.3 Připojení senzoru k řídicí desce mCore

Řídicí deska mCore použitá v robotu mBot pracuje při napětí 5 V (viz A), ale senzor PMW3325 potřebuje napájení 1,8 V až 2,1 V. Umí však pomocí vnějšího zdroje napětí pracovat s logickými úrovněmi až do 3,3 V (zjištěno měřením na myši - není v katalogovém listu). Bylo proto použito jen jedno napájení a vyřešen přímý převod úrovní 2 V $\leftrightarrow$ 5 V.

Existují hotové a cenově dostupné moduly umožňující obousměrný převod napěťových úrovní. Tyto převodníky ale využívají jednoduché zapojení tranzistoru s pracovním odporem, a proto byla na místě obava z pomalých hran při frekvenci SPI 1 MHz. Po konzultaci s odborníkem byly proto použity jednosměrné integrované převodníky určené přímo pro převod rychlých číslicových signálů. Tyto převodníky jsou však dostupné pouze ve velmi malých pouzdrech, a nebylo tak možné jednoduché ověření s využitím nepájivého pole. Návrh i realizace převodníku byly proto provedeny s odbornou pomocí na univerzální desce plošného spoje. Toto řešení umožnilo snadné upevnění senzoru na konstrukci mBotu. Schéma výsledného zapojení je uvedeno v příloze C.

Kapitola 4

Pohyb na souřadnice

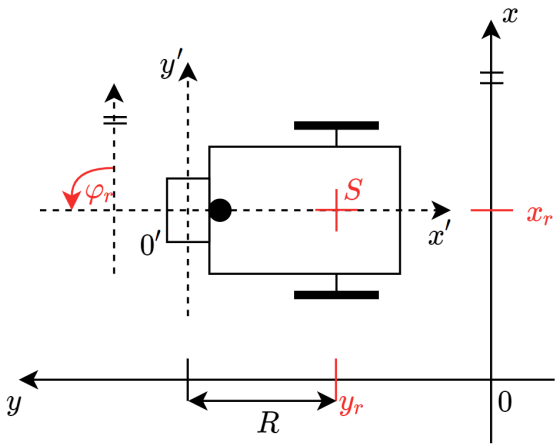
4.1 Výpočet souřadnic z informací od senzoru

Robot vybavený optickým senzorem má v rovině 3 stupně volnosti (souřadnice středu osy kol $S(x_r, y_r)$ a úhel φ_r). Senzor robotu měří změny v souřadnicích x' a y' a následnými přepočty lze získat globální souřadnice x_r, y_r. Orientace globálního souřadnicového systému byla zvolena tak, aby vhodně odpovídala pohybu robotu. V momentě zapnutí robotu směřuje jeho šasi v kladném směru osy x, kladný směr osy y směřuje doleva a bod S se nachází v počátku. Globální souřadnicový systém je vždy vztažen k počáteční pozici robotu. Zvolená orientace souřadnicového systému zachovává pravotočivou bázi. Pro lepší představu viz obrázek 4.1.

Senzor je umístěn v přední části robotu (před kluzným kolem). Spojnice bodu S a senzoru je za předpokladu, že se kola nesmekají do stran, tečnou k trajektorii pohybu robotu. Trajektorii tak lze při dostatečně častém čtení změn polohy ze senzoru aproximovat sledem krátkých pohybů po úsečkách s otáčením kolem středu S. Podle této aproximace se bod umístění senzoru pohybuje po kružnici se středem v bodě S, a proto má změna v souřadnici x' význam změny polární souřadnice r a změna v souřadnici y' význam změny polární souřadnice φ. Pro malé změny úhlu φ je možné považovat změnu v souřadnici y' za změnu obloukové míry.

Kvůli orientaci senzoru zvolené při jeho zabudování bylo nutné použít vyčtené hodnoty změn polohy s opačným znaménkem. Protože hodnoty změn polohy jsou v registru BURST_MOTION_READ vyjádřeny v počtu pixelů, bylo také nutné zjistit rozlišení senzoru v pixelech na mm. Měřením byla stanovena jeho hodnota ve směru osy x jako $res_x = 100$ pixelů na mm a ve směru osy y jako $res_y = 87$ pixelů na mm.

Následující obrázek znázorňuje volbu souřadnicového systému:



Obrázek 4.1: Zvolený souřadnicový systém robotu

Výpočet změn polárních souřadnic je popsán těmito rovnicemi:

$$\Delta r = \frac{-\Delta x'}{res_x} = \frac{-\Delta x'}{100}, \tag{4.1}$$

$$\Delta\varphi = \frac{-\Delta y'}{res_y \cdot R} = \frac{-\Delta y'}{87 \cdot 94.67} \approx \frac{-\Delta y'}{8236}, \tag{4.2}$$

kde $R = 94,67$ mm je vzdálenost bodu umístění senzoru od středu S, $\Delta x'$ a $\Delta y'$ jsou hodnoty v pixelech získané čtením registru BURST_MOTION_READ.

Úhel otočení robotu φ_r je potom součtem dosavadních změn úhlu φ:

$$\varphi_r = \sum \Delta\varphi \tag{4.3}$$

a po každém přičtení je provedena jeho normalizace na interval $[-\pi, \pi]$. Ta je vyřešena prostým odčítáním nebo přičítáním 2π, neboť pro pohyb na souřadnice je podstatný pouze směr, ve kterém je robot natočen v rámci globálního souřadnicového systému. Souřadnice x_r a y_r pak lze dopočítat následovně:

$$\Delta x = \Delta r \cos\varphi_r, \tag{4.4}$$

$$\Delta y = \Delta r \sin\varphi_r, \tag{4.5}$$

$$x_r = \sum \Delta x, \tag{4.6}$$

$$y_r = \sum \Delta y. \tag{4.7}$$

Pro výpočty spojené s určováním polohy byla otestována implementace s proměnnými typu double a knihovními funkcemi sin(), cos(). Po dokončení programu se ukázalo, že jeden výpočet polohy robotu (souřadnice x_r, y_r a úhel φ_r) trvá méně než 300 μs. To se jevilo jako dostatečně rychlé, což potvrdilo další testování. Funkce, která souřadnice robotu tímto způsobem aktualizuje je pojmenována *calc_coords*() a jejími vstupy jsou změny v souřadnicích x' a y' vyjádřené v pixelech.

4.2 Jednoduchý PSD regulátor na směr pohybu

Po experimentálním ověření správnosti výpočtů souřadnic robotu byl implementován kód pro regulaci pohybu určitým směrem. Cílem bylo vytvořit jednoduchý PSD regulátor jako funkci, která pomocí změn výkonu na motorech srovná robot do požadova-ného směru. Tato funkce byla pojmenována *calc_power_by_direction*() a jejím vstupem je kromě požadovaného úhlu α v radiánech také druhá mocnina vzdálenosti k cílovému bodu d. Úhel α určuje směr od aktuální pozice robotu (x_r, y_r) k cílovému bodu (x_c, y_c) a je normalizován na interval $[-\pi, \pi]$. Druhá mocnina vzdálenosti je použita z důvodu snížení výpočetní náročnosti.

Nejdříve je vypočtena odchylka úhlů $e = \alpha - \varphi_r$, která je normalizována na interval $[-\pi, \pi]$. Následně je vynásobením odchylky regulačními konstantami (K_p, K_{p2}, K_s, K_d) získána hodnota rozdílu výkonů motorů. Konstanta K_{p2} je proporcionální složka, která se násobí druhou mocninou odchylky se zachováním jejího znaménka. Její použití se ukázalo v dřívějších projektech jako vhodné, neboť napomáhá regulaci při příliš velké počáteční odchylce bezprostředně po zahájení pohybu k cílovému bodu. Sumační konstanta K_s je pro dosažení nulové regulační odchylky v případě robotu mBot nutná, protože je třeba kompenzovat rozdílné rychlosti motorů při shodném požadovaném výkonu. Hodnoty konstant byly určeny experimentálně jako $K_p = 30000$, $K_{p2} = 1000$, $K_s = 30$, $K_d = 0$. Rozdíl výkonů motorů pro i. cyklus algoritmu je pak spočítán jako

$$\Delta P_i = K_p \cdot e_i + K_{p2} \cdot e_i \cdot |e_i| + K_s \cdot \sum_{j=1}^{i} e_j + K_d(e_i - e_{i-1}). \tag{4.8}$$

Z dřívějších zkušeností vyplynulo, že při snaze o co nejrychlejší pohyb na souřadnice není řízení obou motorů najednou vhodný způsob, protože velmi často dochází k saturaci rychlejšího motoru na maximálním výkonu. Proto jsou akční zásahy prováděny pouze snížením výkonu o spočítaný rozdíl ΔP_i na odpovídajícím motoru (dle znaménka spočítaného rozdílu výkonů ΔP_i). S přibližováním robotu k cílovému bodu se zvyšuje rychlost změny odchylky, proto je výkon na rychlejším motoru P_0 omezován v závislosti na vzdálenosti robotu od cílového bodu.

Závislost výkonu P_0 na druhé mocnině vzdálenosti k cílovému bodu d je následující:

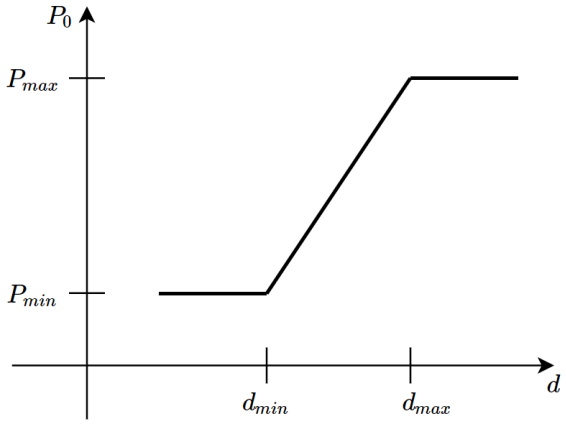
$$P_0 = P_{min} + a \cdot (d - d_{min}), \tag{4.9}$$

kde a je směrnice určená jako

$$a = \frac{P_{max} - P_{min}}{d_{max} - d_{min}} \tag{4.10}$$

a $P_{min}, P_{max}, d_{min}$, a d_{max} jsou konstanty.

Hodnota P_0 je navíc omezena na interval $[P_{min}, P_{max}] = [150, 255]$ pro vzdálenosti menší než $d_{min} = 50^2$ mm a větší než $d_{max} = 100^2$ mm. Hodnota výkonu 255 odpovídá maximálnímu výkonu motoru mBotu a hodnota 150 byla určena experimentálně, stejně jako d_{min} a d_{max}.



Obrázek 4.2: Závislost výkonu P_0 na druhé mocnině vzdálenosti k cílovému bodu d

Také byly zavedeny saturace $\sum e \in [-100, 100]$ a $\Delta P \in [-P_0, P_0]$. Suma odchylek je samozřejmě nulována při každém úspěšném dosažení cílového bodu.

4.3 Kompletní výpočet regulačního zásahu

Výpočet regulačního zásahu je v principu poměrně jednoduchý a je implementován ve funkci *regulate*(), jejímiž parametry jsou souřadnice cílového bodu x_c, y_c. V té se nejprve pomocí funkce *calc_coords*() vypočítají aktuální souřadnice robotu. Následně se spočítá druhá mocnina vzdálenosti mezi současnou polohou robotu (x_r, y_r) a cílovým bodem (x_c, y_c) a úhel k cílovému bodu α pomocí funkce atan2():

$$d = (x_c - x_r)^2 + (y_c - y_r)^2, \tag{4.11}$$

$$\alpha = \text{atan2}(y_c - y_r, x_c - x_r). \tag{4.12}$$

Nakonec je zavolána funkce *calc_power_by_direction*() a vypočtené výkony jsou aplikovány na motory robotu. Funkce *regulate*() je opakovaně volána s periodou T_s, dokud vypočtená druhá mocnina vzdálenosti k cílovému bodu není menší než druhá mocnina tolerované minimální vzdálenosti. Perioda T_s definuje časovou konstantu regulátoru a celého systému (jedná se o vzorkovací periodu). Její hodnota byla určena na základě dřívějších zkušeností jako $T_s = 10$ ms. Tolerance zbývající vzdálenosti k cílovému bodu byla po testování zvolena jako 20 mm.

Kapitola 5

Realizace samostatného modulu

5.1 Návrh a realizace desky plošných spojů

Protože se předpokládalo, že výsledný modul bude dále používán na fakultě, bylo vhodné navrhnout desku plošných spojů s možností kusové, příp. hromadné výroby. Návrh byl realizován s odbornou pomocí v prostředí Eagle (https://www.autodesk.com/...), které je pro nekomerční účely dostupné zdarma.

Vzhledem k požadavkům na jednoduchost a snadné připojení modulu ke stavebnici mBot byla komunikace senzoru PMW3325 a desky mCore realizována prostřednictvím rozhraní I2C. Toto rozhraní je na desce mCore dostupné na standardních konektorech RJ25. Nabízela se možnost využít nějaký integrovaný převodník I2C $\leftrightarrow$ SPI, nebo připojit senzor k mikrokontroléru a realizovat komunikaci mezi senzorem a deskou mCore pomocí něj. Výhoda řešení s mikrokontrolérem je zejména přidaný výpočetní výkon, který lze využít pro výpočty spojené s určováním polohy robotu. Komunikace s deskou mCore potom může probíhat na vyšší úrovni (např. zadávání cílových bodů a potvrzování jejich dosažení, změna regulačních konstant). Jako vhodná volba se jevily mikrokontroléry řady STM32. Při porovnávání různých modelů z této řady vůči vhodným převodníkům I2C $\leftrightarrow$ SPI bylo zjištěno, že mikrokontroléry STM32 jsou výrazně dostupnější, přičemž nabízejí širokou škálu dalších využitelných periferií (např. USB, UART). Proto bylo vybráno řešení s mikrokontrolérem.

Z dostupných modelů u distributorů elektronických součástek (Farnell, TME) byl vybrán nejlevnější model s potřebnými periferiemi, konkrétně STM32F070F6P6. Tento mikrokontrolér je napájen z 3,3 V, kterému odpovídají i úrovně periferních signálů. Jak je uvedeno v kapitole 3.3, senzor PMW3325 je při správném napětí na pinu VDDIO s těmito úrovněmi kompatibilní. Bylo proto možné vynechat převodníky napěťových úrovní směrem k senzoru. Navíc, protože rozhraní I2C je realizováno piny s tolerancí 5V signálů, nejsou potřeba ani převodníky směrem k desce mCore. Bylo však nutné doplnit zdroj napětí 3,3 V pro napájení procesoru.

Napájení celého zapojení je zajištěno z desky mCore (5 V) kabelem s konektory RJ25, který je součástí stavebnice mBot. Na pin header JP4 (1x4) je vyveden UART mikrokontroléru používaný pro nahrávání i ladění programu. Před nahráváním programu je nutné procesor restartovat, k čemuž slouží pin header JP2 (1x2), a uvést do režimu BOOT pomocí pin headeru JP1 (1x2). Na volné GPIO piny mikrokontroléru byly přidány dvě signalizační LED diody. Výroba několika kusů navržené desky plošných spojů byla realizována u výrobce Gatema a jejich osazení provedeno ručně. V příloze D je uvedeno

schéma zapojení a v příloze E obrázky DPS se seznamem součástek.

5.2 Programování mikrokontroléru

Po osazení desky plošných spojů bylo třeba zvolit vhodné prostředí pro tvorbu a nahrávání programu v jazyce C. Jednou z možností bylo online prostředí Mbed (os.mbed.com). Avšak po přidání všech knihoven potřebných pro komunikaci (SPI.h, I2CSlave.h, Serial.h) zabíral výsledný program téměř celou flash paměť mikrokontroléru (32 kB). Bylo téměř jisté, že tuto variantu nebude možné použít, neboť bylo nutné do programu přidat ještě výpočty spojené s určováním polohy a regulací pohybu.

Jako lepší varianta se ukázalo prostředí Keil μVision 5 (www2.keil.com/mdk5) a knihovny HAL (High Abstraction Layer). Pro správnou konfiguraci mikrokontroléru a vygenerování šablony kódu, která definuje a inicializuje použité periferie, je určen software STM32CubeMX (https://www.st.com/en/...). Po provedení počátečního poměrně obsáhlého nastavení je možné začít používat funkce HAL knihoven, které umožňují komunikaci pod přerušením přes UART nebo na I2C sběrnici. Pro komunikaci se senzorem PMW3325 přes SPI nebylo přerušení použito, neboť výpočty pro určení polohy probíhají okamžitě po přijmu dat ze senzoru. Díky tomu nebylo nutné synchronizovat přerušení a hlavní smyčku programu ani řešit priority přerušení vzhledem k vysoké důležitosti obsluhy přerušení na I2C sběrnici. K nahrávání programu slouží software STM32CubeProgrammer (https://www.st.com/en/...). Úplná verze programu po zahrnutí všech potřebných knihoven a po implementaci výpočtů v datovém typu double zabírá výrazně méně místa než v případě prostředí Mbed (přibližně 24 kB).

5.3 Komunikace na I2C sběrnici

Jak již bylo zmíněno výše, výpočty spojené s určováním pozice a regulací pohybu byly přesunuty do mikrokontroléru. S tím souvisí chování mikrokontroléru v komunikaci; plní roli řízeného obvodu (tzv. slave) a reaguje na zasílané příkazy. Průběžně aktualizuje hodnoty pozice robotu a výkonu potřebného pro regulaci pohybu na cílový bod. Deska mCore poté může číst spočítané hodnoty výkonu a aplikovat je přímo na motory. Pro správný směr otáčení kol je nutné aplikovat na vyčtenou hodnotu výkonu znaménko odpovídající montáži motoru na šasi robotu. Výkony jsou spočtené podle vztahů uvedených v kapitole 4.2, a tedy jsou v potřebném rozsahu.

V programu STM32CubeMX byla definována role mikrokontroléru jako řízený obvod a zvolena jeho adresa jako 0x7c. Komunikace na sběrnici probíhá standardním způsobem. Řídicí obvod nejdříve v jedné zprávě zapíše adresu (7 bitů) a informaci o tom, zda bude následovat zápis, nebo čtení (bit R/$\overline{\text{W}}$), dále hodnotu příkazu a v případě zápisu také data. V případě čtení následuje zpráva od řízeného obvodu s požadovanými daty. Určité proměnné senzoru má smysl uvažovat pouze jako blok, neboť např. obě souřadnice cílového bodu je třeba zadat vždy současně. Tyto bloky jsou čteny/zapisovány jako celek podle hodnoty přijatého příkazu.

5.3.1 Seznam příkazů

Seznam definovaných příkazů je uveden v následující tabulce:

Příkaz	Hodnota	Blok dat
CMD_CONTROL	0	pohyb robotu
CMD_STATUS	1	stav modulu
CMD_VERSION	2	verze software
CMD_POSITION	3	souřadnice robotu
CMD_TARGET	4	souřadnice cílového bodu
CMD_POWER	5	výkon motorů
CMD_KP	6	regulační konstanta K_p
CMD_KP2	7	regulační konstanta K_{p2}
CMD_KS	8	regulační konstanta K_s
CMD_KD	9	regulační konstanta K_d
CMD_RESX	10	rozlišení senzoru v ose x
CMR_RESY	11	rozlišení senzoru v ose y
CMD_PHI0	12	parametr φ_0

Tabulka 5.1: Příkazy definované v použitém komunikačním protokolu

5.3.2 Struktura bloků

Program mikrokontroléru posílá při požadavku čtení vždy celý aktuální datový blok. V případě zápisu jsou proměnné s přístupem čtení ignorovány, je však nutné dodržet délku datového bloku. Pořadí proměnných je dáno jejich pozicí ve struktuře, čemuž odpovídá pořadí zapisovaných/čtených dat, viz následující tabulka:

Příkaz (blok)	Délka bloku [B]	Proměnné	Přístup
CMD_CONTROL	1	control	čtení a zápis
CMD_STATUS	1	status	čtení
CMD_VERSION	1	version	čtení
CMD_POSITION	8	pos_x, pos_y, pos_phi_deg, target_phi_deg	čtení a zápis čtení a zápis čtení a zápis čtení
CMD_TARGET	6	target_x, target_y, tolerance	čtení a zápis čtení a zápis čtení a zápis
CMD_POWER	3	power_control power_left power_right	čtení čtení čtení
CMD_KP	4	Kp	čtení a zápis
CMD_KP2	4	Kp2	čtení a zápis
CMD_KS	4	Ks	čtení a zápis
CMD_KD	4	Kd	čtení a zápis
CMD_RESX	2	res_x_div100	čtení a zápis
CMR_RESY	2	res_y_div100	čtení a zápis
CMD_PHI0	2	phi_0_deg_div1000	čtení a zápis

Tabulka 5.2: Datová struktura definovaná v použitém komunikačním protokolu

5.3.3 Popis proměnných a povolené hodnoty

Následující tabulka obsahuje typy jednotlivých proměnných spolu s omezením hodnot, kterých mohou nabývat, a popisem jejich významu:

Proměnná	Typ	Povolená hodnota	Význam
control	byte	MOVE_STOP (0x00) MOVE_START (0x01)	změň stav na STOPPED změň stav na ON_THE_MOVE
status	byte	NOT_READY (0x00) STOPPED (0x01) ON_THE_MOVE (0x02) POINT_REACHED (0x03)	modul není připraven stav zastaveno stav v pohybu stav dosažení cílového bodu
version	byte	libovolná	verze software
pos_x	short int	libovolná	souřadnice robotu x_r [mm]
pos_y	short int	libovolná	souřadnice robotu y_r [mm]
pos_phi_deg	short int	libovolná	souřadnice robotu φ_r [°]
target_phi_deg	short int	libovolná	azimut k cílovému bodu [°]
target_x	short int	libovolná	souř. cílového bodu x_c [mm]
target_y	short int	libovolná	souř. cílového bodu y_c [mm]
tolerance	short int	neomezeno v SW (doporučeno [20, 200])	tolerance vzdálenosti k cílovému bodu [mm]
power_control	byte	BRAKE (0) FORWARD (1) REVERSE (2)	zastavit pohyb vpřed pohyb vzad (v současné verzi SW nepoužito)
power_left	byte	libovolná	výkon levého motoru
power_right	byte	libovolná	výkon pravého motoru
Kp	int_32	libovolná	regulační konstanta K_p
Kp2	int_32	libovolná	regulační konstanta K_{p2}
Ks	int_32	libovolná	regulační konstanta K_s
Kd	int_32	libovolná	regulační konstanta K_d
res_x_div100	short	libovolná	rozlišení v x [$\frac{1}{100}$ p/mm]
res_y_div100	short	libovolná	rozlišení v y [$\frac{1}{100}$ p/mm]
phi_0_deg_div1000	short	libovolná	parametr φ_0 [$\frac{1}{1000}°$]

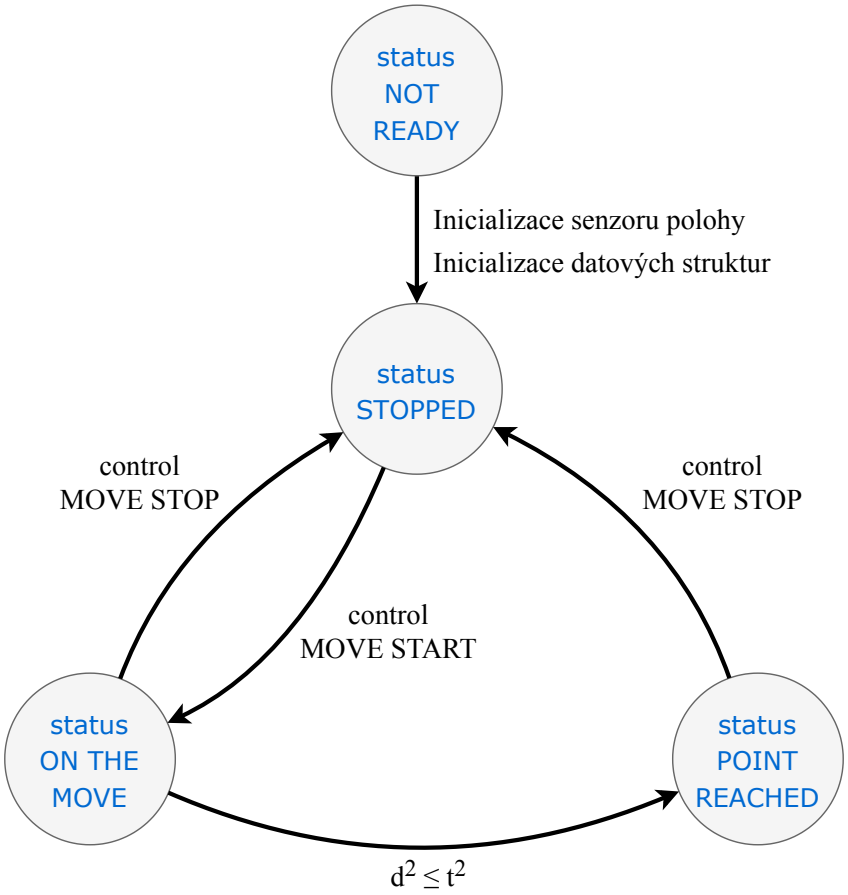
Tabulka 5.3: Proměnné definované v použitém komunikačním protokolu

5.3.4 Stavový diagram

Regulační část programu mikrokontroléru se může nacházet v několika stavech s ohledem na požadavek od řídicího obvodu a na aktuální pozici. Po spuštění se nachází ve stavu NOT READY, ve kterém probíhá inicializace senzoru, datových struktur a dalších periferií. Po inicializaci přejde do stavu STOPPED, ve kterém očekává zadání souřadnic cílového bodu, případně dalších parametrů pro určení polohy a regulaci.

Po zápisu hodnoty MOVE START do proměnné control přejde do stavu ON THE MOVE a do proměnných pro výkony motorů ukládá aktuální vypočtené hodnoty výkonu s periodou $T_s = 10$ ms. V případě dosažení okolí cílového bodu (viz podmínka pro toleranci vzdálenosti k cílovému bodu v kapitole 4.2) přejde do stavu POINT REACHED. Tento stav je možné opustit pouze do stavu STOPPED zápisem hodnoty MOVE STOP do proměnné control.

Hodnoty proměnných lze zapisovat pouze ve stavu STOPPED z důvodu zajištění správné synchronizace s probíhajícími výpočty. Následující diagram znázorňuje možné stavy a přechody mezi nimi:



Obrázek 5.1: Stavový diagram

Uvedený vztah $d^2 \leq t^2$ vyjadřuje splnění podmínky pro dosažení cílového bodu.

Kapitola 6

Výsledky práce

6.1 Přesnost určení polohy robotu

Informace získané ze senzoru polohy odpovídají pohybu robota v polárních souřadnicích (osa x - ujetá vzdálenost, osa y - změna úhlu, viz kapitola 4.1). Bylo proto rozhodnuto ověřit přesnost určení každé z těchto složek zvlášť.Byly zvoleny takové trajektorie pohybu, aby při měření ujeté vzdálenosti nastala co nejmenší změna úhlu a při měření změny úhlu co nejmenší změna vzdálenosti. V prvním případě šlo o přímku, v druhém o rotaci robotu kolem osy procházející bodem S (viz obrázek 4.1).

Měření vzdálenosti bylo realizováno spuštěním motorů robotu na pevnou dobu t s potřebným výkonem. Po zastavení byla odečtena změna souřadnic určená modulem a změřena skutečná poloha robotu milimetrovým měřidlem. Určená ujetá vzdálenost d_u a skutečná ujetá vzdálenost d_s byly poté vypočteny z Pythagorovy věty. V následující tabulce jsou uvedeny změny souřadnic robotu a ujeté vzdálenosti spolu s jejich absolutní chybou $e_a = d_u - d_s$ a relativní chybou $e_r = e_a/d_s$:

Doba [s]	Skutečná poloha [mm]			Určená poloha [mm]			Chyba [mm]	Chyba [%]
t	Δx_s	Δy_s	d_s	Δx_u	Δy_u	d_u	e_a	e_r
3	610	17	610,2	631	57	633,6	23,3	3,8
	614	15	614,2	636	63	639,1	24,9	4,1
	624	22	624,4	643	67	646,5	22,1	3,5
	623	19	623,3	641	61	643,9	20,6	3,3
	628	17	628,2	647	69	650,7	22,4	3,6
2	415	25	415,8	437	45	439,3	23,6	3,7
	420	15	420,3	442	33	443,2	23,0	5,5
	423	21	423,5	442	40	443,8	20,3	4,8
	414	30	415,1	434	49	436,8	21,7	5,2
	413	25	413,8	436	44	438,2	24,5	5,9

Tabulka 6.1: Měření přesnosti určení ujeté vzdálenosti

Měření přesnosti určení změny úhlu bylo provedeno spuštěním motorů na danou dobu t s výkony potřebnými k tomu, aby se robot otáčel na místě. Na šasi robotu byly umístěny značky, pomocí kterých bylo po zastavení robotu možné změřit konečný relativní úhel (v intervalu $\pm 180°$) za použití úhloměru s rozlišením 1°. Následně byla na základě znalosti počtu otočení robotu o 180° dopočítána skutečná absolutní změnu úhlu. Tento přepočet byl aplikován rovněž na změnu úhlu určenou senzorem, neboť i ta je počítána jako relativní v intervalu $\pm 180°$. Byla určena absolutní chyba $e_a = \Delta\varphi_{ru} - \Delta\varphi_{rs}$ a relativní chyba $e_r = e_a/\Delta\varphi_{rs}$. Výsledky jsou uvedeny v následující tabulce:

Doba t [s]	Skutečná $\Delta\varphi_{rs}$ [°]	Určená $\Delta\varphi_{ru}$ [°]	Chyba e_a [°]	Chyba e_r [%]
3	572	583	11	-1,9
	553	578	25	-4,5
	575	582	7	-1,2
	570	567	-3	0,5
	610	619	9	-1,5
2	384	374	-10	2,6
	388	392	4	-1,0
	355	358	3	-0,8
	393	395	2	-0,5
	375	377	2	-0,5

Tabulka 6.2: Měření přesnosti určení změny úhlu

Při měření ujeté vzdálenosti se ukázalo, že určené hodnoty Δy_{ru} a vzdálenosti d_u vykazují výraznou odchylku směrem do kladných hodnot (viz tabulka 6.1). Jako možná příčina byla určena změna rozlišení senzoru v důsledku jiné výšky nad povrchem oproti montáži ověřovacího vzorku bez mikrokontroléru. Změna výšky senzoru nad povrchem vede ke změně vzdálenosti reprezentované jedním pixelem, resp. celým snímačem, čímž se změní rozlišení senzoru. Z tohoto důvodu bylo zavedeno rozlišení senzoru pro výpočty spojené s určováním polohy jako parametr, který je možné zadat z řídicího obvodu.

I přes provedené korekce rozlišení vykazoval senzor konstantní odchylku v hodnotách y_{ru} při jízdě na delší vzdálenost. Přesnost měření změny úhlu navíc neodpovídala očekávané přesnosti vzhledem k surovým datům vyčteným ze senzoru (pomocí ladicí sériové linky). Tato odchylka byla vykompenzována pootočením souřadnicového systému senzoru oproti souřadnicovému systému robotu o konstantní úhel φ_0 pomocí rotační matice. Hodnotu φ_0 je možné zadat s ostatními proměnnými z řídicího obvodu.

V následujících tabulkách uvádím výsledky měření po provedení korekcí zmíněných na předchozí straně:

Doba [s]	Skutečná poloha [mm]			Určená poloha [mm]			Chyba [mm]	Chyba [%]
t	x_s	y_s	d_s	x_u	y_u	d_u	e_a	e_r
3,5	696	40	697,2	685	31	685,7	-11,4	-1,6
	695	-15	695,2	684	-18	684,2	-10,9	-1,6
	693	4	693,0	682	13	682,1	-10,9	-1,6
	704	38	705,0	696	22	696,4	-8,7	-1,2
	713	21	713,3	701	13	701,1	-12,2	-1,7
3	615	7	615,0	603	13	603,1	-11,9	-1,9
	613	38	614,2	602	25	602,5	-11,7	-1,9
	600	0	600,0	592	6	592,0	-8,0	-1,3
	612	10	612,1	600	18	600,3	-11,8	-1,9
	604	16	604,2	593	16	593,2	-11,0	-1,8
2	402	27	402,9	395	15	395,3	-7,6	-1,9
	399	29	400,1	392	17	392,4	-7,7	-1,9
	402	28	403,0	394	18	394,4	-8,6	-2,1
	399	26	399,9	390	16	390,3	-9,5	-2,4
	403	10	403,1	396	7	396,1	-7,1	-1,8
1	194	6	194,1	190	3	190,0	-4,1	-2,1
	198	9	198,2	194	7	194,1	-4,1	-2,1
	195	8	195,2	191	5	191,1	-4,1	-2,1
	198	10	198,3	192	5	192,1	-6,2	-3,1
	195	7	195,1	191	4	191,4	-4,1	-2,1

Tabulka 6.3: Měření přesnosti určení ujeté vzdálenosti po zavedení φ_0

Doba [s]	Skutečná $\Delta\varphi_{rs}$ [°]	Určená $\Delta\varphi_{rs}$ [°]	Chyba [°]	Chyba [%]
3,5	662	666	4	0,6
	660	663	3	0,5
	669	662	-7	-1,0
	654	656	2	0,3
	669	662	-7	-1,0
3	562	563	1	0,2
	561	565	4	0,7
	574	577	3	0,5
	579	582	3	0,5
	572	579	7	1,2
2	354	357	3	0,8
	370	371	1	0,3
	375	370	-5	-1,3
	374	370	-4	-1,1
	386	389	3	0,8
1	176	175	-1	-0,6
	172	172	0	0,0
	175	175	0	0,0
	175	173	-2	-1,1
	168	168	0	0,0

Tabulka 6.4: Měření přesnosti určení změny úhlu po zavedení φ_0

6.2 Přesnost pohybu na zadané souřadnice

Byla provedena další měření, která kromě přesnosti určení polohy zhodnotila také přesnost pohybu na zadané souřadnice. Ta je kromě přesnosti určení polohy závislá zejména na zvolených regulačních konstantách. Rovněž je nutné od požadované vzdálenosti k cílovému bodu odečíst zadanou toleranci. V momentě splnění této tolerance robot sice přestane regulovat a zastaví motory, ale nějakou dobu se ještě pohybuje v původním směru. Hodnoty těchto parametrů byly ponechány na hodnotách uvedených v kapitole 4.1.

6.2.1 Pohyb po přímce

Pro první měření byla zvolena požadovaná změna souřadnic tak, aby robot nemusel provádět žádné otáčení, tedy aby se pohyboval po přímce v kladném směru osy x:

$$(\Delta x_{r1}, \Delta y_{r1}) = (500, 0), \quad (\Delta x_{r2}, \Delta y_{r2}) = (250, 0), \tag{6.1}$$

resp. po odečtení tolerance

$$(\Delta x'_{r1}, \Delta y'_{r1}) = (480, 0), \quad (\Delta x'_{r2}, \Delta y'_{r2}) = (230, 0). \tag{6.2}$$

Pro souřadnici x_r byla stanovena absolutní chyba

$$e_{xa} = \Delta x_{ru} - \Delta x_{rs}, \tag{6.3}$$

kde Δx_{ru} je změna souřadnice x_r určená senzorem a Δx_{rs} je skutečná změna souřadnice x_r, a relativní chyba

$$e_{xr} = e_{xa}/\Delta x_{rs}. \tag{6.4}$$

Pro souřadnici y_r byla stanovena pouze absolutní chyba $e_{ya} = \Delta y_{ru} - \Delta y_{rs}$, protože tuto souřadnici regulátor během pohybu na zvolené souřadnice záměrně udržuje na nule, a relativní chyba by tak byla 100 %. Skutečná změna souřadnic robotu byla změřena milimetrovým měřidlem.

Požadované [mm]				Skutečné [mm]		Určené [mm]		Chyba [mm]		Chyba [%]
Δx_r	Δy_r	$\Delta x'_r$	$\Delta y'_r$	Δx_{rs}	Δy_{rs}	Δx_{ru}	Δy_{ru}	e_{xa}	e_{ya}	e_{xr}
500	0	480	0	500	6	495	0	-5	-6	-1,0
				500	-6	494	0	-6	6	-1,2
				499	-3	493	0	-6	3	-1,2
				500	-3	496	0	-4	3	-0,8
				497	-8	494	0	-3	8	-0,6
250	0	230	0	248	3	246	0	-2	-3	-0,8
				247	-3	244	0	-3	3	-1,2
				243	0	241	0	-2	0	-0,8
				247	2	245	0	-2	-2	-0,8
				249	2	248	0	-1	-2	-0,4

Tabulka 6.5: Měření přesnosti pohybu na souřadnice pro pohyb v přímce

6.2.2 Pohyb s otočením

Pro druhé měření byla požadovaná změna souřadnic zvolena tak, aby robot musel provést otočení a následný pohyb:

$$(\Delta x_{r1}, \Delta y_{r1}) = (-500, 0), \quad (\Delta x_{r2}, \Delta y_{r2}) = (-250, 0), \tag{6.5}$$

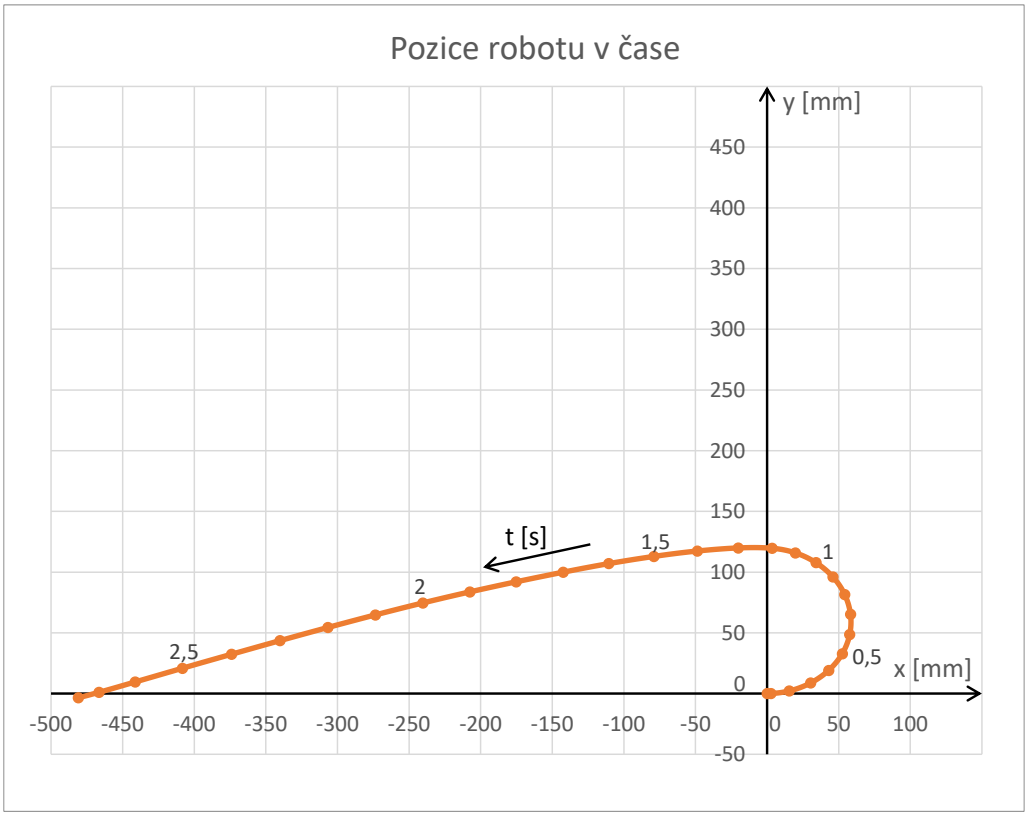
resp. po odečtení tolerance

$$(\Delta x'_{r1}, \Delta y'_{r1}) = (-480, 0), \quad (\Delta x'_{r2}, \Delta y'_{r2}) = (-230, 0). \tag{6.6}$$

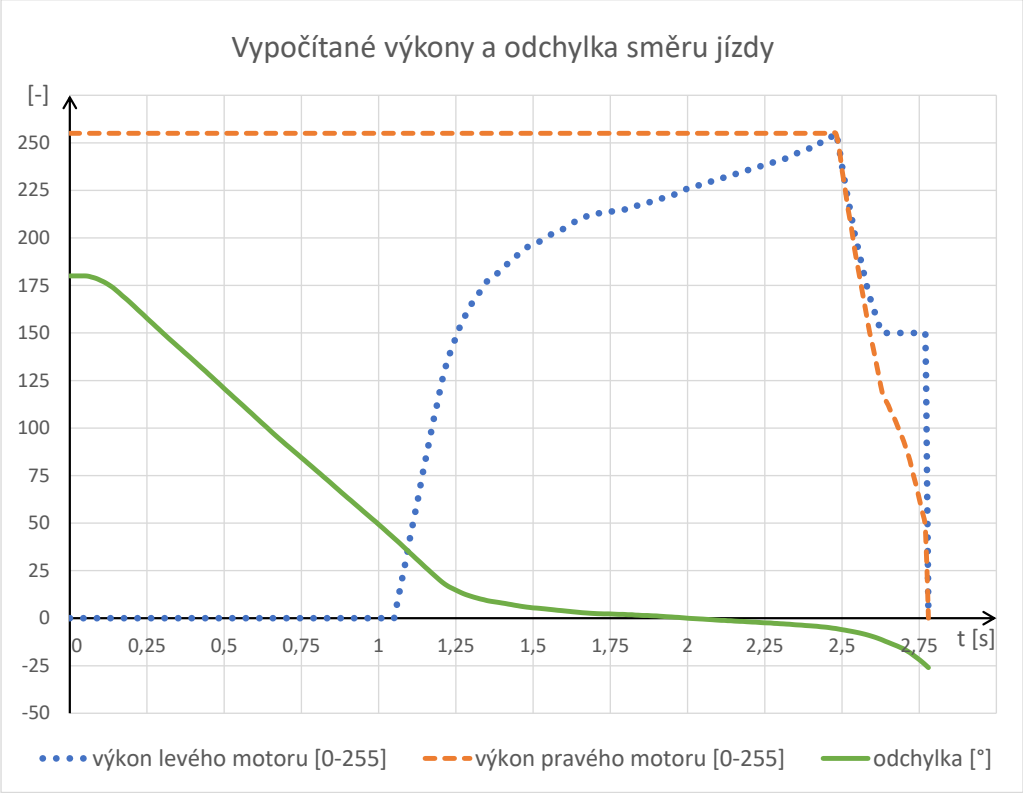
Absolutní chyby byly stanoveny analogicky k předchozímu měření. Relativní chyby si lze představit jako absolutní chyby vůči 50 centimetrům ujeté vzdálenosti, avšak jejich exaktní měření nebylo realizovatelné.

Požadované [mm]				Skutečné [mm]		Určené [mm]		Chyba [mm]	
Δx_r	Δy_r	$\Delta x'_r$	$\Delta y'_r$	Δx_{rs}	Δy_{rs}	Δx_{ru}	Δy_{ru}	e_{xa}	e_{ya}
-500	0	-480	0	-491	10	-485	5	6	-5
				-499	-9	-487	-10	12	-1
				-496	-12	-486	-7	10	5
				-487	-15	-490	-10	-3	5
				-515	-3	-506	-1	9	2
-250	0	-230	0	-249	-7	-242	-11	7	-4
				-251	0	-245	1	6	1
				-245	-10	-247	-15	-2	-5
				-239	-10	-244	-14	-5	-4
				-250	-9	-246	-12	4	-3

Tabulka 6.6: Měření přesnosti pohybu na souřadnice pro pohyb s otočením



Obrázek 6.1: Průběh souřadnic robotu při testovacím pohybu



Obrázek 6.2: Průběh vypočítaných výkonů a odchylky směru jízdy při testovacím pohybu

6.2.3 Pohyb ve čtverci

Pro třetí měření byly zvoleny body v rozích čtverce o straně délky jeden metr s vyhodnocením odchylky v posledním bodě (totožný s počátečním bodem). Tato trajektorie pohybu eliminuje chybu měření vzdálenosti, neboť se tato chyba kompenzuje na protilehlých stranách. Dále nezávisí na počátečním nasměrování robotu a převádí měření změny úhlu na měření vzdálenosti.

Díky velké ujeté vzdálenosti je možné měřit velmi malé změny úhlu, které jsou jinak silně ovlivněny nepřesností počátečního natočení robotu. Hlavním účelem tohoto měření bylo ověřit, že přesnost určení změny úhlu je dostatečná pro předpokládaný účel použití výsledného modulu.

Jako výchozí bod byl zvolen bod $(x_r, y_r, \varphi_r) = (0, 0, 0)$ a byly stanoveny pouze absolutní chyby určení změn souřadnic. Relativní chyby si lze představit jako absolutní chyby vůči čtyřem metrům ujeté vzdálenosti, avšak jejich exaktní měření nebylo realizovatelné.

Směr pohybu	Skutečné [mm]		Určené [mm]		Chyba [mm]	
	Δx_{rs}	Δy_{rs}	Δx_{ru}	Δy_{ru}	e_{xa}	e_{ya}
proti směru hod. ručiček	20	-5	-6	10	-26	15
	32	0	5	9	-27	9
	25	-15	0	5	-25	20
	22	25	0	6	-22	-19
	27	4	-1	6	-28	2
po směru hod. ručiček	17	-30	1	-7	-16	23
	-26	-28	0	-3	26	25
	5	5	-8	-11	-13	-16
	25	-24	-2	-7	-27	17
	27	-8	2	-6	-25	2

Tabulka 6.7: Měření přesnosti pohybu na souřadnice pro pohyb ve čtverci

6.3 Diskuze výsledků

Provedená měření prokázala, že robot je s navrženým modulem schopen pohybu na zadané souřadnice s relativní chybou vztaženou k celkové ujeté vzdálenosti v jednotkách procent (jednotky centimetrů na metr). Absolutní chyby vypočítaných souřadnic robotu vůči jeho skutečným souřadnicím se pohybují rovněž v jednotkách centimetrů. Podobně malé jsou i absolutní chyby skutečných souřadnic robotu po dosažení cílového bodu vůči souřadnicím cílového bodu.

Protože při výpočtech pro učení polohy robotu dochází k dvojitému převodu mezi kartézskými a polárními souřadnicemi, ukázalo se jako klíčové přesné určení změny úhlu natočení robotu. Při pohybu robotu v přímce je požadovaná změna úhlu nulová a relativní chybu přesnosti určení změny úhlu nemá smysl určovat. Zásadní vliv pak má absolutní chyba, která se násobí s ujetou vzdáleností. Oproti tomu vliv absolutní chyby v určení ujeté vzdálenosti se s ujetou vzdáleností zmenšuje.

Ukázalo se, že největší vliv na absolutní chybu v určení úhlu natočení robotu má přesnost montáže modulu na šasi robotu (natočení v horizontální rovině v důsledku vůle v montážních otvorech). Tento vliv byl kompenzován zavedením parametru φ_0, který je možné určit např. na základě výsledků měření pro jízdu v přímce.

Dále byly zavedeny parametry rozlišení senzoru v obou kolmých směrech (osa x, osa y), které umožňují nastavit skutečnou rozlišovací schopnost ovlivněnou především výškou montáže senzoru nad snímaným povrchem. Výška montáže by bez zavedení těchto parametrů musela být provedena s přesností na desetiny milimetrů. V přesnosti určení polohy robotu hrají také velkou roli drobné nerovnosti snímaného povrchu. Z těchto důvodů by bylo pro budoucí použití vhodné snímat povrch z větší vzdálenosti pomocí jiného senzoru nebo s využitím speciální optiky.

Z pohledu použití robotu pro výukové nebo prezentační účely se funkčnost navrženého modulu osvědčila. Pohyb robotu na zadané souřadnice je plynulý a dosahuje nejvyšší rychlosti, které je mBot schopen. Hodnoty regulačních konstant použitého regulátoru je možné měnit z řídicího obvodu, což umožňuje snadné experimentování s jejich nastavením. Je také možné používat modul pouze pro určení polohy a navrhnout vlastní regulátor v řídicím obvodu.

Modul v současné verzi nepodporuje pohyb na cílové body bez zastavení. Jako možné vylepšení lze proto navrhnout doplnění průjezdních bodů, což by umožnilo plynulejší a rychlejší pohyb po jiné trajektorii, než je přímka. Zajímavým vylepšením by také bylo umožnění řízení pohybu v zadaném směru, kdy by řídicí obvod mohl plynule ovládat robot a výpočty regulačních zásahů v reálném čase by stále probíhaly v modulu. Řídicí obvod by tak mohl na základě informací z dalších senzorů průběžně upravovat trajektorii pohybu robotu, např. kvůli neočekávané překážce na původní trajektorii.

Kapitola 7

Závěr

V této práci byl navržen a realizován snadno připojitelný modul pro stavebnici mBot umožňující určení souřadnic polohy robotu a zajišťující výpočetní podporu pro regulaci pohybu na zadané souřadnice. Z uvažovaných metod určování polohy byla vybrána odometrie s využitím optických senzorů z počítačových myší.

Modul je realizován na samostatné desce plošných spojů a pro montáž jsou využity otvory se závity, které jsou přítomné na šasi robotu mBot. Připojení k řídicí desce mCore je realizováno pomocí standardního rozhraní a dodávaných kabelů s konektory. Toto řešení má charakter přídavného modulu a nevyžaduje žádné úpravy stavebnice. Komunikace modulu s řídicí deskou mCore je velmi jednoduchá, probíhá na I2C sběrnici, jejíž obsluha je součástí knihoven Arduino. Kvůli mechanickým dispozicím není možné současně s tímto modulem použít modul pro sledování čáry. To však nepovažuji za významný nedostatek, neboť nový modul může sledování čáry nahradit řízením polohy na odpovídající souřadnice. Jiná omezení pro připojitelné moduly stavebnice nejsou.

Změřené hodnoty přesnosti určení polohy ukazují, že vybraná metoda je pro požadovaný účel, tj. pohyb po hladkém zpevněném povrchu, vhodná. Drobným nedostatkem je, že se použitý senzor z počítačové myši pohybuje velmi nízko nad snímaným povrchem. Kvůli tomu mohou i malé změny vzdálenosti senzoru od povrchu vést ke změně poměru mezi počtem pixelů a změřenou vzdáleností. Částečně lze tento vliv kompenzovat nastavením hodnot příslušných parametrů v programu. Navíc je pravděpodobné, že by ho bylo možné téměř odstranit snímáním z větší vzdálenosti od povrchu za použití vlastní optiky nebo specializovaných senzorů. Lze potom předpokládat, že by bylo možné snímat i hrubý povrch.

Seznam použité literatury

[1] Pérula-Martínez, R. – García-Haro, J. M. – Balaguer, C. – et al.: Developing Educational Printable Robots to Motivate University Students Using Open Source Technologies, *Journal of Intelligent & Robotic Systems*, [online], 20. 1. 2015. [cit. 20. 5. 2020], Dostupné z: https://raulperula.github.io/papers/Perula-Martinez2015a.pdf.

[2] Trobaugh, J. J.: *Winning Design! LEGO MINDSTORMS NXT Design Patterns for Fun and Competition*, Apress, Berkeley, CA, 2010. ISBN: 978-1-4302-2965-0.

[3] Makeblock: Webová stránka stavebnice mBot, [online], [cit. 23. 2. 2020]. Dostupné z: https://www.makeblock.com/mbot.

[4] Makeblock: Assembling manual, [online], [cit. 23. 2. 2020]. Dostupné z: http://cdnlab.makeblock.com/mBot%20Consrtruction%20Manual.pdf.

[5] Makeblock: Quick start guide, [online], [cit.23. 2. 2020]. Dostupné z: http://cdnlab.makeblock.com/mBot%20Quick%20Start%20Guide%20.pdf.

[6] Voda, Z.: Arduino robot mBot je tu! Co nabídne?, [online], 16. 2. 2015. [cit. 20. 5. 2020], Dostupné z: https://arduino.cz/arduino-robot-mbot-je-tu-co-nabidne/.

[7] Makeblock-official: Makeblock-Libraries, [online], [cit. 23. 2. 2020]. Dostupné z: https://github.com/Makeblock-official/Makeblock-Libraries.

[8] neelandan: Mouse Cam, [online], [cit. 15. 2. 2020]. Dostupné z: https://www.instructables.com/id/Mouse-Cam/.

[9] Avago Technologies: Katalogový list senzoru ADNS-2610, [online], [cit. 15. 2. 2020]. Dostupné z: https://media.digikey.com/pdf/Data%20Sheets/Avago%20PDFs/ADNS-2610.pdf.

[10] PixArt: Webová stránka k senzoru PixArt PMW3325, [online], [cit. 10. 2. 2020]. Dostupné z: https://www.pixart.com/products-detail/13/PMW3325DB-TWV1.

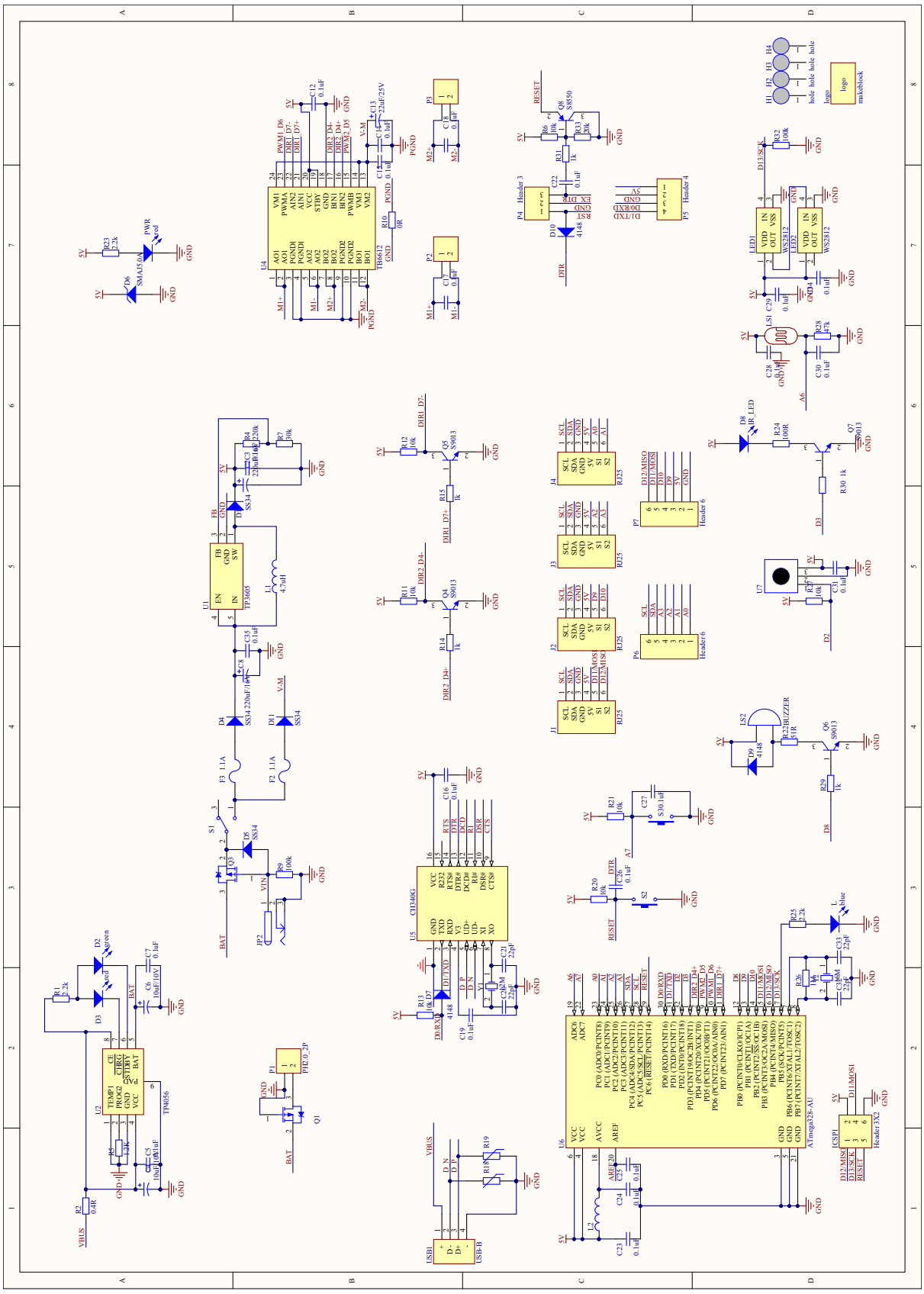
[11] PixArt: Katalogový list senzoru PixArt PMW3325, [online], [cit. 10. 2. 2020]. Dostupné z: https://www.pixart.com/_getfs.php?tb=product&id=13&fs=ck2_fs_en.

[12] PixArt: Katalogový list senzoru PixArt PMW3310, [online], [cit. 10. 2. 2020]. Dostupné z: https://www.codico.com/shop/media/datasheets/PixArt_PMW3310DH.pdf.

[13] Makeblock: mCore, [online], [cit. 11. 4. 2020]. Dostupné z: http://learn.makeblock.com/en/mcore/.

Příloha A

Schéma desky mCore



Obrázek A.1: Schéma desky mCore [13]

Příloha B

Vybrané specifikace optického senzoru PMW3325

V tabulce B.1 jsou uvedeny specifikace a v tabulce B.2 registry senzoru PMW3325, které přímo souvisí s jeho použitím v této práci a které jsou uvedeny na webu výrobce [10] nebo v katalogovém listu [11].

Pouzdro	8-PDIP
Pracovní napětí	1,8-2,1 V
Pracovní proud	6,7 mA
Rychlost sledování	100 ips
Maximální zrychlení	20 g
Maximální rozlišení	5000 cpi
Komunikační rozhraní	SPI

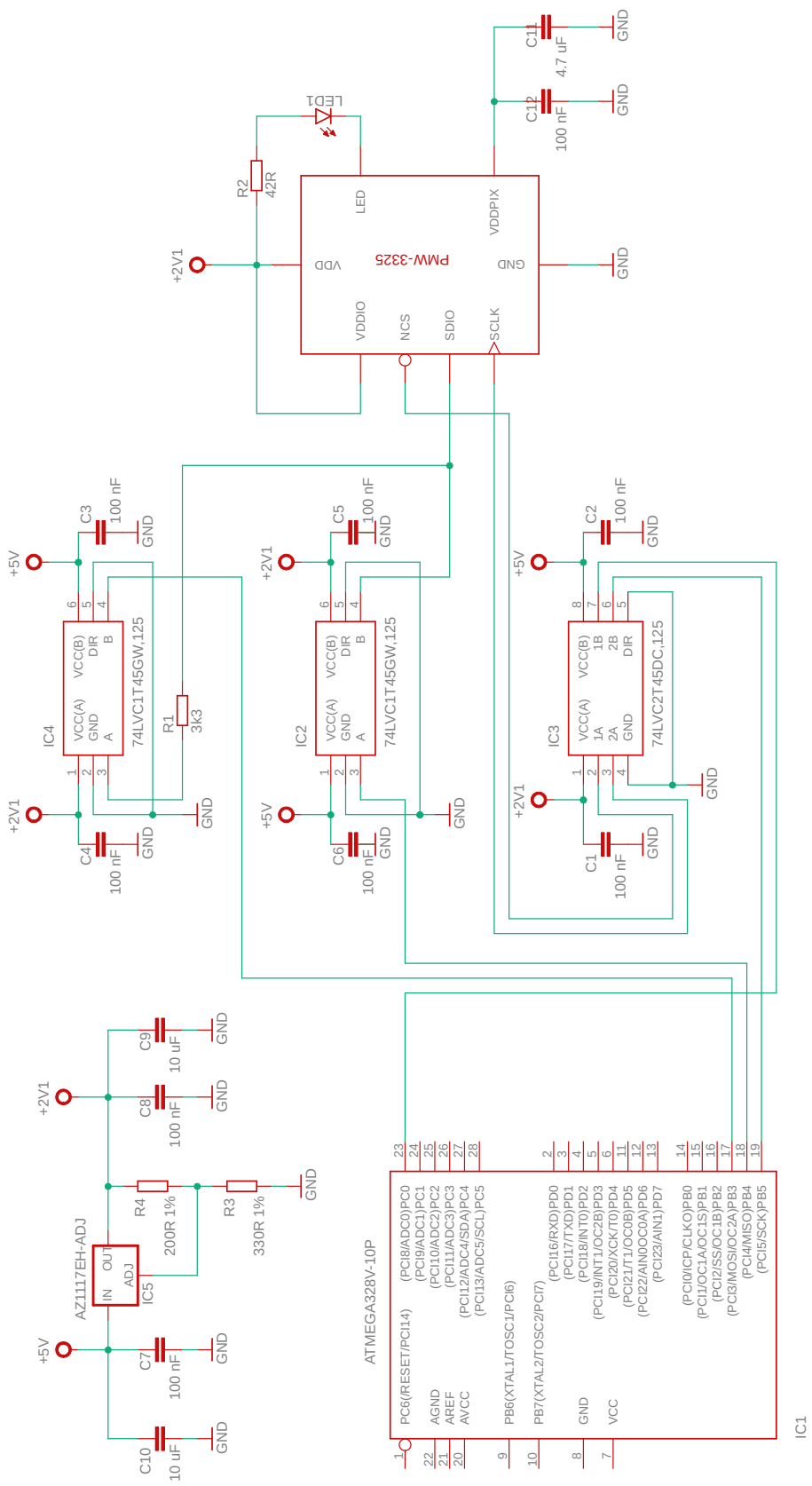
Tabulka B.1: Vybrané specifikace senzoru PMW3325

Adresa	Název registru	Přístup
0x00	PRODUCT_ID	čtení
0x16	BURST_MOTION_READ	čtení
0x1B	RESOLUTION	čtení a zápis
0x3A	POWER_UP_RESET	zápis

Tabulka B.2: Vybrané registry senzoru PMW3325

Příloha C

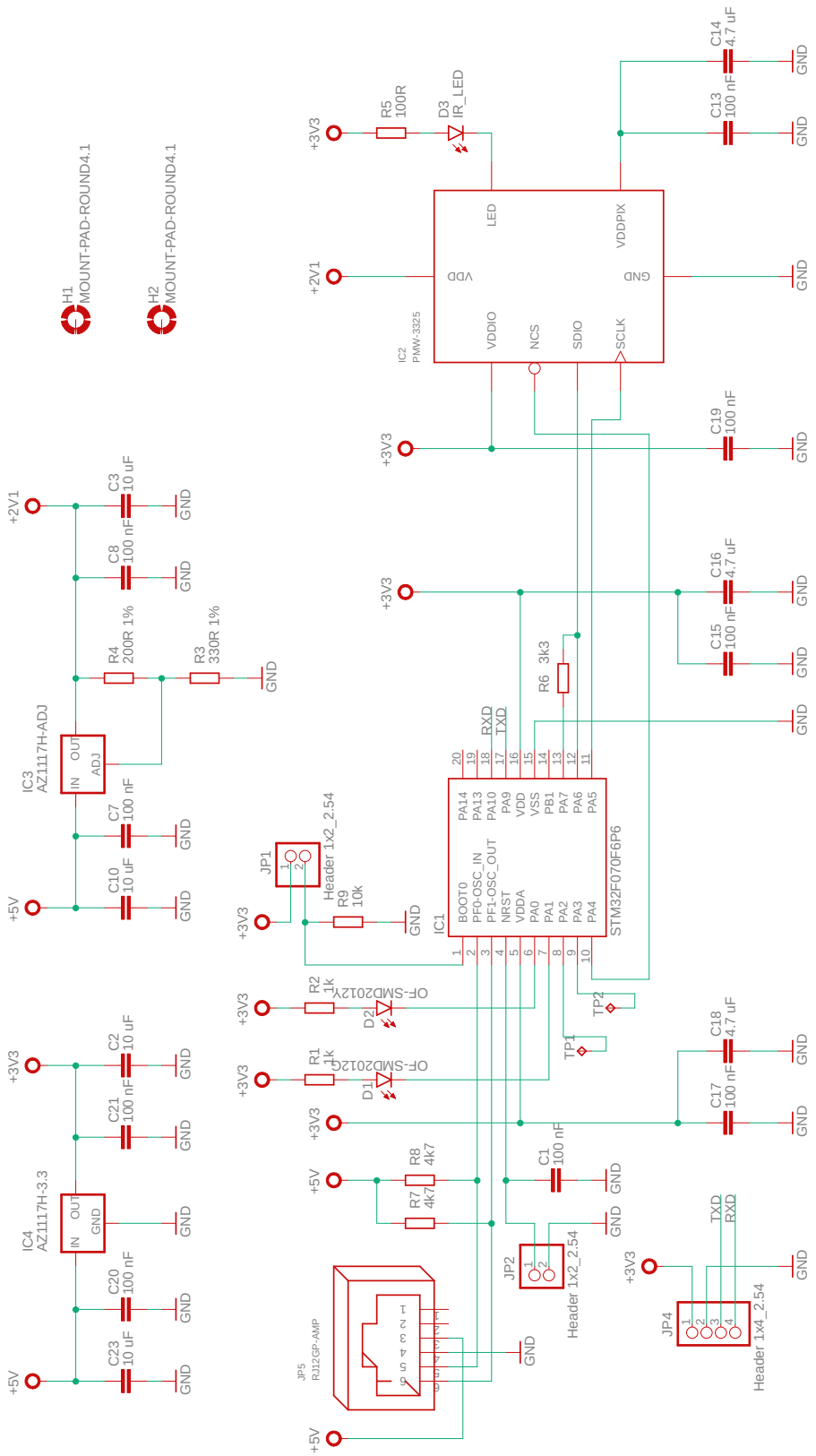
Schéma zapojení testovacího vzorku modulu



Obrázek C.1: Schéma zapojení testovacího vzorku se senzorem PMW3325

Příloha D

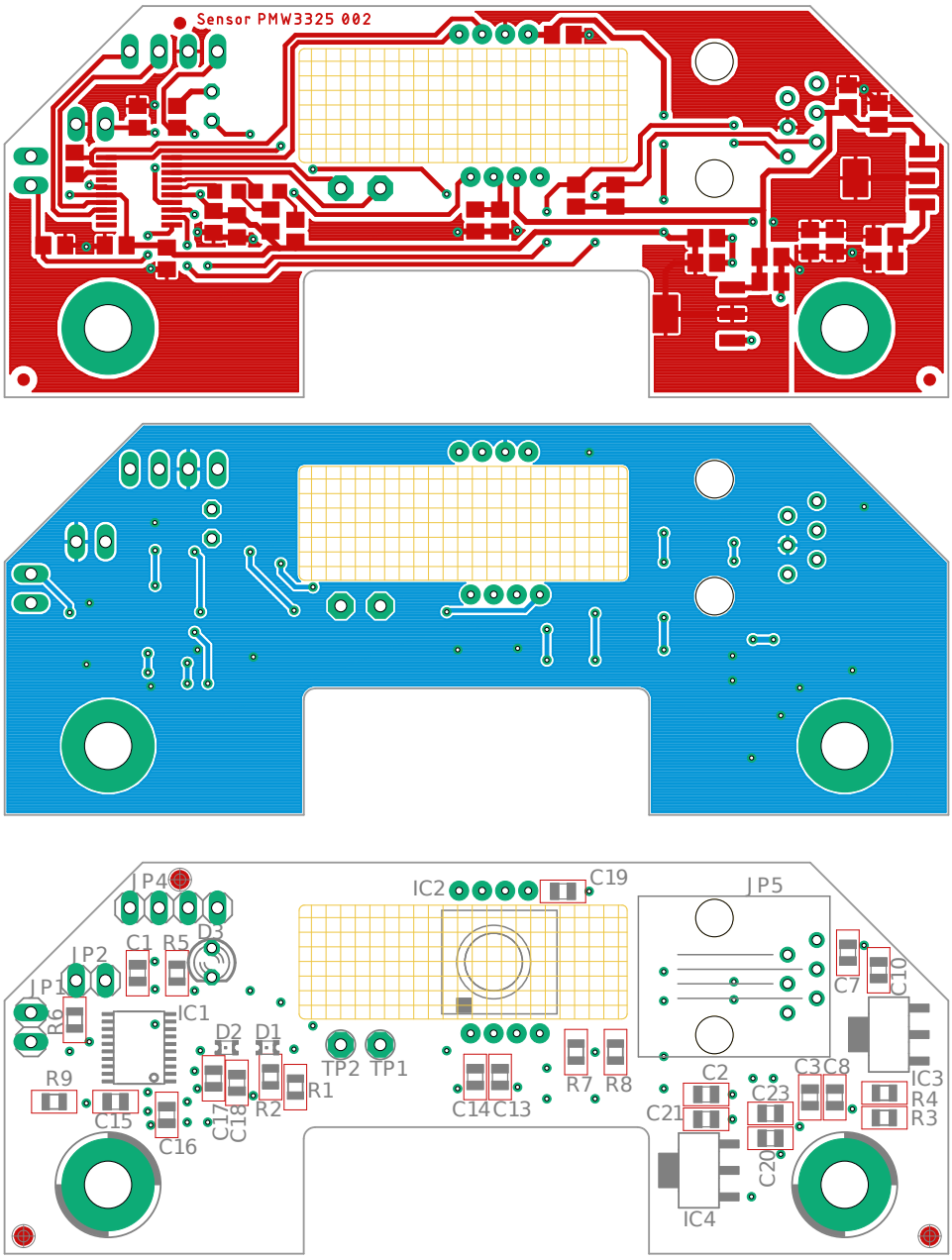
Schéma navržené desky plošných spojů



Obrázek D.1: Schéma DPS se senzorem PMW3325 a procesorem STM32F070F6P6

Příloha E

Deska plošných spojů



Obrázek E.1: Deska plošných spojů

Quantity	Value	Package	Parts	Description
1	100R	R0805	R5	
1	200R 1%	R0805	R4	
1	330R 1%	R0805	R3	
2	1k	R0805	R1, R2	
1	3k3	R0805	R6	
2	4k7	R0805	R7, R8	
1	10k	R0805	R9	
9	100n / 16V	C0805	C1, C7, C8, C13, C15, C17, C19, C20, C21	X5R or X7R
3	4.7u / 16V	C0805	C14, C16, C18	X5R or X7R
4	10u / 16V	C0805	C2, C3, C10, C23	X5R or X7R
1	STM32F070F6P6	TSSOP-20	IC1	Microcontroller
1	PMW-3325	PMW-3325	IC2	Mouse sensor
1	AZ1117H-ADJ	SOT223	IC3	800mA Low-Dropout Linear Regulator
1	AZ1117H-3.3	SOT223	IC4	800mA Low-Dropout Linear Regulator
2	Header 1x2_2.54	PINHD-1X2	JP1, JP2	PIN HEADER
1	Header 1x4_2.54	PINHD-1X4	JP4	PIN HEADER
1	RJ12GP-AMP	RJ12STRAIGHT	JP5	RJ25 (RJ12 6p6c) THT socket
1	OF-SMD2012G	LED0805	D1	Green SMD LED
1	OF-SMD2012Y	LED0805	D2	Yellow SMD LED
1	IR_LED	LED3MM	D3	Mouse sensor IR LED

Tabulka E.1: Seznam použitých součástek

Příloha F

Funkce pro komunikaci s modulem vytvořené v Arduino IDE

Během testování programu pro mikrokontrolér bylo vytvořeno několik funkcí v programu pro řídicí obvod (desku mCore) v prostředí Arduino IDE. Tyto funkce jsou uvedeny v následující tabulce:

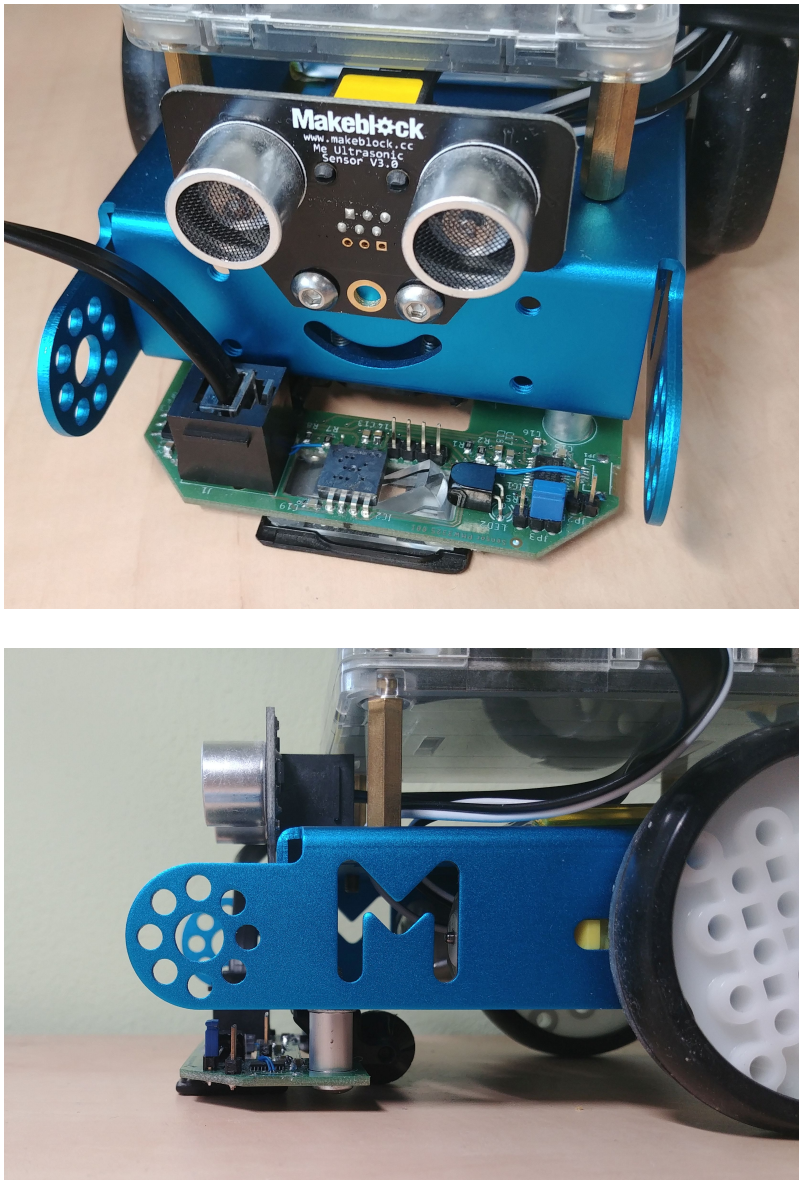
Název funkce	Argumenty	Význam
sensor_wr_control	byte value	hodnota proměnné control
sensor_wr_position	short pos_x short pos_y short pos_phi_deg	pozice robotu x_r [mm] souřadnice robotu y_r [mm] souřadnice robotu φ_r [°]
sensor_wr_target	short x short y short tolerance	souřadnice cílového bodu x_c souřadnice cílového bodu y_c tolerance vzdálenosti
sensor_wr_reg_const	byte cmd int_32 value	příkaz pro regulační konstantu hodnota regulační konstanty
sensor_wr_res_x	short res_x	hodnota rozlišení v x
sensor_wr_res_y	short res_y	hodnota rozlišení v y
sensor_wr_phi_0	short phi_0	hodnota parametru φ_0

Tabulka F.1: Funkce pro zápis hodnot

Pro čtení hodnot proměnných slouží funkce sensor_read_command(byte cmd), která vyčte hodnoty odpovídajících proměnných podle hodnoty předaného příkazu cmd a uloží je do příslušných globálních proměnných.

Příloha G

Fotografie robotu s výsledným modulem



Obrázek G.1: Fotografie robotu