VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

UPGRADE ROBOTA TRILOBOT

TRILOBOT UPGRADE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK POLÁŠEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Polášek Patrik**
Program: Informační technologie
Název: **Upgrade robota Trilobot**
Trilobot Upgrade
Kategorie: Vestavěné systémy

Zadání:

1. Seznamte se s šasi robota Trilobot. Nastudujte Robotický operační systém (ROS), práci s Arduinem a senzory používané v robotice (enkodéry, IMU, sonary, IR senzory, Kinect).
2. Navrhněte změny na šasi robota Trilobot tak, aby se na něj daly připevnit požadované senzory. Navrhněte schéma elektroniky nutné pro připojení senzorů a včetně napájení z baterie a nebo ze zdroje. Pro čtení dat ze senzorů použijte Arduino a naměřená data posílejte do ROS.
3. Navržené úpravy a potřebný kód implementujte. Pro vytvoření úchytů senzorů použijte 3D tiskárnu.
4. Otestujte funkčnost senzorů a pohyb robota.

Literatura:

- McComb, G, Predko, M.: Robot Builder's Bonanza. McGraw-Hill, 2006, ISBN 0-07-146893-5
- Cook, D.: Intermediate Robot Building. Apress, 2004, ISBN 1-59059-373-1

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 31. října 2019

Abstrakt

Cílem této práce je vytvoření robota pohybujícího se na kolech, propojení komunikace mezi senzory a mikropočítači s pomocí Robotického operačního systému (ROS). Senzory jsou připevněny na plastových uchýtech vytisklých na 3D tiskárně. Mikropočítač Arduino obstarává nízkoúrovňové signály pro čtení dat ze senzorů a signály k ovládání motoru, druhý a výkonnější mikropočítač ODROID-XU4 na němž běží jádro ROS a grafická aplikace umožňující ovládat robota na dotykovém displeji.

Abstract

The main aim of this work is to create a robot moving on wheels, create communication among all the sensors and microcomputers with the help of the Robot Operating System (ROS). Sensors are mounted on a plastic handle that is printed on 3D printer. The Arduino microcomputer manages low-level signals for reading sensor data and signals to control the engine, the another one and more powerful ODROID-XU4 microcomputer runs the core of ROS and a graphical application that allows controlling robot on the touchscreen.

Klíčová slova

Trilobot, ROS, Arduino, IR senzor, ultrazvukový senzor, robot, I2C, IMU, MiniIMU-9 v3, SRF08, Sabertooth 2X5, stejnosměrný motor, odometry, enkodéry, napájecí zdroj, A/D převodníky

Keywords

Trilobot, ROS, Arduino, IR sensor, ultrasonic sensor, robot, I2C, IMU, MiniIMU-9 v3, SRF08, Sabertooth 2X5, DC motor, odometers, encoders, power supply, A/D convertors

Citace

POLÁŠEK, Patrik. *Upgrade robota Trilobot*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Upgrade robota Trilobot

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Patrik Polášek
28. května 2020

Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce, panu Ing. Jaroslavu Rozmanovi Ph.D., za poskytování rad, součástek, které jsem využil při tvorbě robota a velmi rychlé odezvy na stav bakalářské práce.

Také bych chtěl poděkovat rodině, za výpomoc při upravování konstrukce robota a poskytnutí nástrojů, které byly v době nouzového stavu v ČR hůře sehnatelné.

Obsah

Kapitola 1

Úvod

Tato práce popisuje vytvoření robota. Robot je obecně velmi široký pojem, pod kterým si každý představí trochu něco jiného. Někdo si vybaví kuchyňského robota nebo robotický vysavač – to jsou roboti, kteří svou činností usnadňují každodenní život jejich majitelům, jiným leží na mysli komplexní robot se schopnostmi blížícími se těm lidským, tzv. humanoid. Rozdíly mezi nimi mohou být znatelné a vytvořit nějakou přesnou definici slova robot je obtížné až nemožné.

Využití robotů je široké a odvíjí se od schopností robota. Může se jednat o drobné domácí pomocníky, roboty v průmyslu, průzkumné roboty až po specializované roboty pro vojenské účely. Příkladem průzkumného robota je robot schopný létat, tzv. dron, který se může dostat do míst člověku běžně nedostupných. Dalším příkladem může být nasazení robota v podmínkách lidem nebezpečných, například radiace, příliš vysoké nebo nízké teploty. Příkladem vojenského robota je robot průzkumný, jehož úkolem je prozkoumat trasu a odhalit nástrahy, které mohou ohrozit lidský život.

Robot Trilobot, o kterém je tahle práce, se pohybuje na kolech. Jde tedy o pozemního robota. Jeho cílem je přemísťovat se na základě pokynů od uživatele a informovat o svém okolí prostřednictvím senzorů.

Kapitola 2 pojednává o hardwarových součástech robota. Je zde popsána konstrukce robota Trilobot, rozměry robota, senzory a akční členy, enkodéry, napájení a princip analogově digitálního převodníku. Aby spolu mohli všechny jednotlivé části komunikovat, musejí být propojeny nejčastěji sběrnicí a v této kapitole je popsán princip připojování zařízení na sběrnici I2C.

V pořadí 3. kapitola je o softwarových prostředcích. Použité mikropočítače se liší výkonem a od výkonu se odvíjí jejich účel a schopnosti. V této kapitole jsou popsány programové prostředky, vývojová prostředí a knihovny použité k naprogramování robota.

Kapitola číslo 4 je o implementaci. Zde jsou popsány konkrétní použité součástky, jejich připojení, použité programové konstrukce. Jde o praktickou aplikaci poznatků z kapitol 2 a 3.

Kapitola 5 zahrnuje návrhy na další rozšíření robota Trilobot. Tato práce se snaží poskytnout fungující základ pro pohybujícího se robota, jehož kód půjde snadno upravovat a rozšiřovat o další funkcionalitu navíc.

Kapitolou 6 je stručně shrnut celý projekt, cíl práce a překážky, které ovlivnily buď samotnou práci nebo její průběh.

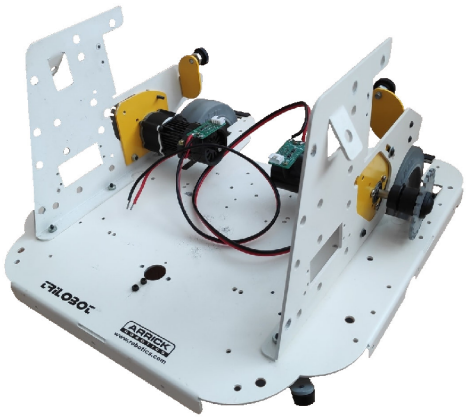
Kapitola 2

Hardwarové součásti robota

Nedílnou součástí robota tvoří hardwarové členy. Dříve byl hardware úzce propojen se softwarem a u většiny případů se jednalo o tzv. řešení na míru. Tento trend je již delší dobu na ústupu, ale občas se stále objeví neobvyklá řešení vyžadující specifický přístup. Následující podkapitoly popisují konkrétní hardwarové prvky použité pro řízení robota, jejich propojení, účel, ke kterému jsou použity a případné nejasnosti, komplikace a jejich řešení. Snaha je použít generická řešení, která nevyžadují speciální ovladače k přístupu k hardwaru. Důležité je najít vhodný poměr mezi výkonem, spotřebou a cenou, zatímco si jsou tyto složky navzájem protichůdné.

2.1 Konstrukce robota

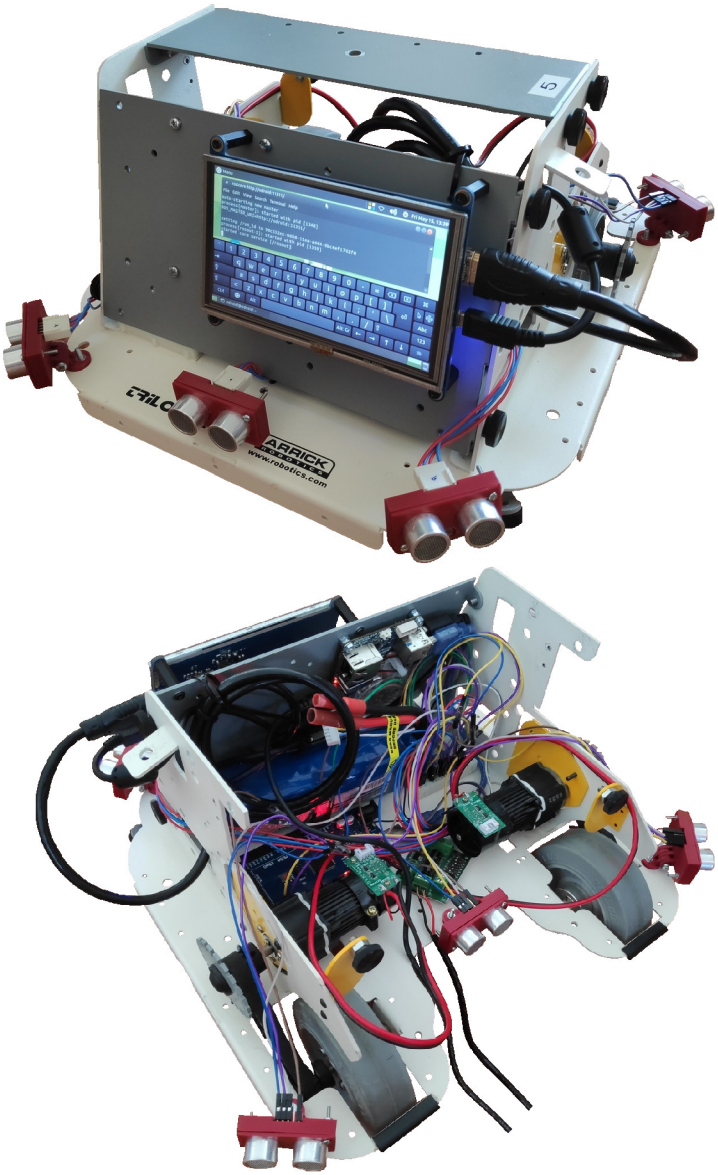
Robot Trilobot je tvořen bytelnou železnou konstrukcí. Železná základna obsahuje spoustu vyvrtaných otvorů, které mohou posloužit pro připevnění dalších zařízení, případně k rozšíření základny dalšími díly. Ke konstrukci jsou připevněny dva nové stejnosměrné motory, jeden pro každé zadní kolo. Zadní kola jsou využita k pohonu, v přední části robota se nachází ještě pomocné kolečko. Lépe než popis vystihne tělo robota obrázek 2.1.



Obrázek 2.1: Původní konstrukce robota Trilobot, pouze s osazenými novými motory. Rozměry těla jsou šířka 28,5 cm, výška 20,5 cm, délka 30,5 cm.

Připevnění všech použitých senzorů a obrazovky vyžadovalo provést dodatečné úpravy v podobě vyvrtání několika děr k připevnění obrazovky a mikropočítačů. Ultrazvukové senzory jsou připevněny plastovými úchyty vytisknutými na 3D tiskárně. Návrh úchytů pro senzory a pomocné kolečko ve formátu STL pro 3D tisk mi poskytl vedoucí bakalářské práce Ing. Jaroslav Rozman, Ph.D.

Finální podoba robota po provedení všech změn je zachycena na obrázku 2.2.



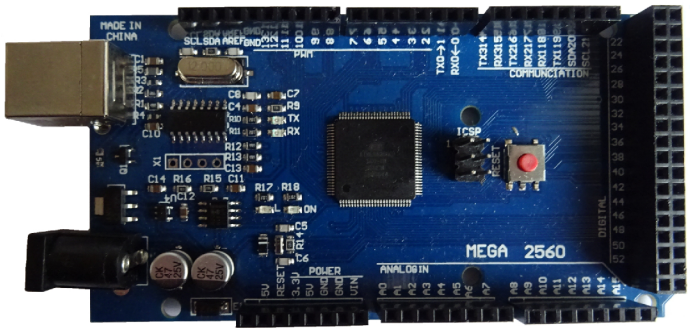
Obrázek 2.2: Výsledná podoba robota po provedení všech změn a úprav.

2.2 Arduino Mega 2560

Jde o často využívanou a populární vývojovou desku zobrazenou na obrázku 2.3. Jak návrh hardwaru, tak i software je dostupný jako open source, což umožňuje provádět úpravy a rozšíření pro specifické účely. Svobodu úprav využila řada firem a vyrábí různé klony.

Klony jsou zpravidla plně kompatibilní s originálem Arduina, k jejich programování se používají stejné nástroje. Klon může zaujmout dostupnější cenou nebo funkcionalitou navíc v podobě většího množství pinů. Nestandardní rozšíření se pak nejčastěji řeší voláním funkcí z externích knihoven.

Jádrem Arduina Mega 2560 je mikropočítač Atmel ATmega2560 taktován na 16 MHz. Dostupných je 16 vstupních analogových pinů a 54 vstupně výstupních digitálních pinů. Operační napětí je 3,3 V nebo 5 V. Rozhraní USB typu B poskytuje komunikaci pro nahrání programů z počítače, sloužit může i jako sériová komunikace či k zajištění napájení.



Obrázek 2.3: Klon mikrokontroléru Arduino Mega 2560.

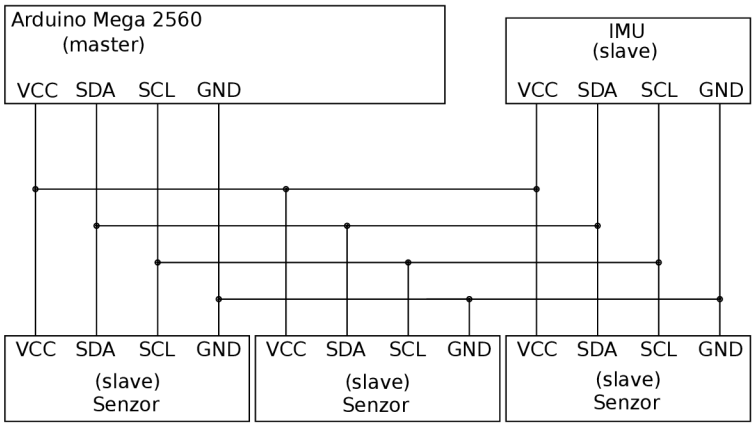
Na vývojové desce Arduino není provozován žádný vyšší operační systém. Účelem tohoto mikropočítače je řízení a ovládání nízkoúrovňových signálů. Složitější zpracování, analýzu a uchování dat je potřeba provádět na výkonnějším mikropočítači. Dostupná paměť SRAM (Static Random Access Memory) je 8 KB, dostupná paměť pro program činí 256 KB, z čehož 8 KB zabírá bootloader. Časté a jednoduché konstrukce realizované pomocí funkcí vedou na paměťově náročnější aplikace. Další technické specifikace jsou uvedeny na webových stránkách[1] Arduina.

2.2.1 I2C sběrnice

Arduino podporuje komunikaci přes sběrnici I2C, někdy označovanou též jako IIC nebo I^2C. Sběrnice využívá piny SDA (Serial Data, pin 20) a SCL (Serial Clock, pin 21). Na obrázku 2.4 lze vidět princip zapojení zařízení ke sběrnici. Pro samotnou komunikaci jsou vyžadovány pouze dva vodiče, SDA a SCL, zde jsou navíc vodiče VCC a GND, které slouží pro napájení zařízení připojených na sběrnici.

Na sběrnici vždy komunikuje jedno zařízení typu Master. V tomto případě je to Arduino Mega 2560. Dále je ke sběrnici I2C připojeno jedno nebo více zařízení typu Slave, konkrétně jde o ultrazvukové senzory popsané v kapitole 2.4 a součástku IMU (Inertial Measurement Unit – inerciální měřicí jednotka) typicky obsahující gyroskop, akcelometr a magnetometr. Více informací o IMU se lze dozvědět v sekci 2.6. Frekvenci komunikace vždy určuje člen Master. Běžně používaná rychlost komunikace je 9600 baudů, což odpovída přenesení 9600 bitů za sekundu. Master zahajuje komunikaci vysláním 8 bitů po SDA vodiči. Sedm bitů určuje adresu zařízení (možnost připojit až 128 zařízení) a osmý bit značí, zda se budou data číst nebo zapisovat. Následně probíhá samotný přenos dat.

[1] https://store.arduino.cc/arduino-mega-2560-rev3



Obrázek 2.4: Schéma zobrazující princip propojení senzorů s Arduinem Mega 2560 na sběrnici I2C.

2.3 ODROID-XU4

Informace pochází z uživatelského manuálu [7]. Jednodeskový počítač ODROID-XU4 je další výpočetní jednotka použita pro řízení robota. Základem je osmijádrový ARM (Advanced RISC Machine) procesor Exynos, dostupné jsou 2 GB paměti RAM a velikostně je mikropočítač podobný jako Arduino Mega 2560. Operační systém je potřeba dodat na microSD kartě nebo na paměťovém modulu eMMC. Na desce je přítomný mechanický přepínač mezi microSD a eMMC paměťmi a poměrně snadno se dá nedopatřením přepnout. Další konektivitu tvoří dva USB konektory verze 3.0, jeden USB konektor ve verzi 2.0, HDMI konektor pro připojení obrazovky a konektor RJ-45 pro připojení k internetové síti. Wi-fi modul na desce není přítomný, ale lze využít externí USB Wi-fi adaptér.

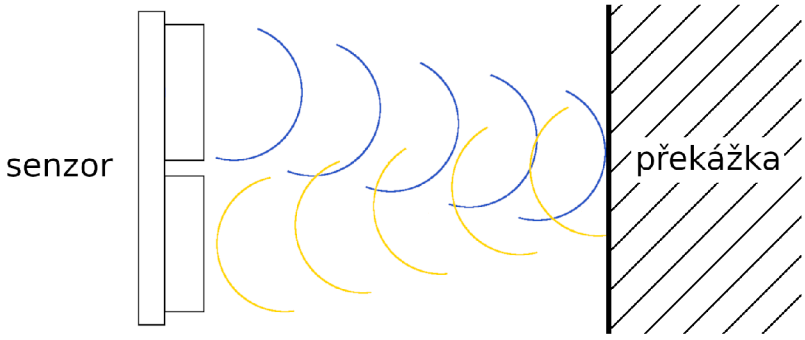
Připojení obrazovky přes HDMI podporuje automatickou detekci rozlišení, která funguje dobře. Nicméně vynutit jde i manuální nastavení rozlišení. Nastavení je závislé na použitém operačním systému na ODROID-XU4. Na Ubuntu Mate 18.04 se nastavení rozlišení nachází na microSD kartě na oddílu boot v souboru boot.ini. Všechny možnosti zakomentované znamenají automatickou detekci rozlišení. Odkomentovat je nutné vždy jen jeden řádek s volbou rozlišení! Při odkomentování více řádků se bere v potaz první odkomentovaná možnost a ostatní se ignorují.

Dvě stavové diody informují o stavu zařízení. Trvale rozsvícená červená dioda značí dostatečné napájení. Trvale svítící modrá dioda značí fázi bootování operačního systému (první spuštění může trvat i 5 minut), při úspěšném běhu operačního systému dioda modře bliká. Operační systém doporučuji při skončení práce vypínat, neočekáváné ukončení může poškodit spouštěcí oddíl na microSD kartě, další spuštění se zasekne ve fázi bootování a je potřeba operační systém nově nahrát na microSD kartu.

2.4 Ultrazvukový senzor pro měření vzdálenosti

Ultrazvukový senzor vysílá ke změření vzdálenosti ultrazvukové vlny a detekuje jejich ozvěnu odraženou od překážky. Dráha se pak spočítá jako rychlost zvuku vynásobená časem, čas se musí vydělit dvěmi, protože zvuk urazí dráhu tam i zpět. Senzor a šíření ultrazvuku je znázorněno na obrázku 2.5. Ultrazvukové senzory (sonary) je možno nasadit v různých prostředích, která dělají problém infračerveným senzorům. Mezi problémová pro-

středí, která ale ultrazvukový senzor zvládá, patří například skleněné plochy, které světlo propouští, zrcadlové plochy nebo hladina vody, která mění lom světla a část ho pohlcuje. [4]

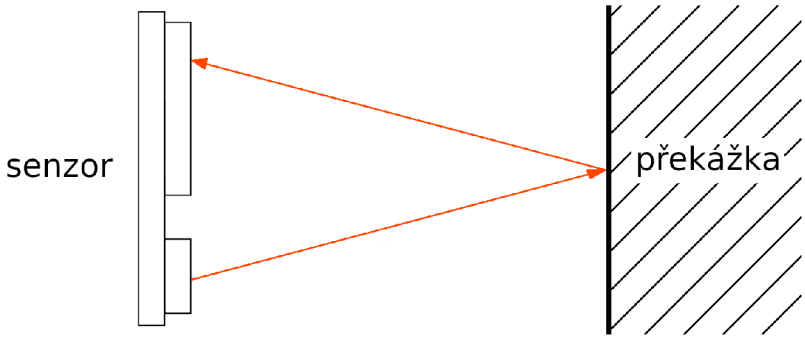


Obrázek 2.5: Princip měření vzdálenosti ultrazvukovým senzorem. Vysílač mění přivedenou elektrickou energii na mechanické vlnění. Na senzor působí odražené vlny, a ten mění mechanickou energii zpět na elektrickou.

Použité ultrazvukové senzory jsou senzory SRF08. Senzor je vyráběn pro připojení na I2C sběrnici. Mezi klíčové vlastnosti patří dosah až 11 metrů a snímaná kruhová výseč s úhlem 55°. Úhel bývá v praxi o něco užší. [2]

2.5 Infračervený senzor pro měření vzdálenosti

Často používaná a zavedená zkratka pro označení infračerveného senzoru je IR (infrared) senzor. IR senzor se skládá ze dvou hlavních částí – zdroje světla a senzoru. Ze zdroje světla je vysláno infračervené světlo, které se od překážky odrazí a dopadne na senzor. Dráha se dá vypočítat jako rychlost světla vynásobená dobou letu paprsku. Doba letu se musí podělit dvěmi, protože paprsek cestuje tam i zpět a urazí dráhu dvakrát. Šíření a detekce světla je znázorněna na obrázku 2.6. Hlavní rozdíly oproti ultrazvukovým senzorům jsou dány fyzikálními vlastnostmi světla a jeho šířením. Přesnější výsledky jsou naměřeny ve tmě, přímé osvětlení sluncem výsledky zkresluje. Význam má také povrch, od kterého se světlo odráží. Například sklo světlo propouští, voda část světla pohlcuje a mění lom světla, temný povrch světlo pohlcuje. [4]



Obrázek 2.6: Vysílač i přijímač infračerveného světla má podobu diody. Diody mají kolem sebe vystouplé okraje z černého plastu, aby nedocházelo k detekci nežádoucích odrazů.

[2] https://www.robot-electronics.co.uk/htm/sonar_faq.htm

Sharp GP2Y0A41SK0F je infračervený senzor, který poskytuje analogová data. Rozsah změřitelné vzdálenosti je 4 – 30 cm.

2.6 Inerciální měřicí jednotka

Inerciální měřicí jednotka je velmi známá pod často používanou zkratkou IMU (Intertial Measurement Unit). IMU je zařízení, které používá měřicí systémy jako gyroskopy, akcelometry a magnetometry k určování relativní polohy, rychlosti a zrychlení. IMU je citlivé zařízení na okolní rušení. Vlivem okolních vibrací nebo rušivých signálů dochází ke snadnému vzniku chyb měření. [8]

Na robotu Trilobot jsou použita IMU zařízení L3GD20 a MiniIMU-9 v3 vyráběná firmou Pololu. Zařízení se připojují na sběrnici I2C. MiniIMU-9 v3 je kombinace zařízení L3GD20H a LSM303D. L3GD20 je gyroskop a měří pouze rotaci v osách x, y a z, ale neměří náklon, tzn. když se gyroskop ustálý v nějaké pozici, nelze zjistit jeho aktuální náklon, protože zrychlení rotace je nulové. Verze s H se liší pouze menší spotřebou (6 mA místo 7 mA) a trochu menšími rozměry. Obě verze si jsou velmi podobné a spolehlivě je lze rozlišit podle uspořádání součástek na desce plošných spojů.

MiniIMU obsahuje jak gyroskop, tak akcelometr, který měří zrychlení a magnetometr pro určení aktuální orientace robota. Orientace se vztahuje vzhledem k severu. Magnetometr tak zastává funkci digitálního kompasu. Určuje pozici natočení ve stupních v rozsahu 0 až 360 stupňů – sever je 0°, východ 90°, jih 180°, západ 270°.

Součástky IMU je potřeba pevně připevnit ke konstrukci robota. Pohyb a otáčení nepřipevněného akcelometru, gyroskopu nebo magnetometru v průběhu jízdy robota by mohl vyvolat nežádoucí účinky jako je například nepravdivá informace o zrychlení nebo o aktuální orientaci. Chybná data použitá k navigování by vedla k opačnému výsledku než jaký byl záměr. Negativní vliv mají také vibrace od motoru, které zkreslují výsledky získané z akcelometru. Akcelometr je citlivá součástka sama o sobě, lepší přesnosti lze dosáhnout přečtením více hodnot a provedením jejich průměru. Případně se může z výpočtu vyřadit jedna hodnota, která se výrazně liší od ostatních.

2.7 Řídicí jednotka motoru

Řídicí jednotka motoru, Sabertooth 2X5, musí obstarávat dvě hlavní funkce, aby byl robot schopen pohybu.

První důležitou funkcí je řízení množství energie dodávané ze zdroje do motorů. Napětí ze zdroje musí být regulováno, aby nedošlo ke zničení samotných motorů. Polovodičové součástky a rezistory mají určité maximální mezní hodnoty napětí a při jejich překročení dojde k průrazu. Průraz polovodičového přechodu tranzistoru nebo izolační vrstvy rezistoru vede ke zničení součástky, součástka tak přijde o svoje vlastnosti a stane se trvale vodivou. Regulování napětí také umožňuje použít širší spektrum zdrojů, které nemusí mít přesnou konstantní hodnotu napětí.

Druhou a neméně důležitou funkcí je správa dodávání energie k motorům na základě řídicích signálů. Jako řídicí signál dovoluje Sabertooth 2X5 použít spojitý signál nebo digitální hodnoty přenášeny sériovým přenosem.

Jelikož řídící jednotka disponuje více režimy[3], je mezi nimi nutno rozlišovat a nastavit patřičné parametry. K tomu slouží šest mechanických přepínačů. Dostupné režimy jsou

[3] https://www.dimensionengineering.com/datasheets/Sabertooth2x5.pdf

analogový vstup, R/C vstup pro ovládání rádiovým vysílačem a přijímačem, zjednodušený sériový přenos, sériový přenos s využitím paketů.

Zvolený zjednodušený sériový přenos obsahuje následující možnosti. Nastavení přepínače do pozice dolů znamená posunout ho blíže k popiskům s číslem přepínače 1 – 6.

Přepínač 1 je v horní pozici.

Přepínač 2 je v dolní pozici.

Přepínač 3 závisí na použité baterii. Pokud jde o Lithiovou baterii, přepínač je v dolní pozici, jinak je v horní pozici.

Přepínače 4 a 5 ovládají rychlost sériového přenosu. Přepínač 4 v dolní poloze a přepínač 5 v horní poloze nastaví rychlost 9600 baudů.

Přepínač 6 dovoluje řídit více Sabertooth jednotek jedním sériovým kanálem. Dolní pozice značí použití módu Slave Select, signálem na S2 se vybírá jednotka, které je sériový signál na S1 určen. Přepínač v horní pozici značí použití pouze jedné Sabertooth jednotky, příkazy jsou posílány na vstup S1.

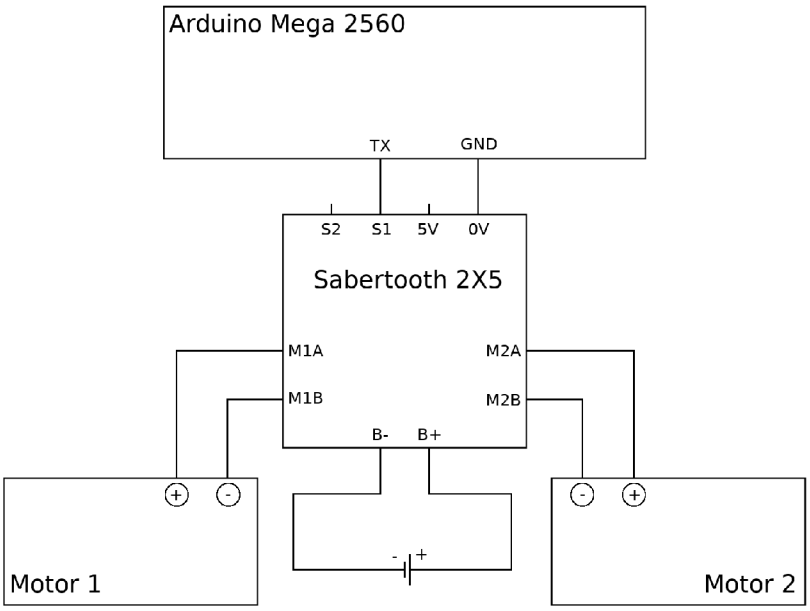
Schéma zapojení řídicí jednotky motoru je na obrázku 2.7. Vodiče jsou zde uchyceny svorkou na šroubek. Přívody S1 a S2 slouží pro řídicí signály. Oba jsou použity například při řízení spojitým signálem, kdy S1 ovládá jeden motor a S2 druhý motor. Režim zjednodušený sériový přenos používá k ovládání motorů jeden bajt, a ten je přenášen pouze přes přívod S1. S2 zůstává nezapojený. Přívod označený 0V představuje nulovou hodnotu signálu vůči řídicím signálům přiváděným na S1, případně S2, tedy uzemnění. Na přívod 5 V generuje řídicí jednotka hodnotu napětí 5 V a proud 100 mA, které můžou posloužit jako napájení rádiového přijímače, který zachytává řídicí signály pro S1 a S2. Přívody B+ a B− slouží pro přívod napájení motorů. Pomocí přívodů M1A (+) a M1B (−) se připojuje jeden motor. Obdobně pro přívody M2A (+) a M2B (−) s druhým motorem.

2.8 Stejnosměrný motor

Informace o stejnosměrných motorech pochází z [4]. Motor představuje část robota, která mění elektrickou energii na mechanickou. Typicky jde o rozpohybování rotační části, hřídele, na které je umístěno kolo. V praxi se velmi často objevují mechanické převodníky energie jako jsou ozubená kola nebo řemeny. Menší ozubené kolo je umístěno na hřídeli od motoru a větší ozubené kolo bývá umístěno na hřídeli kola robota. K vykonání jedné otočky většího ozubeného kola je zapotřebí více otoček menšího ozubeného kola. Dalším způsobem je propojení hřídele motoru s hřídelí kola řemenem. Tento způsob se nachází i u robota Trilobot. A opět je možné použít jiné průměry hřídelí nebo pomocných kol na řemeny a měnit tak rychlost otáčení kola.

Snad nejběžnějším typem stejnosměrných motorů jsou motory využívající dva a více opačných pólů magnetů. Konstrukci stejnosměrného motoru s nejdůležitějšími částmi lze vidět na obrázku 2.8.

Mezi magnety je umístěna otočná část nazývaná rotor. Na konci rotoru je díl nazývaný komutátor, jehož úkolem je střídavě přivádět proud na vinutí v rotoru. Průchod proudu rotorem vyvolává magnetické účinky a způsobuje rotaci rotoru mezi magnety. Proud ze zdroje se přenáší na části komutátoru přes tzv. kartáče nebo štětinky. Otáčením motoru se kartáče opotřebovávají, až dojde k přerušení kontaktů mezi kartáči a komutátorem,



Obrázek 2.7: Schéma znázorňující propojení Arduina, řídicí jednotky Sabertooth 2X5 a motorů.

nedochází k průchodu proudu a motor se tak stane nepoužitelný. Hlavní výhodou většiny stejnosměrných motorů je snadná změna směru. Změny směru lze dosáhnout průchodem proudu v opačném směru, tj. stačí zaměnit kladný a záporný pól na zdroji [4].

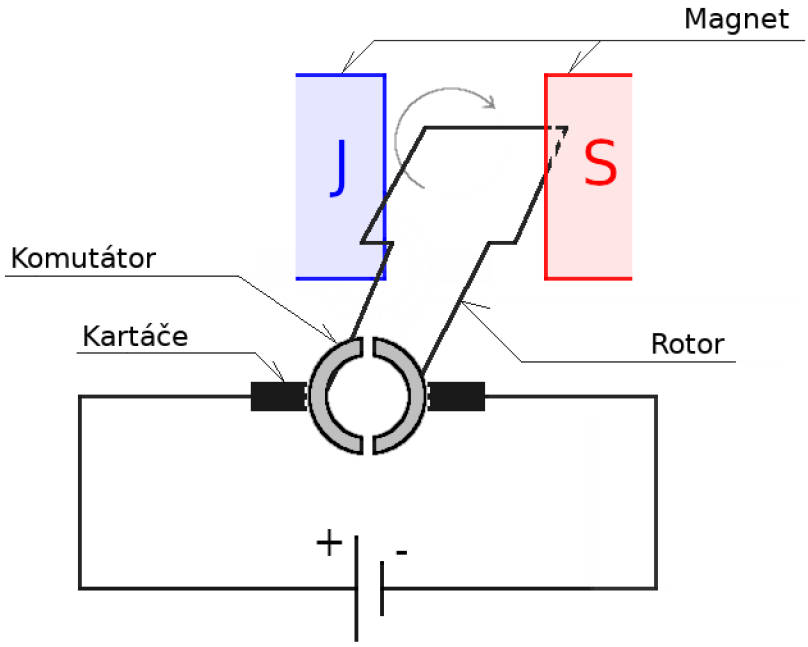
2.9 Obrazovka Adafruit

Obrazovka slouží pro interakci uživatele s robotem. Obsahuje 4 úchyty pro montáž se šroubky. Výrobce na webových stránkách[4] uvádí velikost displeje 5 palců s rozlišením 800×480 pixelů. Dostupný je konektor HDMI, který slouží pro přenos obrazu a konektor microUSB k napájení a k přenosu řídicích signálů.

Obrazovka je dotyková a využívá rezistivní technologii. Ta spočívá ve využívání tlaku, kdy se dvě vodivé vrstvy přiblíží a v místě dotyku se sníží odpor, místo se stane vodivé a dojde tak k detekci dotyku. Horní vrsta je z průžného materiálu a spodní vrstva je vodivá matice. Dotyk způsobí zkrat na určitých bodech v matici. Výhodou je možnost ovládání v rukavicích nebo s použitím jiných předmětů, například stylusu. [2]

Naopak nevýhodou je horší detekce dotyků na okrajích displeje, zde je potřeba vyvinou trochu větší tlak než někde blíže ke středu obrazovky. Přítomné jsou i tři stavové diody, které mohou pomoci při řešení problémů s obrazovkou. Trvale svítící zelená dioda informuje o dostatečném napájení. Trvale svítící modrá dioda potvrzuje HDMI přenos obrazu. Blikající červená dioda informuje o přenosu řídicích signálu, tj. fungování dotyků. Při dotyku několikrát rychle problikne.

[4] https://www.adafruit.com/product/2260



Obrázek 2.8: Zjednodušené schéma stejnosměrného motoru, který pomocí elektrické energie roztáčí hřídel. Dynamo a alternátor naopak dobíjí baterii otáčením hřídele nějakou vnější silou. Alternátor má komutátor pevně spojený s kartáči, takže proud chvíli teče jedním směrem, pak druhým. Dynamo má komutátor připevněný na rotoru, a otáčí se spolu, takže je proud vždy veden jedním směrem.

2.10 Kinect

Tato kapitola vychází z [9]. Pro fungování autonomního robota je důležitou schopností rozpoznávat okolí. V minulosti se využívaly laserová pole nebo stereo kamery. Laserová pole stála v řádu tisíců dolarů a ve výsledku vytvářela 2D snímky, stereo kamery představovaly levnější řešení a navíc vytvářely 3D mapy okolí, ale vyžadovaly velký výpočetní výkon. Obě řešení se tak příliš nerozšířila mimo laboratoře. Řešení bylo objeveno s příchodem herní konzole Xbox 360 se senzorem Kinect pro ovládání konzole pohybem těla. Kinect má dostatečný výkon k vyhodnocování 3D mapy okolí v reálném čase. Senzor se skládá z následujících částí:

- **Čtyři mikrofony** pro sledování polohy, odkud je zvuk šířen.
- **Infračervený vysíláč** – infračervené světlo je po dopadu na povrch zkresleno.
- **Hloubková kamera** analyzuje infračervené zkreslení a utváří 3D mapu objektů.
- **Naklápěcí motorek** pro změnu náklonu a sledování objektu.
- **USB kabel** k přenosu dat mezi senzorem a zařízením.
- **Barevná kamera** zachycuje obraz pro větší detail objektů.

Pro odvětví robotiky Kinect představoval velký potenciál a začaly vznikat neoficiální ovladače na Kinect. Microsoft později poskytl vývojovou sadu (SDK – Software Development

Kit) pro Kinect. Cena řešení robotů, která dříve vyšla na tisíce dolarů, se díky Kinectu snížila na několik stovek dolarů. Microsoft Kinect dosáhl konce životnosti a jeho výroba a prodej skončily. V současnosti Microsoft poskytuje na svých stránkách Kinect Azure[5], jehož SDK poskytuje na GitHub stránkách[6], podporován je i Linux Ubuntu a architektura arm64. Ovšem hardwarové nároky jsou poměrně náročné pro použití ve vestavných systémech a robotice (4 GB RAM, GPU Intel HD620). Sám Microsoft uvádí, že se nejedná o nástupce Xbox Kinectu.

Alternativy k Xbox Kinect jsou například Intel RealSense, Stereolabs ZED nebo Primesense Carmine 1.09. Přehled dostupných ovladačů pro 3D senzory je dostupný na ROS stránkách[7]. Senzory, které jsou aktuálně vyráběné, většinou vyžadují výkonné mikropočítače s podporou USB 3.0 a 4 GB RAM. Například zmíněný Stereolabs ZED navíc používá specifickou Nvidia knihovnu CUDA a vyžaduje tak mikropočítač s grafikou od Nvidia[8].

2.11 Odometry

Tato kapitola vychází z informací z [4]. Odometrie je technika výpočtu otáček kol. Informace o otáčkách umožňuje určit ujetou vzdálenost i rychlost otáčení kola, tedy rychlost robota. V praxi se velmi často používá pojem enkodéry, který má stejný význam jako odometry. K měření otáček se nejčastěji používají optické enkodéry, magnetické enkodéry nebo mechanické enkodéry.

Optické enkodéry

Základem optických enkodérů je disk – kódové kolo, které připomíná ozubené kolo. Kódové kolo je připevněné na stejnou hřídel jako kolo robota. Infračervená dioda vysílá paprsek na zuby kódového kola, které svým otáčením střídavě přerušuje paprsek. Detekovat přerušovaný signál lze dvěma způsoby, transmisivním a reflexním.

Transmisivní způsob využívá děr mezi zuby kódového kola. Infračervený paprsek má volný průchod a může dopadnou na fotodiodu, která reaguje na světlo a generuje signál. Otáčením ozubeného kola se mění pozice zubů, které zabraňují průchodu světla. Střídavě se mění stav diody a lze vyhodnocovat otáčení kola.

Reflexní způsob využívá vlastnosti odrážení a pohlcování infračerveného světla od různých povrchů. Některé zuby mohou být tvořeny bílou barvou, která lépe odráží světlo, ostatní zuby, představující „díry“, mohou být tvořeny černou barvou, která světlo pohlcuje. Otáčením kódového kola se mění intenzita odráženého IR světla a na fotodiodě se generují pulzy, kterými lze vyhodnocovat otáčení kola. Transmisivní i reflexní způsob je znázorněn na obrázku 2.9.

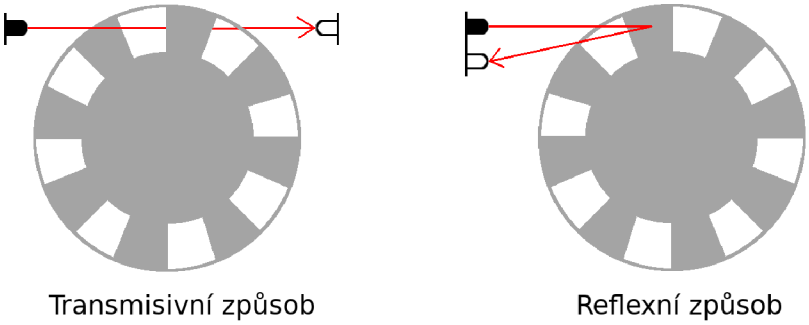
Pokud by ve skutečnosti existoval disk podobný tomu z obrázku 2.9, a pokud by obvod disku byl 16 cm, jedna změna signálu na fotodiodě by znamenala posun o 1 cm (podmínkou je velikost kola robota shodná s velikostí disku). S počtem zubů a mezer mezi nimi úzce souvisí přesnost optického enkodéru. Podobný disk s obvodem 16 cm, ale pouze se 4 zuby by byl schopen rozlišit posun o 2 cm. Kolo se také neotáčí pořád o stejně velké vzdálenosti. Může nastat situace, kdy paprsek dopadá přesně na hranu zubu. U reflexního způsobu se

[5] https://www.microsoft.com/en-us/p/azure-kinect-dk/8pp5vxmd9nhq?activetab=pivot:overviewtab

[6] https://github.com/microsoft/Azure-Kinect-Sensor-SDK

[7] http://wiki.ros.org/Sensors/3D%20Sensors

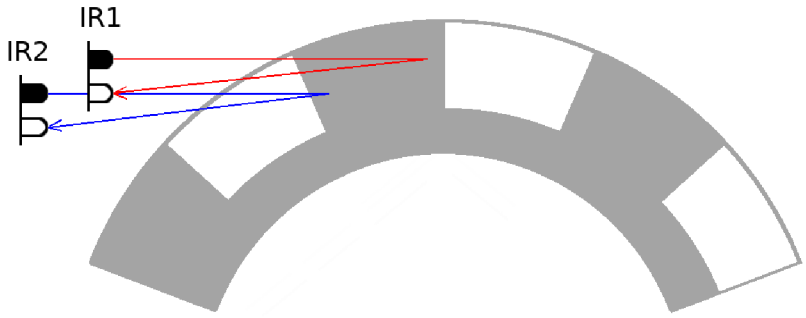
[8] https://www.stereolabs.com/docs/ros/



Obrázek 2.9: Kódový disk transmisivního enkodéru střídavě propouští a blokuje průchod infračerveného paprsku. Zuby disku reflexního enkodéru střídavě odráží nebo pohlcují infračervený paprsek.

odráží zpět na fotodiodu. Kolo provede posun např. o 1,9 cm a paprsek se stále odráží. Stav se nezměnil a není detekován žádný posun. Z toho plyne, že čím víc zubů má disk, tím lepší je jeho rozlišovací schopnost posunu. Další nepřesnosti může způsobovat například prokluzování kol na hladkém povrchu. Kódový disk se otáčí pořád stejně, ale kolo prokluzuje na místě.

Existuje i mechanismus pro zjištění směru otáčení kola. V některých situací je dobré znát, jestli se kolo otáčí dopředu nebo dozadu. Mechanismus využívá dvou infračervených diod, které jsou od sebe kousek posunuty. Princip lze vidět na obrázku 2.10.



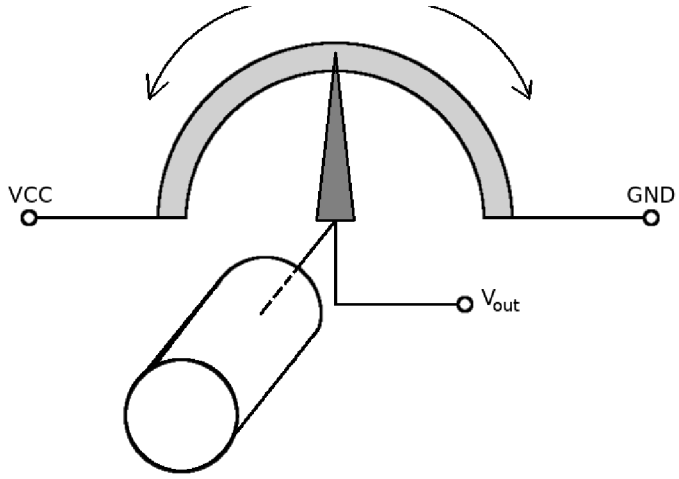
Obrázek 2.10: Určení směru otáčení optického enkodéru. Otáčením disku doprava se nejdříve přeruší modrý paprsek ze senzoru IR1 a později červený paprsek od senzoru IR2. Otáčením směrem doleva je přerušen jako první červený paprsek od senzoru IR1, následně pak modrý paprsek od senzoru IR2

Magnetické enkodéry

Magnetické enkodéry využívají Hallův jev – polovodič citlivý na magnetické pole. Ozubený kódový disk je železný. Detektor půlzu se skládá ze senzoru Hallova jevu a magnetu. Je-li zub kódového disku blízko magnetu, na polovodič začne působit magnetická síla a dostane se do propustného stavu – generuje signál. Až se zub odkloní od magnetu, magnetická síla se zmenší a polovodič se dostane do uzavřeného stavu.

Mechanické enkodéry

Mechanické enkodéry mohou být založeny na principu ozubeného kola. Otáčením ozubeného kola zuby mechanicky zaráží nějaké tlačítko – spínač – a tím se generuje signál. Velkou nevýhodou je mechanický kontakt, který se opotřebovává. Umístění mechanického enkodéru na kolo robota, které se velmi často otáči, není praktické.
Častější použití mechanických enkodéru je například v reproduktorech. V konflících k nastavení hlasitosti je ukryt otočný potenciometr. Otáčením knoflíku se mění odporová dráha potenciometru. Nevýhodou je nemožnost protáčení, takže nelze použít pro měření otáček kol. Navíc se hlasitost nastaví většinou jednou za čas a nedochází k neustálému mechanickému opotřebování. Otočný potenciometr je na obrázku 2.11.



Obrázek 2.11: Princip potenciometru. Otáčením knoflíku se posouvá otočný jezdec a mění se odporová dráha. Takto vzniklý dělič napětí mění velikost V_{out}. Pokud je jezdec vlevo, na výstupu V_{out} se objeví 5 V. S otáčením jezdce doprava se napětí na V_{out} snižuje.

2.12 Napájení robota

Motory, senzory, mikropočítače a další součásti robota vyžadují ke své funkčnosti elektrickou energii. Dodání energie se řeší nejčastěji napájecím zdrojem nebo baterií. Existuje celá řada zdrojů i baterií, které se liší svou velikostí, maximálním dodávaným výkonem a u baterií je důležitým parametrem i kapacita.

Elektromechanické zdroje

Elektromechanické zdroje využívají rotační pohyb k vytváření elektrické energie. Rotační pohyb zajišťuje hřídel. Hřídel je možno roztočit vrtulí poháněnou větrem. Ve větším měřítku se princip objevuje u větrných elektráren, kde je roztáčena velká turbína. Více praktické je použití dynama (vyrábí stejnosměrný proud) nebo alternátoru (vyrábí střídavý proud). S dynamem se lze setkat například u ruční svítilny, kde se točením s klikou vyrábí energie pro svícení. Alternátor se objevuje například v osobních automobilech, kde se jízdou a otáčením hřídelí s koly dobíjí akumulátor.

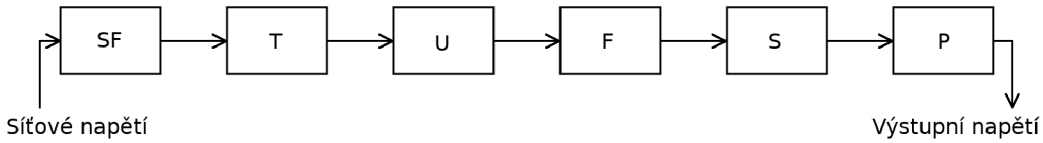
Výše popsané principy většinou nelze použít k samotnému napájení (vítr nefouká pořád, auto nejezdí jenom z kopce), ale slouží jako podpůrné zdroje k obnově části energie zpět a dobíjení baterie.

Síťové zdroje

Mezi hlavní parametry zdrojů patří maximální výkon, účinnost a stabilita zdroje. Jako síťové zdroje se využívají klasické zdroje (využívající transformátor) a spínané zdroje (spínané tranzistorem). Důležitou vlastností všech zdrojů je udržet stabilní napětí na výstupu i při zapojení různě velkých zátěží.

Klasické zdroje

Základním prvkem klasických zdrojů je transformátor s různým počtem vinutí. Transformátor mění vstupní napětí na požadované výstupní napětí. Samotný tranformátor ale nestačí a zdroj má další pomocné obvody pro vytváření použitelného signálu napětí. Blokové schéma klasického zdroje se všemi částmi je na obrázku 2.12. Účinnost klasických zdrojů je nižší než u spínaných zdrojů. Klasické zdroje bývají někdy označovány jako lineární zdroje.



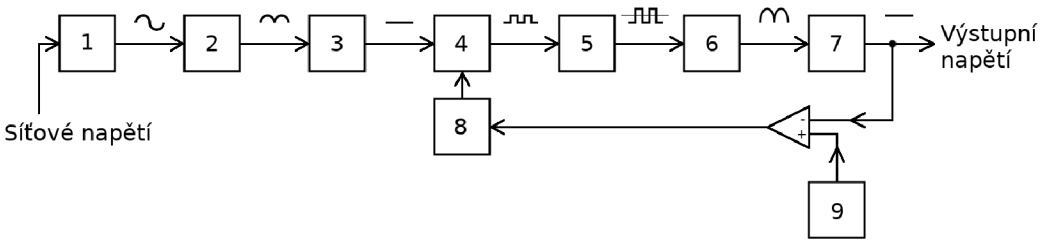
Obrázek 2.12: Blokové schéma klasického zdroje. Prvky jsou postupně zleva síťový filtr, transformátor, usměrňovač, filtr, stabilizátor, pojistka.

- Síťový filtr (SF) omezuje rušení, které se může nacházet v síti. Také může zabraňovat vzniklému rušení ve zdroji šířit se dál do sítě.
- Transformátor (T) provádí převod napětí.
- Usměrňovač (U) je prvek měnící střídavý průběh napětí na stejnosměrný průběh.
- Filtr (F) vyhlazuje průběh usměrněného napětí, aby se na výstupu neobjevovaly skokové změny.
- Úkolem stabilizátoru (S) je zajistit stejné napětí na výstupu při zapojení různých zátěží.
- Ochranná pojistka (P) chrání proti zkratu. Často se jedná o tavnou pojistku, která se při zkratu zničí, ale ochrání koncové zařízení, které je zpravidla mnohonásobně dražší.

Spínané zdroje

U spínaných zdrojů je spínací prvek zatěžován impulsně. Zatížení je krátkodobé a spínač může spínat i vyšší výkon, než který by byl přípustný u klasického zdroje. Také účinnost spínaných zdrojů dosahuje vyšší efektivity než klasické zdroje. Další výhodou je velikost zdrojů a efektivita převodu napětí, kde jsou velké rozdíly napětí, například z 240 V na 12 V. Mezi nevýhody spínaných zdrojů patří pomalejší reakce na změnu zátěže. Rychlé spínání spínacího prvku je zase zdrojem rušení, které je v některých případech nežádoucí.

I spínací zdroj používá transformátor, ale ze speciálních materiálů, aby byl schopen operovat v řádu desítek až stovek kHZ. Spínací zdroj může fungovat na frekvenci sítě, ten je o něco jednodušší, nicméně na obrázku 2.13 lze vidět blokové schéma obecného spínacího zdroje se zpětnou vazbou, který pracuje na vyšších frekvencích. Rychlost reakce zpětné vazby má velký vliv na udržení stabilního napětí na výstupu. Zpětná vazba ovládá spínač, který dávkuje napětí ze vstupu.



Obrázek 2.13: Blokové schéma spínaného zdroje. Za každým blokem je znázorněn příklad signálu, který z daného bloku vystupuje. Prvky jsou síťový filtr (1), usměrňovač(2), vyhlazovací filtr (3), spínač (4), transformátor (5), usměrňovač (6), výstupní filtr (7). Ve zpětné vazbě je pak zapojen referenční zdroj (9) pro porovnání signálu a řízení spínače (8).

1. Vstupní filtr odfiltrovává rušení ze sítě a zamezuje šíření rušení způsobené zdrojem do sítě.
2. Usměrňovač usměrňuje střídavé napětí na stejnosměrné.
3. Vyhlazovací filtr vyhlazuje usměrněný signál.
4. Spínač střídavě spíná a vytváří tak obdélníkový signál.
5. Transformátor mění velikost napětí obdélníkového signálu.
6. Usměrňovač usměrňuje obdélníkový signál z transformátoru.
7. Výstupní filtr vyhlazuje výstupní signál.
8. Řízení spínače pro zachování stabilního napětí.
9. Referenční zdroj porovnává signál s výstupním signálem.

Baterie

Jako baterie jsou označovány elektrochemické články. Obecně se baterie mohou rozdělit do 3 kategorií.

Suché články, které slouží k jednorázovému vybití. Suchý článek se objevuje například v podobě knoflíkové baterie v hodinkách, v podobě tužkové baterie v dálkových ovladačích. I přes svou malou velikost poskytují suché články dostatek výkonu a kapacity pro napájení jednoduchých, nenáročných zařízení. Doba výdrže přitom může dosahovat několika měsíců až jednotek let.

Akumulátory, které lze opakovaně dobíjet. Akumulátorům se věnuje tato kapitola. Opakované dobíjení je vhodné pro použití na středně výkonné zařízení. S akumulátory se lze setkat v mobilních telefonech, v nositelných zařízeních, v rádiově řízených RC modelech... U tohoto typu zařízení je cílem vydržet několik hodin provozu, kdy jsou zařízení

aktivně využívaná. Následně v době, kdy není zařízení používáno (například telefon v noci), se zařízení může dobíjet.

Poslední skupinou jsou speciální chemické palivové články pro pohon raketových motorů. Palivové články vyžadují specifický přístup, dochází k prudkým chemickým reakcím. Obtížným úkolem je postupné uvolňování látek a řízení chemických reakcí.

Akumulátory

Informace o akumulátorech pochází z [6]. Základní koncept akumulátoru je tvořen anodou, katodou a elektrolytem. V současnosti se používají akumulátory na bázi Li-ion. Lithium je lehký kov s vhodnými vlastnostmi, které se hodí pro využití Lithia jako anody. Jako katoda se používá oxid kovu. Při vybíjení akumulátoru dochází k chemickým reakcím a vodivé ionty se shromažďují na katodě. Mezi anodou a katodou se ještě nachází v akumulátoru izolační vrstva, která zabraňuje samovolné reakci uvnitř baterie. Až se nachází většina vodivých iontů na katodě, baterie je vybitá a není schopna dalšího přenosu. Dobití baterie přesune vodivé ionty zpět do původní pozice na anodu.

Nabíjení Li-ion baterie probíhá v režimu označovaném jako CCCV (Constant Current, Constant Voltage). Nejdříve je baterie nabíjena konstantním proudem, postupně jak se baterie dobíjí, roste napětí uvnitř baterie. Jakmile dosáhne baterie určité hranice napětí, napětí je udržováno a proud je omezován. Přebití baterie na příliš velké napětí by nemusela vydržet izolační vrstva uvnitř baterie, došlo by k jejímu průrazu. Baterií by tak tekl velký zkratový proud a došlo by k nenávratnému zničení akumulátoru.

Určité ochrany mohou být zavedeny i při vybíjení baterií. Například ochrana před kompletním vybitím a ztrátou napětí, ochrana proti přehřátí, řízení výstupního proudu. Akumulátory se mohou skládat z více bateriových článků, pak je vhodné hlídat rovnoměrné vybíjení a nabíjení všech článků v akumulátoru, což prodlužuje celkovou životnost.

Na principu Li-ion je založena spousta typů baterií. Například baterie LiPo používá opět Lithium, ale místo elektrolytu je Polymer. LiS zase využívá prvek Síru. Tyto chemické prvky ovlivňují vlastnosti baterie a přináší i jistá omezení. Chemické reakce mohou vyvolávat příliš velké teploty, které je potřeba externě chladit nebo se může jednat o velmi vzácné prvky, jejichž cena je mnohonásobně vyšší.

2.13 Analogově digitální převodníky

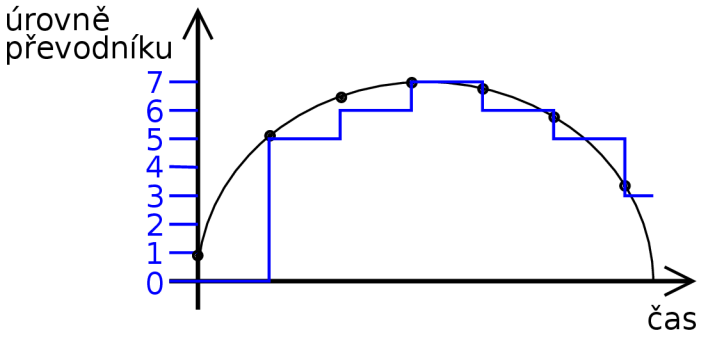
Analogově digitální převodníky jsou známé pod zkráceným názvem AD převodníky. Některé senzory poskytují na svém výstupu spojitou veličinu, zatímco počítače pracují s digitálními hodnotami. K digitalizaci spojité hodnoty se používají různé postupy a obvody podle nároků na rychlost a přesnost převodu. Některé principy jsou popsány dále.

Jako první musí být signál převeden na odpovídající napěťovou úroveň. Signál, který se objevuje na senzoru je většinou velmi malý (v řádu μV nebo mV) a bývá zesilovačem převeden na napěťový rozsah mikrokontroléru 0 - 5 V. Zesílení signálu zajišťují obvody nacházející se na senzoru.

Následuje vzorkovací obvod, který v určitých intervalech odebírá vzorky analogového signálu. Nyquistův-Shannonův teorém je požadavek na frekvenci vzorkování, aby z odebraných vzorků šel opět sestavit původní signál bez ztráty informace. Podmínkou teorému je, aby vzorkovací frekvence byla aspoň dvojnásobná než je maximální frekvence vzorkovaného signálu. [1]

Možné je vzorkovat i signály s vyšší frekvencí, ale signál musí být periodický – opakující se. Některé osciloskopy vzorkují rychlejší periodický signál odebíráním vzorků za různě dlouhé intervaly. Pokaždé se trefí do jiné části periody signálu a odebráním dostatečného počtu vzorků z více period umožňuje sestavit průběh jedné periody rychlejšího periodického signálu.

Posledním krokem je již samotný převod napětí na digitální hodnotu. Počet bitů významně ovlivňuje rozlišovací schopnost, ale i rychlost a komplikovanost převodníku. Na obrázku 2.14 je ukázána činnost AD převodníku.



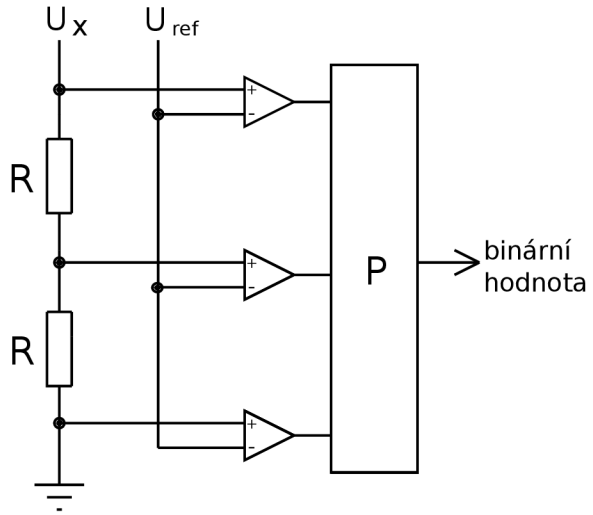
Obrázek 2.14: Činnost 3-bitového AD převodníku. Černá čára znázorňuje spojitý signál, který je výstupem ze senzoru. Černé puntíky označují vzorky odebrané vzorkovačem. Modrou čárou je vyznačen průběh digitalizovaného signálu.

Paralelní komparační převodník

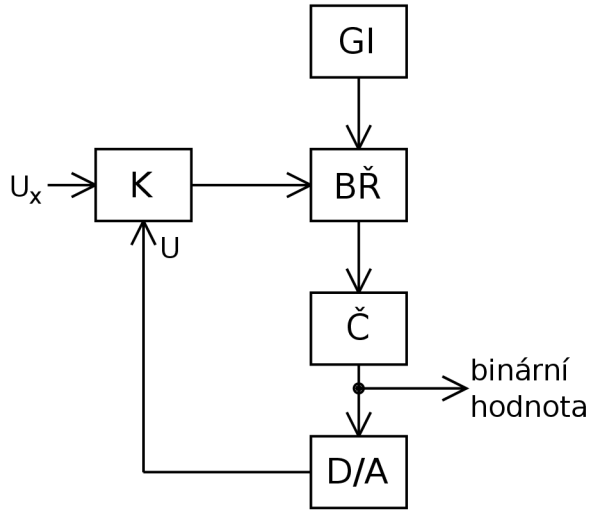
Paralelní AD převodník patří k těm rychlejším. Neznámé napětí je porovnáváno s referenčním napětím, které je rozčleněno na napěťové úrovně řadou odporů. Signály z komparátorů jsou následně převedeny na binární hodnotu. Paralelní převodník, který rozlišuje N bitů potřebuje rozeznat 2^N úrovní a potřebuje $2^N - 1$ komparátorů (pokud jsou všechny komparátory rovny 0, pak nepotřebujeme komparátor na 0 úroveň signálu). Paralelní převodníky s větším počtem bitů by zabraly velkou plochu a spotřebovali mnoho komparátorů. Větší počet bitů převádí AD převodníky s postupnou aproximací. Paralelní převodník lze vidět na obrázku 2.15

Převodník s postupnou aproximací

Na čítači se postupně nastavují hodnoty od nejvíce významného bitu MSB (Most Significant Bit) až po nejméně významný bit LSB (Least Significant Bit). Například u 8 bitového převodníku se nejdříve nastaví hodnota 1000 0000. Hodnota se porovná s neznámým napětím na vstupu, pokud je menší, nastaví se 0000 0000, pokud je hodnota větší, ponechá se 1000 0000. Pak se zkusí nastavit druhý bit 1100 0000 a opět se porovná se vstupem. Takhle se postupně nastavují všechny bity. Převod trvá delší dobu než u paralelních převodníků, výhody se projevují až při převodu vyššího počtu bitů. Například pro převod 24 bitů by paralelní převodník vyžadoval $2^{24} - 1$ komparátorů. Blokové schéma AD převodníku s postupnou aproximací je na obrázku 2.16



Obrázek 2.15: Paralelní komparační převodník. Neznámé napětí U_x je porovnáváno s úrovněmi referenčního napětí U_{ref}. Převodní blok (P) převede hodnoty z komparátorů na binární hodnotu. Tento konkrétní převodník na obrázku rozliší úrovně 0, 1, 2 a 3, jedná se tedy o 2-bitový paralelní převodník.



Obrázek 2.16: AD převodník s postupnou aproximací. Bloky představují komparátor (K), generátor impulsů (GI), blok řízení (BŘ), čítač (Č) a digitálně analogový převodník (D/A).

Kapitola 3

Softwarové prostředky k programování robota

Nízkoúrovňové operace dnes bývají většinou zapouzdřeny do funkcí z knihoven, určených pro konkrétní zařízení a součástky. Usnadňuje se tím proces programování. Bitové operace využívají pojmenovaná makra, což zvyšuje čitelnost kódu a jeho znovupoužitelnost. Vývojová prostředí poskytují kontrolu syntaktických pravidel a umožňují debugování kódu. Program ve vestavných systémech se liší od klasického programu tím, že běží v nekonečné smyčce a neměl by nikdy skončit.

Důležité je ošetřit situace, kdy může dojít k zacyklení programu či jeho uváznutí. Dobrou prevencí je vyhnout se programovým konstrukcím, které můžou vést na zackylení či uváznutí, a pokud se takovým situacím nedá vyhnout, řešením může být použití časovače a přerušení operace co způsobila zacyklení nebo uváznutí. Další softwarová omezení, která jsou úzce spojená s hardwarem, vyzívají k hledání nových řešení a jiných cest. Příkladem je nedostupnost modulu pro bezdrátovou komunikaci, nebo možnost komunikovat současně pouze s jedním zařízením na I2C sběrnici.

Následující kapitoly popisují použitý software, jeho nastavení a knihovny ke komunikaci mezi senzory a mikropočítači.

3.1 Programování Arduina

Vývoj a překlad aplikací probíhá na počítači, přeložený program se poté nahraje přes sériové rozhraní USB z počítače do paměti Arduina. K naprogramování mikropočítače Arduino se používá integrované vývojové prostředí Arduino IDE[1], při instalaci je potřeba nainstalovat USB ovladače Arduino. Programovací jazyk Arduino je založen na programovacím jazyce C. Arduino IDE poskytuje některé kontroly chyb při překladu, čímž usnadňuje vývoj aplikací.

První použití Arduino IDE vyžaduje zvolení verze vývojové desky a vybrání sériového portu ke komunikaci. To se provádí ve volbách Tools, kde je možno specifikovat Board, v tomto případě konkrétně možnost Arduino Mega or Mega2560. Další možnost Port se opět nachází ve volbách Tools. Zde je potřeba vybrat USB rozhraní v počítači, ke kterému je Arduino připojeno. K ověření správné funkčnosti slouží předvytvořené programy – nazývané *„sketches“*. Vhodným příkladem je program blikající s diodou umístěnou přímo na desce, ten se nachází ve volbách File – Examples – Basics – Blink. Na linuxu je nutno před samotným nahráním programu do Arduina přidat oprávnění uživatele ke komunikaci přes

[1] https://www.arduino.cc/en/Main/Software

sériový port příkazem[2] `sudo usermode -a -G dialout <username>`. Uživatelské jméno se dá zjistit příkazem `whoami`. Aplikování nově nastavených pravidel dojde při odhlášení a opětovném přihlášení uživatele do Linuxu. Naspodu textového editoru se nachází stavové okno s informacemi o průběhu překladu. Po úpěšném překladu je program nahrán do Arduina, zde se automaticky spustí.

Základní struktura programu

Každý program pro Arduino se musí skládat nejméně ze dvou funkcí. [4]

```
void setup() {
  ...
}
```

```
void loop() {
  ...
}
```

Funkce `setup()` provádí nastavení hardwaru Arduina, automaticky se provede při prvním spuštění programu. Jsou zde například specifikace vstupně výstupních portů, které program využívá. Funkce `loop()` se provede automaticky po provedení fáze nastavení. Jedná se o nekonečný cyklus, ve kterém se nachází kód udávající celou logiku programu.

Datové typy

Jelikož programovací jazyk vychází z jazyka C, obsahuje obdobné datové typy, nicméně jsou zde i výjimky jako absence datového typu double. Níže se nachází seznam podporovaných datových typů: [4]

- int – celá čísla v rozsahu -32 768 až 32 767
- unsigned int – celá nezáporná čísla v rozsahu 0 až 65 535
- byte – celá nezáporná čísla zahrnující 0 až 255
- boolean – nabývá hodnot true a false
- float – desetinné číslo s přesností na 15 desetinných míst
- String – text, často sloužící pro zobrazení na LCD nebo jako zpráva přenášená na počítač

Knihovna ros.h

Přidání knihovny ros.h do Arduina vyžaduje mít nainstalovaný ROS a rozšiřující balíky rosserial. Všechny potřebné kroky instalace jsou popsány v kapitole 3.2.2. Po úspěšné instalaci stačí přejít do adresáře, ve kterém se nachází knihovny Arduino a příkazem vygenerovat knihovnu ros.h. Restartováním Arduino IDE dojde ke znovunačtení knihoven, což lze ověřit dostupností sady příkladů použití v nabídce File – Examples – ros_lib.

```
cd ~/Arduino/libraries

rosrun rosserial_arduino make_libraries.py .
```

[2] https://www.arduino.cc/en/Guide/Linux#toc6

Ostatní rozšiřující knihovny

Další knihovny lze instalovat ze Správce knihoven. Ten se nachází v programu Arduino IDE, v menu položce Sketch – Include Library – Manage Libraries. Vhodné je zmínit knihovnu L3G od Pololu použitou ke čtení akcelometru, knihovnu LSM303 od Pololu ke čtení magnetometru. Knihovny pro ovládání řídicí jednotky motoru Sabertooth 2X5 jsou dostupné ke stažení na webových stránkách DimensionEngineering[3].

3.2 Robotický operační systém

Tato kapitola vychází z [5]. Robotický operační systém neboli Robot operating system (známý pod zkratkou ROS) je platforma spojující vědomosti a technologie, na kterých spolupracují akademičtí výzkumníci zabývající se počítačovou technikou a počítačovým viděním, experti na robotiku, síťový experti, lidé z průmyslu i další nadšenci. Jde o softwarový prostředek pro spolupráci jednotlivých prvků robotických systémů. Slouží ke sběru a vizualizaci dat, řízení, zajištění komunikace a spolupráce robotů, senzorů, věstavěných systémů a dalších řešení z oblasti Internet Věcí. ROS je veden jako open source projekt s velkou a aktivní komunitou, do které spadají i celosvětově známé společnosti jako NASA či BWM...

I když se v názvu ROS objevuje sousloví operační systém, nejedná se o běžný operační systém, ale o tzv. meta-operační systém vyvíjený a podporovaný na operačních systémech Ubuntu, Debian, Linux Mint a Fedora. Nové verze vychází zpravidla každé 2 roky při vydání Ubuntu LTS (Long Term Support) a podpora ROS je 5 let. Výhodou Ubuntu je dostupnost jak pro procesory x86, tak pro procesory ARM a navíc má Ubuntu početnou komunitu, což se hodí při řešení problémů, které nejspíš někdo z komunity již řešil. Použití ARM procesorů ve vestavěných systémech je vhodnější především díky nižší spotřebě a dostatečnému výkonu. Mezi často používané mikropočítače patří například ODROID-XU4 nebo Raspberry Pi 3 B+. Lze použít i spoustu dalších, důležité je ověřit dostupnost systému Ubuntu, případně Debian na daný mikrokontrolér. Další podkapitoly se zabývají instalací Ubuntu 18.04 na ODROID-XU4 s ARM procesorem a instalaci ROS Melodic.

3.2.1 Instalace Ubuntu Mate na ODROID-XU4

Jako úložiště lze použít eMMC paměťový modul nebo microSD kartu. Zde bude popsána instalace s využitím microSD karty. Velikost karty je vhodné zvolit 16 GB nebo více, podporovány jsou standardy SDHC i SDXC, tudíž lze použít i karty s kapacitami 64 GB, 128 GB. Dále bude potřeba microSD adaptér a pokud počítač nedisponuje čtečkou SD karet, zvolit některou externí USB čtečku karet.

Wiki stránky[4] pro ODROID-XU4 obsahují Ubuntu ke stažení. K vytvoření bootovacího oddílu na microSD kartě slouží software Etcher[5]. Celý proces spočívá ve vybrání úložiště, vybrání obrazu se systémem a spuštění flashování, které probíhá automatizovaně v režii programu Etcher. Lze jen upozornit na nepozornost u výběru úložiště SD karty, obvzlášť při současném používání dalších rozšiřujících pamětí, napříkald USB flash disků. Chyba při bootování ODROID-XU4 z microSD karty je znázorněna trvale rozsvícenou modrou LED diodou. Řešením je provést opětovné flashování nebo vyzkoušení jiného obrazu systému nebo alespoň jiné verze operačního systému. Následně jsou popsány specifické problémy a

[3] https://www.dimensionengineering.com/info/arduino
[4] https://wiki.odroid.com/odroid-xu4/os_images/linux/ubuntu_4.14/ubuntu_4.14
[5] https://www.balena.io/etcher/

jejich řešení při flashování, které se mohou vyskytnout na systémech Windows 10 a Linux Mint.

Informace o flashování na Windows 10

Spuštění programu se správcovským oprávněním může vyřešit neúpěšné flashování. Pokud problémy přetrvávají, u provádění opakovaného flashování pomáhá při každém pokusu ukončit Etcher a znovu spustit program.

Naformátování microSD karty korektně provádí nástroj `diskpart` v příkazové řádce. Příkazem `list disk` se zobrazí dostupné disky. `select disk <číslo disku>` zvolí disk jako aktivní. Opětovným `list disk` lze zkontrolovat a ujistit se, který disk je vybrán a nachází se u něj symbol `*`. Příkaz `clean` provede samotné smazání obsahu microSD karty.

MicroSD karta se obnoví provedením formátování popsaném v předchozím odstavci a následném vytvoření nového oddílu s patřičným formátováním např. FAT32 nebo NTFS pomocí nástroje **Správa disků**, který je dostupný po rozkliknutí nabídky start pravým tlačítkem.

Informace o flashování na Linux Mint

K obnovení microSD karty poslouží software GParted. Je třeba si dát pozor při volbě úložiště a správně zvolit microSD kartu a odstranit oddíly na ní vytvořené. Následně se dá vytvořit nový oddíl a zvolit správné formátování. Defaultní je ext4, vhodné je zvolit FAT32, NTFS nebo exFAT, které jsou podporovány i na Windows. Ke všem úkonům jsou vyžadována `sudo` oprávnění.

3.2.2 Instalace ROS Melodic na Ubuntu Mate

Instalace vyžaduje internetové připojení, které je možno přivést do ODROID-XU4 pouze přes ethernetový kabel. Domácí router nemusí být vždy po ruce a dvě následující podkapitoly popisují, jak sdílet wifi připojení z laptopu s operačním systémem Windows 10 nebo Linux Mint přes kabel do ODROID-XU4.

Sdílení wifi připojení přes kabel na systému Windows 10

Nabídka, dostupná po rozkliknutí start pravým tlačítkem, obsahuje volbu **Síťová připojení**. V sekci **Ethernet** se nachází možnost **Změnit možnosti adaptéru**. Následně je třeba vybrat rozhraní s dostupným internetovým připojení, tedy wifi rozhraní a pravým tlačítkem rozkliknout nabídku a vybrat **Vlastnosti**. Na kartě **sdílení** zaškrtnout možnost **Umožnit ostatním uživatelům v síti využívat připojení k internetu tohoto počítače** a z nabídky zvolit rozhraní, ke kterému je připojen ODROID-XU4.

Sdílení wifi připojení přes kabel na systému Linux Mint

Provést určité kroky je potřeba jak na straně laptopu, tak na ODROID-XU4. Prvním krokem je propojení obou zařízení ethernetovým kabelem. Na laptopu v nastavení **Síťová připojení** se nachází konkrétní drátové připojení. Po jeho rozkliknutí na kartě **Nastavení IPv4** je položka **Metoda**. Zde je potřeba zvolit **Sdíleno s jinými počítači**.

Obdobný postup se opakuje na zařízení ODROID-XU4, ale položku **Metoda** nastavit jako **Ruční**. Na laptopu se příkazem `ifconfig` dá zjistit IP adresa a maska podsítě, potřebná pro vyplnění ručního nastavení na ODROID-XU4. IP adresa je zpravidla ve tvaru

10.42.0.**X**, kde za X lze dostadit hodnotu 2 – 254. Například lze použít adresu 10.42.0.42, maska podsítě je 255.255.255.0. Výchozí brána má adresu stejnou, jako adresa na laptopu, tedy 10.42.0.1. Jako poslední zbývá vyplnit DNS server, který má opět stejnou adresu 10.42.0.1. [3]

Po uložení změn zbývá jen jejich aplikování. K tomu se stačí odpojit a znovu připojit do sítě, která propojuje ODROID-XU4 a laptop. Odpojení a opětovné připojení se musí provést na laptopu i na ODROID-XU4.

Aktuální kroky instalace se nachází na wiki stránkách[6] ROS.

Prvním krokem je povolení repozitářů „restricted“, „universe“ a „multiverse“

Zaškrtávací pole jsou v nastavení **Software & Updates**.

Přidání ROS repozitáře pro stahování softwaru z packages.ros.org

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release
-sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Nastavení veřejného zabezpečovacího klíče

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-ke
y C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Aktuální klíč se může lišit. Dostupný je na ROS wiki stránkách[7] v sekci instalace u konkrétní verze ROS pro dané operační systémy.

Před instalací je doporučeno aktualizovat knihovny a nainstalovat aktualizace

```
sudo apt update && sudo apt upgrade
```

Instalace ROS Melodic

Na výběr jsou tři různé edice, které se liší nainstalovanými balíky v základu. Já jsem zvolil edici **base**, která obsahuje základní balíky pro funkčnost a komunikaci bez grafických nástrojů. Edice **desktop** obsahuje navíc balíky rqt a rviz, což jsou grafické 2D a 3D nástroje. Edice **destop-full** navíc obsahuje 2D a 3D simulační nástroje. Při potřebě dalších rozšíření se dají balíky samostatně doinstalovat později.

1. `sudo apt install ros-melodic-ros-base`
2. `sudo apt install ros-melodic-desktop`
3. `sudo apt install ros-melodic-desktop-full`

Zobrazení dostupných balíků provede následující příkaz. Seznam je poměrně obsáhlý a doporučuji použít druhou variantu, která provádí výpis postupně.

```
apt search ros-melodic

apt search ros-melodic | more
```

[6] http://wiki.ros.org/ROS/Installation
[7] http://wiki.ros.org

Rozšiřující balíky lze doinstalovat následujícím příkazem. PACKAGE zastupuje název balíků. Názvy použitých balíků pro komunikaci ROS s Arduinem jsou rosserial-arduino a rosserial.

```
sudo apt install ros-melodic-PACKAGE
```

Před použitím ROS je nutné provést inicializaci rosdep

```
sudo apt-get install python-rosdep

sudo rosdep init

rosdep update
```

Tato sada příkazů provede instalaci nástroje rosdep, potřebnou inicializaci repozitáře a aktualizace. Instalace je specifická pro systém Ubuntu.

Nastavení prostředí

```
echo "/opt/ros/melodic/setup.bash" >> ~/.bashrc

source ~/.bashrc

printenv | grep ROS
```

První dva příkazy zařídí, že proměnné prostředí, které využívá ROS jsou automaticky přidány do nově spuštěné instance bash. Třetí příkaz ověřuje přidání, klíčové jsou `ROS_ROOT` a `ROS_PACKAGE_PATH`.

3.2.3 Základní struktura ROS

- **Nodes** jsou spustitelné procesy určené ke vzájemné komunikaci.
- **Message** je struktura složená ze základních datových typů (integer, floating point, boolean, ...) nesoucí data potřebná ke komunikaci.
- **Topics** slouží k přenosu zpráv a pravidlem je mít pro každý účel zvláštní topic, například jeden topic pro senzory a druhý pro ovládání motoru.
- **Master** pokytuje služby všem uzlům – nodes – v systému ROS. Navádí uzly ke správnému doručování zpráv.
- **rosout** je základní topic pro přenos zpráv, něco jako stdout v jazyku C.
- **roscore** je příkaz, který provede spuštění ROS Master, zajistí logování zpráv a základní funkcionalitu pro fungování systému ROS.

Těchto šest položek tvoří jádro celé komunikace [5]. V komunikaci se nejčastěji budou vyskytovat pojmy topic a message.

Topic lze chápat jako téma. Při rozhovoru lidí se nejdříve navrhne téma a o něm probíhá rozhovor. Postupně se probírají jednotlivá témata a nemíchají se všechny informace dohromady. Stejným pravidlem se řídí topic. Vhodné je vytvořit pro každou oblast jiný topic, například zvlášť pro informace z ultrazvukových senzorů, zvlášť pro IMU senzor a zvlášť pro pokyny k ovládání motoru.

Messages udávají strukturu zprávy. Messages lze chápat jako jakýsi komunikační protokol. Pokud se budou bavit dva lidé s navzájem si cizím jazykem, nastává problém, jak informace správně interpretovat. Message stanoví pevnou strukturu zpráv, například vzdálenost senzoru jako integer číslo. Tím získá sdělovací i přijímající zařízení povědomí o tom, jaký typ informací se očekává. Praktická ukázka s vysvětlením přenosu dat je v implementační sekci 4.6.

Master je centrem komunikace. Přijímá žádosti od uzlů k odebírání nebo poskytování dat. Master pak řídí přeposílání dat pouze určitým uzlům, které mají o data zájem a udržuje informaci o aktuálně připojených uzlech synchronizačními zprávami. Uzly představují spustitelné aplikace, které se hlásí masteru.

Kapitola 4

Implementace řešení

Kapitola o implementaci popisuje konkrétní řešení, části kódů, zařízení a softwarové prostředky popsané v kapitolách 2 a 3. Jde o aplikaci znalostí a vědomostí, které vedou k propojení všech částí a vytvoření tak funkčního celku – robota Trilobot. Kapitola je zaměřena na softwarovou část projektu a na zapojení obvodů. Také je zde popsán průběh komunikace mezi zařízeními. Mechanická konstrukce robota je popsána v sekci 2.1.

4.1 Získání adresy zařízení na I2C sběrnici

Výrobcem nastavená adresa by měla mít hodnotu `E0`, nicméně není komplikované informaci ověřit. Arduino obsahuje v nabídce File – Examples – Wire příklad pojmenovaný **i2c_scanner**. Ten stačí jednoduše přeložit a nahrát do Arduina.

Program postupně projde adresy od 1 až do 127, používá funkce `beginTransmission()` a `endTransmission()` z knihovny `Wire.h`. Pokud je nějaké zařízení s adresou poslanou jako parametr funkce `beginTransmission(adresa)` připojené, povede se navázání kontaktu a funkce `endTransmission()` skončí s návratovým kódem 0 a provede se vypsání adresy. Vypisování adres lze sledovat v rozhraní **Serial Monitor**, které je přístupné z nabídky Tools.

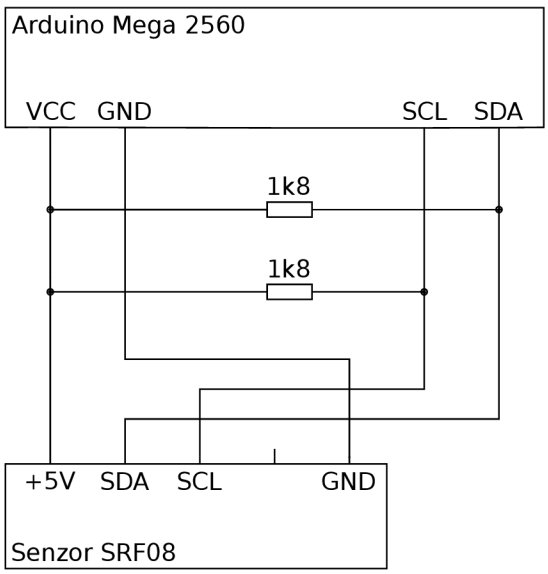
4.2 Připojení, adresování a používání senzorů SRF08

Senzory jsou vyráběny pro připojení na sběrnici I2C. Označení vývodů a schéma připojení s pull-up rezistory je na obrázku 4.1. Stejně jako na obrázku mají senzory SRF08 i ve skutečnosti pět vývodů, přičemž by se čtvrtý pin neměl připojovat a slouží k prvotnímu naprogramování při výrobě senzoru. Pull-up rezistory definují napěťovou úroveň v případě, kdy se obvod dostane do stavu vysoké impedance – pin není aktivní a obvod je rozpojený. Pokud pin není aktivní, je v obvodu udržována napěťová úroveň HIGH (5V). Při aktivování pinu dojde k sepnutí obvodu a následně se v obvodu objeví logická úroveň LOW (0V).

Senzory jsou z výroby nastaveny na hexadecimální adresu `E0`. Adresu lze změnit na `E2`, `E4`, `E6`, `E8`, `EA`, `EC`, `EE`, `F0`, `F2`, `F4`, `F6`, `F8`, `FA`, `FC`, `FE`. Na jednu sběrnici I2C tedy může být připojeno až šestnáct SRF08 senzorů tak, aby nedocházelo ke kolizi adres.

Prvním problémem je, že všechny adresy z předchozího odstavce mají 8 bitů a knihovna Arduina `Wire.h` pracuje se 7 bitovými adresami. Aby se z 8 bitové adresy stala 7 bitová, dojde ke smazání posledního nejméně významného bitu. Adresa `E0`, která má tvar v binární podobě `1110 0000` přijde o poslední bit a vznikne adresa `111 0000`, což je hexadecimální

hodnota `70`. Převedené 8 bitové adresy od `E0` až po `FE` na 7 bitové, které lze použít pro adresování jsou tedy `70`, `71`, `72`, `73`, `74`, `75`, `76`, `77`, `78`, `79`, `7A`, `7B`, `7C`, `7D`, `7E`, `7F`.



Obrázek 4.1: Zapojení ultrazvukového senzoru SRF08 s pull-up rezistory.

Změna adresy senzoru SRF08

Provedení změny adresy vyžaduje 2 věci. Zaprvé musí být na sběrnici připojen pouze jeden senzor s adresou jenž se bude měnit. Všechny ostatní senzory se stejnou adresou se musí odpojit. Zadruhé se na adresu starého senzoru musí zapsat sekvence tří hexadecimálních hodnot a následně nová hodnota adresy. Kód pochází z Arduino programu, který slouží jako příklad použití. [10]

```
Wire.beginTransmission(stara_adresa);
Wire.write(byte(0x00));
Wire.write(byte(0xA0));
Wire.endTransmission();

Wire.beginTransmission(stara_adresa);
Wire.write(byte(0x00));
Wire.write(byte(0xAA));
Wire.endTransmission();

Wire.beginTransmission(stara_adresa);
Wire.write(byte(0x00));
Wire.write(byte(0xA5));
Wire.endTransmission();

Wire.beginTransmission(stara_adresa);
Wire.write(byte(0x00));
Wire.write(byte(nova_adresa));
Wire.endTransmission();
```

Stará adresa se uvádí v 7 bitovém tvaru, například původní adresa `70`, nová adresa se uvádí v 8 bitovém tvaru, například adresa `EE`.

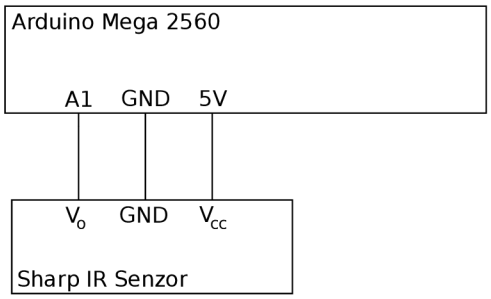
Čtení dat ze senzoru SRF08

Získání dat ze senzoru probíhá v pěti krocích. [10]

```
// 1. krok - vyzvání senzoru ke změření dat v centimetrech
Wire.beginTransmission(adresa_senzoru);
Wire.write(byte(0x00)); // adresování příkazového registru senzoru
Wire.write(byte(0x51)); // příkaz pro meření v centimetrech
Wire.endTransmission();

// 2. krok - prodleva potřebná ke změření vzdálenosti
delay(70);

// 3. krok - adresování registru senzoru, ve kterém jsou naměřená data
Wire.beginTransmission(adresa_senzoru);
Wire.write(byte(0x02));
Wire.endTransmission();

// 4. krok - vyžádání naměřených dat
Wire.requestFrom(112, 2);

// 5. krok - uložení dvou přečtených bajtů do proměnné reading
if (2 <= Wire.available()) {
    int reading = Wire.read(); // načtení vyšších osmi bitů
    reading = reading << 8; // posun bitů na vyšší pozici
    reading |= Wire.read(); // načtení nižších osmi bitů
}
```

4.3 Připojení a získávání dat z infračerveného senzoru

Na obrázku 4.2 lze vidět připojení senzoru Sharp GP2Y0A41SK0F k Arduinu. Trochu atypické je použití 3-pinového konektoru JST PH, který má rozteč pinů menší než standardní propojovací vodiče Arduina.



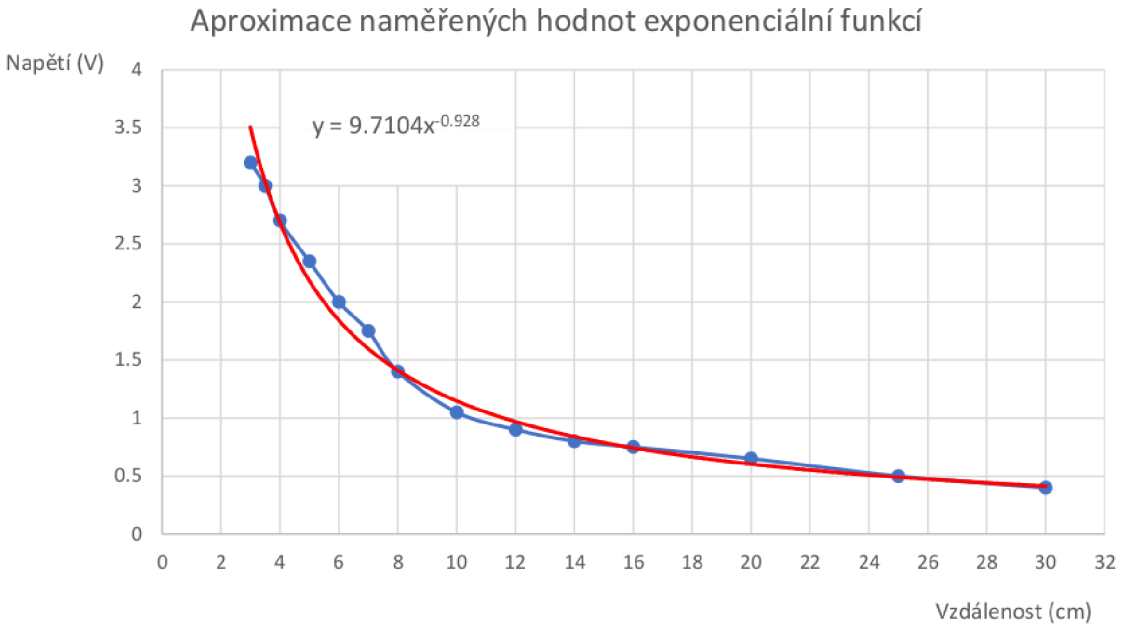
Obrázek 4.2: Schéma zapojení IR senzoru Sharp GP2Y0A41SK0F k Arduinu.

Čtení dat z IR senzoru Sharp GP2Y0A41SK0F

Načtení dat zajistí jediná funkce `analogRead()`. Rozsah vstupního signálu může být 0 - 5 V, tato hodnota je pak navzorkována a převedena na digitální hodnotu. Arduino Mega 2560 má k dispozici 10 bitů, dokáže tedy rozlišit 1024 úrovní vstupního analogového signálu. Pro jiné verze Arduina se počet dostupných bitů může lišit[1].

Technická dokumentace[2] senzoru neuvádí přímý postup převedení získané binární hodnoty na vzdálenost. Dostupný je pouze nelineární graf s naměřenými hodnotami. Hodnoty je možno z grafu vyčíst a přenést je do tabulkového programu Microsoft Excel, který zároveň umí všechny body aproximovat funkcí a vytvořit tak funkční předpis pro získání hodnot. Aproximaci i s přenesenými body lze vidět na obrázku 4.3. Naměřená binární hodnota se nejdříve musí převést na napětí z odpovídajícího rozsahu vynásobením 5 a podělením 1024, poté lze hodnotu dosadit do rovnice za `x`.

```
float value = analogRead(A1); // načtení hodnoty ze senzoru
value = value * 5 / 1024; // převedení na rozsah 0 - 5 V
value = 9.7104 * pow(value, -0.928); // výpočet vzdálenosti
```



Obrázek 4.3: Modře jsou vyznačeny naměřené body z technického manuálu, červenou barvou je vyznačena aproximovaná funkce.

Tento postup umožňuje vzniknutí několika chyb. První chybu způsobuje lidský faktor při nepřesném odečítání hodnot z grafu, další chyba vzniká při aproximaci funkce. Senzor také pracuje s určitou přenosti, kterou ovlivňuje řada faktorů, takže senzor není vhodný na přesná meření v řádu milimetrů.

Například provedení pokusu měření ze vzdálenosti 4 cm a použitím překážky z různých materiálů, bílý papír (papír sloužil i jako referenční překážka v grafu technické dokumen-

[1] https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/
[2] https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a41sk_e.pdf

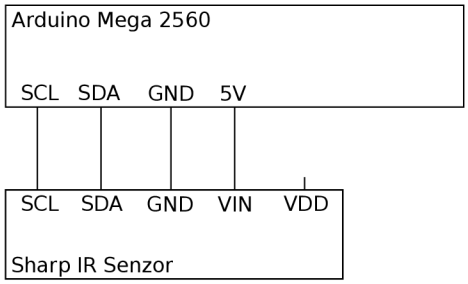
tace), počítačový zdroj z černého plastu a pravítko z průhledného plastu, ukázalo poměrně odlišné výsledky, které jsou zaznamenány v tabulce 4.1.

Materiál	Naměřená binární hodnota	Vypočítaná vzdálenost v cm
Papír	575	3,7255
Černý plast	550	3,8824
Průhledný plast	213	9,3630

Tabulka 4.1: Výsledky měření vzdálenosti IR senzorem ze vzdálenosti 4 cm a s použitím překážek z různých materiálů

4.4 Zapojení a čtení dat z MiniIMU-9 v3

MiniIMU-9 v3 je vyráběno k připojení na I2C sběrnici. Schéma zapojení je na obrázku 4.4. Přívod VDD se může použít k napájení pomocí 3,3 V. V případě napájení 5 V je určen přívod VIN, který obsahuje regulátor napětí. Na součástce MiniIMU-9 v3 jsou dostupné senzory pro měření zrychlení, rotace i digitální kompas k určení natočení vzhledem k severu. Gyroskop[3] a magnetometr[4] se adresují zvlášť. U obou zařízení lze upravit adresu pouze jedním bitem, takže na sběrnici mohou být maximálně dva gyroskopy nebo magnotemetry s unikátními adresami.



Obrázek 4.4: Schéma zapojení MiniIMU-9 v3 k Arduinu..

Následující kódy vychází z programů, které společnost Pololu poskytuje jako příklady použití svých knihoven L3G.h a LSM303.h.

```
L3G gyro;
LSM303 compass;

void setup() {
  gyro.init(); // získá adresu a typ gyroskopu
  gyro.enableDefault(); // zapíše do registru gyroskopu a aktivuje ho

  comapss.init(); // získá adresu a typ kompasu
  compass.enableDefault(); // aktivuje kompas zapsáním do registru
}
```

[3] https://www.pololu.com/file/0J731/L3GD20H.pdf
[4] https://www.pololu.com/file/0J703/LSM303D.pdf

```
void loop() {
  gyro.read(); // změření a uložení hodnot gyroskopu
  compass.read(); // změření a uložení hodnot kompasu

  float compass_angle = compass.heading(); // úhel od severu 0 - 360
  int gyro_x = gyro.g.x; // rotace v ose x
  int gyro_y = gyro.g.y; // rotace v ose y
  int gyro_z = gyro.g.z; // rotace v ose z
}
```

4.5 Ovládání motorů přes řídicí jednotku Sabertooth 2X5

Schéma připojení a nastavení mechanických přepínačů se nachází v kapitole 2.7, použitou knihovnu `SabertoothSimplified.h` poskytuje ke stažení DimensionEngineering[5].

Každý motor se ovládá zvlášť informací, která je obsažena v jednom bajtu. Ovládání motoru má vyčleněné hodnoty -127 až 127. Hodnota 0 motor zastaví, hodnota -127 je maximální chod vzad, hodnota 127 maximální chod vpřed.

Následující kód vychází z programu SimpleExample, který vytvořila společnost Dimension Engineering LLC v roce 2012. Program je dostupný jako příklad použití v knihovně SabertoothSimplified pro Arduino.

```
// objekt, který představuje řadič motoru
SabertoothSimplified ST;

void setup() {
    // rychlost komunikace, která je nastavena na mechanických přepínačích
    SabertoothTXPinSerial.begin(9600);
}

void drive(motor1_value, motor2_value) {
    ST.motor(1, motor1_value); // Ovládání prvního motoru
    ST.motor(2, motor2_value); // Ovládání druhého motoru
}
```

4.6 Přenos dat mezi Arduinem a ODROID-XU4

Předpokladem je mít nainstalovaný a funkční ROS. Popis instalace lze najít v skeci 3.2.2. Následuje kód pro publisher a subscriber na Arduinu, který vychází z wiki stránek[6]. Přenos probíhá přes prostředníka, aplikaci roscore, která řídí komunikaci.

```
std_msgs::Int16 sensor_data; // struktura zprávy s integer číslem

// objekt publisher zveřejňující data do vlákna sensor_data
ros::Publisher data_publisher("sensor_data", &sensor_data);
ros::NodeHandle nh; // uzel, aplikace komunikující s ROS
```

[5] https://www.dimensionengineering.com/info/arduino
[6] http://wiki.ros.org/rosserial/Overview/Publishers%20and%20Subscribers

```
void setup() {
    nh.initNode(); // inicializace uzlu

    // přihlášení publisheru do ROS
    nh.advertise(data_publisher);
}

void loop() {
    // vložení dat ze senzoru do zprávy
    sensor_data.data = read_srf08(adresa);

    data_publisher.publish(sensor_data); // vložení zprávy do bufferu
    nh.spinOnce(); // odeslání dat z bufferu do ROS
}
```

Velmi podobně probíhá získávání dat, ale jsou zde odlišnosti v použití funkcí.

```
ros::NodeHandle nh; // uzel, aplikace komunikující s ROS

// funkce, která je volána při přijetí zprávy
void motor_sub_callback(const std_msgs::Int16& motor_value) {
    int motor_value = (motor_value.data); // uložení dat ze zprávy
}

// objekt subscriber, přijímající zprávy z vlákna motor_topic
ros::Subscriber<std_msgs::Int16> sub("motor_topic", &motor_sub_callback);

void setup() {
    nh.initNode(); // inicializace uzlu
    nh.subscribe(sub); // přihlášení subscriberu do ROS
}

void loop() {
    nh.spinOnce(); // načtení dat přijmutých v bufferu
    delay(100); // prodleva 100 ms
}
```

Použití callback funkce má jednu velkou výhodu. Pokud do stejného topicu chodí více typů zpráv (například Int a Float), callback funkce má jako argument datový typ zprávy. Lze tak odebírat z jednoho vlákna pouze zprávy s daným datovým typem a provádět filtrování zpráv v rámci jednoho vlákna (topicu).

Strukturu (publisher a subscriber)[7] má i grafická aplikace na straně Odroidu. Komunikace na obou zařízeních probíhá následovně. Arduino cyklicky posílá data ze senzorů a očekává data k řízení motoru. Odroid neustále posílá zprávy k řízení motorů a očekává data ze senzorů pro jejich reprezentaci na displeji. Zprávy řízení motoru obsahují 0 (zastavení motorů) a při stisknutém tlačítku se obsah zprávy změní na určitou hodnotu (např. 32 pro

[7] http://wiki.ros.org/roscpp_tutorials/Tutorials/WritingPublisherSubscriber

pohyb vpřed nebo −32 pro pohyb vzad). Po uvolnění tlačítka se opět obsah zprávy změní na 0.
Celou komunikaci řídí centrální ROS aplikace roscore (spustitelná příkazem `roscore` v terminálu). Ta se stará o synchronizaci jednotlivých uzlů a provádí šíření zpráv odpovídajicím zařízením.

Jako prostředník mezi aplikací roscore a uzlem Arduina je ještě jedna aplikace rosserial. Arduino komunikuje přes sériové rozhraní USB, skrze které jsou data serializována. Na straně Odroidu je tak nutné data sestavit do podoby zprávy pro roscore a data deserializovat. A přesně o to se stará aplikace rosserial[8] spustitelná následujícím příkazem.

```
rosrun rosserial_python serial_node.py _port:=/dev/ttyUSB0 _baud:=9600
```

Data zveřejněná do určitého vlákná (topicu) lze zobrazit a ověřit si funkčnost komunikace pomocí příkazu v terminálu.

```
rostopic echo [název topicu]
```

Grafická aplikace k ovládání robota

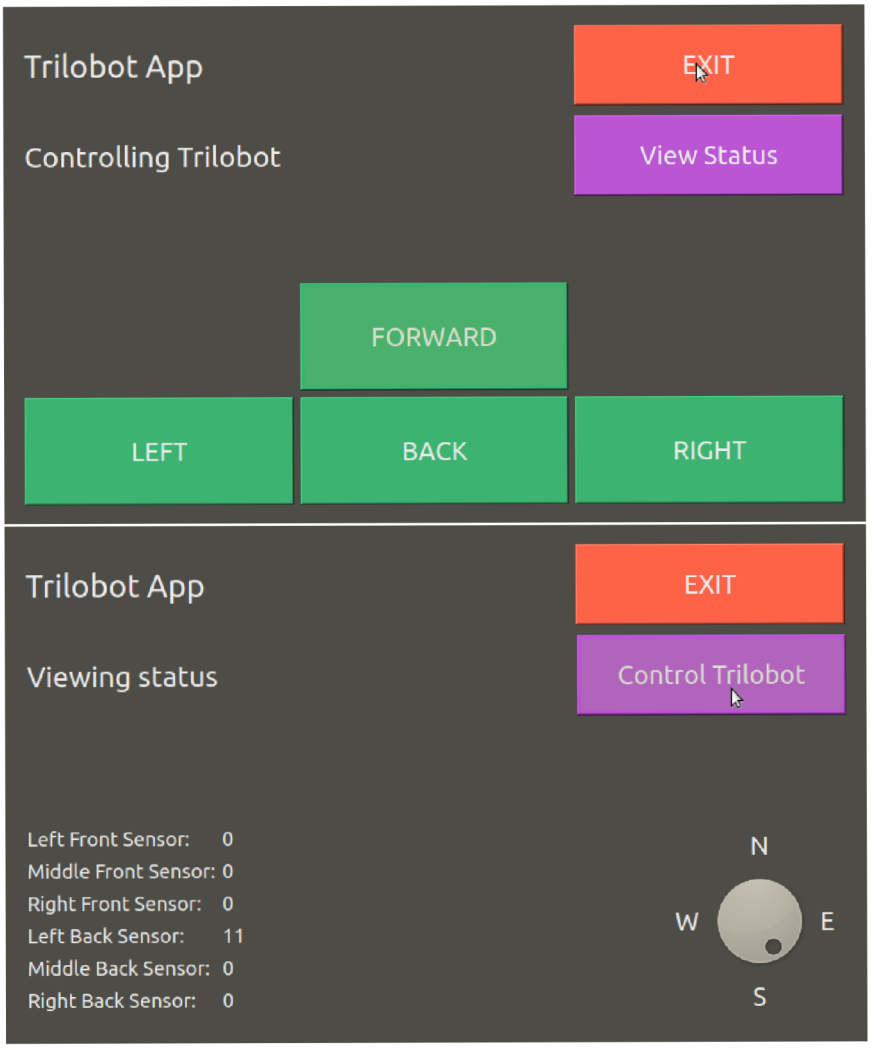
Aplikace je vytvořená v jazyce C++ (implementace ROS se jmenuje roscpp[9]) a nástrojem Qt Creator[10] pro tvorbu grafických rozhraní. Aplikace využívá překladový nástroj catkin[11], sloužící pro překlad ROS projektů. Konkrétně byl použit příkaz `catkin_create_qt_pkg` pro vytvoření struktury souborů základní grafické aplikace se závislotmi na ROS a Qt hlavičkových souborech. A přeložení aplikace se provede příkazem `catkin_make` z hlavního adresáře projektu. Aplikace sestává z více souborů a popisovat zde detailně jednotlivé části mi osobně nepřijde příliš praktické. Zkráceně, aplikace využívá podobné struktury Publisher a Subscriber, které se nachází na začátku této kapitoly při popisu Arduina. Hlavní odlišností je použití časovače. Časovač periodicky každých 100 ms volá funkci, která posílá a přijímá aktuální data. Zatímco Arduino je tvořeno pro vestavné aplikace, které běží neustále, takže nekonečná smyčka je zajištěna automaticky cyklickým voláním funkce `loop()`, v C++ tento efekt zastává právě časovač a opakované volání funkce.

Funkce `spinOnce()` se stará nejen o odesílání a přijímání zpráv z lokálního bufferu, ale zastává i funkci synchronizace mezi aplikací (uzlem) a roscore. Perioda volání tak musí být dostatečně krátká, aby ROS neoznačil uzel jako odpojený a pokaždé vytvářel nové spojení.

Grafická stránka byla navržena s využitím aplikace Qt Designer, která je součástí balíku Qt Creator. Grafické prvky jsou propojeny s funkcemi v C++ s pomocí signálů a slotů. Pokaždé když grafický prvek zaznamená určitou interakci, vytvoří signál – událost (event). Událost zachytí funkce (slot) a provede se. Program tak nemusí neustále testovat všechny grafické prvky, ale k aktivaci dojde pouze při vytvoření události. Mezitím se může program věnovat jiným činnostem, právě jako je periodické zasílání zpráv do ROS.

Aplikace je rozdělena na 2 části, ovládání robota a čtení stavu senzorů. Aplikace běží v celoobrazovkovém režimu a obsahuje prvky dostatečně velké pro vhodné ovládání dotykem na malé obrazovce. Vypisovány jsou informace o vzdálenosti změřené jednotlivými senzory a jednoduchá grafická reprezentace kompasu. Výslednou grafickou podobu aplikace lze vidět na obrázku 4.5.

[8] https://wiki.ros.org/rosserial_arduino
[9] https://wiki.ros.org/roscpp
[10] https://www.qt.io/
[11] https://wiki.ros.org/catkin



Obrázek 4.5: Grafická aplikace pro ovládání robota na dotykovém displeji.

Kapitola 5

Návrhy pro rozšíření

Dostupný je základ pro získávání dat ze senzorů, pohyb robota a fungující komunikace mezi třemi většími celky, které tvoří senzory, Arduino spolupracující se senzory a ODROID-XU4, na kterém běží ROS zajišťující komunikaci s Arduinem. Způsobů, kterými lze vylepšit robota je celá řada a některé z nich se objevují v této kapitole.

Použití baterie místo zdroje napájení

Do řídicí jednotky motoru i mikropočítače ODROID-XU4 je přiváděna energie pomocí zdrojů napájení, což značně limituje pohyb robota – jeho hlavní schopnost. Vhodným řešením by bylo použití dvou baterií s dostatečným výkonem. Zvlášť baterii pro motory a zvlášť pro mikropočítač. Napájení z jedné společné baterie by mohlo představovat problém, kdy motor vyžaduje nárazově velké množství energie. Skokově se zmenší dostupná energie pro mikropočítač, a pokud by klesla pod vyžadované minimum, může dojít k neočekávanému chování mikropočítače nebo k jeho vypnutí.

Senzor detekce propadlin

Stejně jako vystouplá, pevná překážka se v cestě robota může objevit propadlina, příkladem je schodiště. Původně bylo v plánu použít IR senzor k detekci propadlin, ale jeho použití by přineslo řadu nevýhod. Naměřené výsledky v tabulce 4.1 ukazují, že povrch může dramaticky měnit výsledky. Navíc by se komplikoval sjezd a výjezd drobných nájezdových ramp. IR senzor tak lze použít jako doplňující senzor, ale samotně nemůže zastávat funkci detekce propadlin.

Přidání kamery

Dostupnost hloubkových kamer s možností 3D mapování okolí je omezenější a produkce spousty z nich již skončila (Xbox Kinect, Primesense Carmine). Hloubková kamera by dovolila značně rozšířit schopnosti robota například v podobě rozpoznání nájezdových ramp, rozpoznávání překážek, zda je možné překážku objet... Menší nevýhodu může představovat vyšší výpočetní výkon, který zvýší spotřebu.

Postačující by bylo i použití obyčejné kamery, kterou by šlo aktivovat pouze při výskytu neočekávané události (výskyt překážky, nehoda způsobená převrácením robota) a pořídit snímek nebo krátký záznam, který by pomohl vyjasnit situaci.

Pokročilé algoritmy pro plánování trasy

Na trase robota z bodu A do bodu B se může vyskytovat libovolný počet překážek. Tyto překážky navíc mohou nabývat libovolných rozměrů. Kritické jsou situace, když robot přijede na pomezí zdi a volného průjezdu. Situace se musí vhodně detekovat a zajistit, aby robot objel zeď průjezdem místo toho, aby se snažil pokračovat podél zdi.

Rozšíření funkcionality o GPS (Global Positioning System) modul může umožnit určení polohy robota, určení polohy cílového bodu a umožnit tak robotu pokusit se objet překážku ze strany, která vede na kratší trasu do cílové destinace.

Detekce poruch

Pohybem robota a vlivem různých vibrací může dojít k uvolnění propojů. Detekcí poruch a rozpojených částí se může předejít chybným informacím například ze senzorů, které můžou vést k nežádoucímu chování robota.

Sledován může být i stav baterií použitých pro napájení motoru a mikropočítačů. Nízká kapacita může vést k automatickému navrácení robota zpět na základnu, a nebo uvést robota aspoň do stabilní a ustálené polohy.

Bezdrátové ovládání robota

Řízení robota dálkově je jistě zajímavá funkce. Bluetooth modul má dnes většina mobilních zařízení. Nebo lze použít externí Wi-Fi modul. Dálkově by tak bylo možné sledovat polohu robota, stav robota, stav baterie a robota řídit. Centrální jednotka by pak mohla shromažďovat různá statistická data a koordinovat pohyb více robotů. Otázka se tak přesouvá na dostupnost síťového připojení a bezpečnost síťového přenosu.

Kapitola 6

Závěr

Cílem této práce bylo nastudovat konstrukci robota Trilobot, mikropočítače, senzory, enkodéry, Kinect, software Robotický operační systém a získané znalosti využít při implementaci a sestavování robota Trilobot.

Záměr práce jsem splnil a snažil jsem se efektivně obsáhnout všechny body zadání. Sestavený robot využívá k měření vzdálenosti ultrazvukové senzory připevněné plastovými úchyty vytisknutými na 3D tiskárně. Naměřená data ze senzorů a řídicí signály pro motor jsou přenášena skrze Robotický operační systém (ROS), který poskytuje aplikační rozhraní pro komunikaci mezi různými zařízeními. Implementoval jsem potřebné programy, otestoval jsem pohyb robota a údaje dodávané ze senzorů. Navíc jsem vytvořil jednoduchou grafickou aplikaci, která umožňuje ovládání robota a čtení údajů ze senzorů na dotykové obrazovce.

Při vytváření cílového řešení jsem se setkal s některými netypickými problémy, z nichž některé se mi nepodařilo vyřešit. První problém představovalo použití pinu TX1 na mikropočítači Arduino pro řízení ovladače motoru Sabertooth 2X5. Při inicializaci Arduina a komunikaci s ROS se na pinu TX1 objevovaly signály, které negativně ovlivňovaly chování motorů. Použití pinu TX14 místo TX1 problém vyřešilo.

Další problém je řízení motorů. Levý motor je cca první sekundu rychlejší než pravý a při rovném pohybu tak robot zpočátku vybočí na stranu.

Posledním zpozorovaným problémem jsou některé ultrazvukové senzory. Většina senzorů vrací jako naměřenou vzdálenost 0, přitom všechny senzory mají jednotné nastavení i obvodové zapojení a data jsou čtena stejnou funkcí. Vyzkoušel jsem použít různé pull-up rezistory v zapojení, měřit napětí multimetrem, prodlužovat prodlevu ke změření vzdálenosti a vše vypadá vpořádku. Přičemž se senzory hlásí i při testování I2C sběrnice a odpovídají zprávou na dotaz o jejich verzi. Provedl jsem otestování na bezproblémovém senzoru a ten měří vzdálenosti 4 až 15 cm s přeností $\pm$ 1 cm, poté se přesnost zhoršuje.

Práce mi poskytla příležitost prakticky se setkat s chybami na hardwarové úrovni, seznámil jsem se s principy a mechanismy používanými v robotice a rozšířil jsem si znalosti o programování vestavných systémů, jimiž bych se chtěl zabývat i nadále.

V práci by se dalo pokračovat odstraněním současných neduhů. Provedením analýzy signálu motoru na osciloskopu a podle naměřených výsledků signál softwarově nebo hardwarově upravit. Otestovat více senzorů vzdálenosti. Jako další rozšíření se vybízí bezdrátový modul a řízení robota na dálku nebo detekce hardwarových poruch, rozpojení obvodů a sledování stavu robota, jako je například stav baterie.

Literatura

[1] Bucci, D. *Analog Electronics for Measuring Systems.* ISTE Ltd, 2017. ISBN 978-1-78630-148-2.

[2] Chepuri, S. *Touch screen technology* [online]. researchgate.net, únor 2014 [cit. 2020-05-26]. Dostupné z: https://www.researchgate.net/publication/260134012_Touch_screen_technology.

[3] Mau, S. a Graham, J. *Share wireless Internet connection through ethernet* [online]. askubuntu.com, říjen 2013 [cit. 2020-05-27]. Dostupné z: https://askubuntu.com/questions/359856/share-wireless-internet-connection-through-ethernet.

[4] McComb, G. *Robot Builder's Bonanza.* 4. vyd. McGraw-Hill, 2011. ISBN 978-0-07-175035-6.

[5] Pyo, Y., Cho, H., Jung, R. a Lim, T. *ROS Robot Programming.* 1. vyd. ROBOTIS Co., Ltd., 2017. ISBN 979-11-962307-1-5.

[6] Reddy, T. B. *Linden's Handbook of Batteries.* 4. vyd. McGraw-Hill, 2011. ISBN 978-0-07-162419-8.

[7] Roy, R. a Bommakanti, V. *ODROID-XU4 USER MANUAL* [online]. Hard Kernel, Ltd., 2015 [cit. 2020-05-27]. Dostupné z: https://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf.

[8] Siciliano, B. a Khatib, O. *Springer handbook of robotics.* 2. vyd. Springer, 2016. ISBN 978-3-319-32550-7.

[9] Tanz, J. *Kinect Hackers Are Changing the Future of Robotics* [online]. Wired, červen 2011 [cit. 2020-05-02]. Dostupné z: https://www.wired.com/2011/06/mf_kinect/.

[10] Zambetti, N. a Tichenor, J. *I2C SRF10 or SRF08 Devantech Ultrasonic Ranger Finder* [online]. www.arduino.cc, duben 2006 [cit. 2020-05-10]. Dostupné z: https://www.arduino.cc/en/Tutorial/SFRRangerReader.