

Mendelova univerzita v Brně  
Provozně ekonomická fakulta

---

# **Návrh a řízení vše směrového podvozku**

**Bakalářská práce**

Vedoucí práce:  
Ing. Jan Kolomazník, Ph.D.

Jiří Zbořil

Brno 2019



## **Poděkování**

Chtěl bych zde poděkovat Ing. Janu Kolomazníkovi, Ph.D. za nesmírnou pomoc při vedení mé práce, jeho velký přínos při zodpovídání mých otázek a jeho cenných poznámek při konzultacích, které mě udržovaly na správné cestě k cíli. Obrovské poděkování patří mé rodině, která mě neustále podporovala a bez které bych měl těžké chvíle s dokončením této práce.



### **Čestné prohlášení**

Prohlašuji, že jsem práci **Návrh a řízení všeobecného podvozku** vypracoval samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v souladu s platnou Směrnicí o zveřejňování závěrečných prací.

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 18. května 2020

.....

podpis



## **Abstract**

Zbořil, J. Design and control of omnidirectional chasis. Brno, 2020.

Defining of communication between Raspberry Pi and Arduino boards and construction of the chassis.

**Key words:** Omnidirectional chasis, omnidirectional wheels, Arduino, Raspberry Pi

## **Abstrakt**

Zbořil, J. Návrh a řízení vše směrového podvozku. Brno, 2020.

Je zde popsán způsob komunikace mezi jednodeskovými počítači Raspberry Pi a Arduinem a konstrukce podvozku.

**Klíčová slova:** Vše směrový podvozek, vše směrová kola, Arduino, Raspberry Pi



## **Obsah**

<b>1 Úvod a cíl práce</b>	<b>13</b>
<b>2 Přehled literatury</b>	<b>14</b>
<b>3 O všeobecném podvozku</b>	<b>17</b>
3.1 Typy všeobecných kol . . . . .	17
3.2 Pohybování podvozku . . . . .	18
3.3 Matematické přístupy k řízení podvozku . . . . .	19
<b>4 Stavba podvozku a součásti</b>	<b>21</b>
4.1 Typ kol . . . . .	21
4.2 Motory . . . . .	21
4.3 Upevnění kol k motorům . . . . .	23
4.4 Návrh desky podvozku . . . . .	25
4.5 Raspberry Pi . . . . .	26
4.6 Arduino . . . . .	28
4.7 Propojení Arduina s motory . . . . .	29
<b>5 Komunikace Raspberry Pi a Arduina</b>	<b>32</b>
5.1 I2C sběrnice . . . . .	32
5.2 Způsob komunikace . . . . .	34
5.3 Konfigurace a nastavení přenosu I2C . . . . .	35
5.4 Struktura přenášených bytů . . . . .	36
5.5 Získání hodnot instrukcí z bytů na Arduinu . . . . .	37
5.6 Bitové operace použité na Arduinu . . . . .	38
<b>6 Implementace řízení podvozku</b>	<b>40</b>
6.1 Ovládání motorů . . . . .	43
<b>7 Závěr</b>	<b>45</b>
<b>8 Literatura</b>	<b>46</b>



## Seznam obrázků

Obrázek 1: Mecanumské kolo <i>Zdroj:</i> <a href="https://opencircuit.shop/Product/4-inch-100mm-Mecanum-Wheel-Right-Bearing">https://opencircuit.shop/Product/4-inch-100mm-Mecanum-Wheel-Right-Bearing</a>	17
Obrázek 2: Omni kolo <i>Zdroj:</i> <a href="https://www.vexrobotics.com/omni-wheels.html">https://www.vexrobotics.com/omni-wheels.html</a>	18
Obrázek 3: Jízda mecanumského podvozku podle směru otáčení <i>Zdroj:</i> (RoboteQ, Driving Mecanum Wheels)	19
Obrázek 4: Jízda podvozku s Omni koly podle směru otáčení <i>Zdroj:</i> <a href="https://www.servomagazine.com/magazine/article/a-look-at-holonomic-locomotion">https://www.servomagazine.com/magazine/article/a-look-at-holonomic-locomotion</a>	19
Obrázek 5: Výpočet motorů pomocí úhlových konstant a Gaussovy metody <i>Zdroj:</i> (Angel, 2014)	20
Obrázek 6: Výpočet motorů pomocí vektorů X a Y <i>Zdroj:</i> (Swart, 2014)	20
Obrázek 7: Omni kolo použité na podvozku	21
Obrázek 8: Outrunnerový motor (BLDC) <i>Zdroj:</i> <a href="https://arduino-shop.cz/arduino/1633-1000-kv-bezkartacovy-outrunnerovy-motor-a2212-13t.html">https://arduino-shop.cz/arduino/1633-1000-kv-bezkartacovy-outrunnerovy-motor-a2212-13t.html</a>	22
Obrázek 9: Použitý krokový motor Microcon SX16-0402N	22
Obrázek 10: Princip otáčení rotoru přepínáním cívek <i>Zdroj:</i> (Nyms, Krokový motor - princip)	23
Obrázek 11: Model uchytávacího špantu kol v SolidWorks	24
Obrázek 12: Vytisknutý uchytávací špunt mezi kolem a motorem	24
Obrázek 13: Špunt přichycený v kole	25
Obrázek 14: Schéma podvozku	25
Obrázek 15: Načrtlý podvozek na alubondové desce	26
Obrázek 16: Použitý model Raspberry Pi <i>Zdroj:</i> (Raspberry Pi 3 Model B+)	27
Obrázek 17: Arduino Uno Rev3 <i>Zdroj:</i> (Arduino Uno Rev3)	29
Obrázek 18: Motor driver A4988 pro krokové motory	30
Obrázek 19: Zapojení motor driveru A4988 s Arduinem a krokovým motorem	30
Obrázek 20: Výsledná konstrukce podvozku	31
Obrázek 21: Schéma I2C sběrnice při komunikaci <i>Zdroj:</i> (Tišnovský, 2009)	32

Obrázek 22: Testovací zapojení I2C sběrnice . . . . .	33
Obrázek 23: Test rychlosti počtu bajtů při posílání na I2C sběrnici v milisekundách . . . . .	34
Obrázek 24: Jednotková kružnice využitá pro směr jízdy <i>Zdroj:</i> (Matematika.hys.cz, Trigonometrie) . . . . .	40
Obrázek 25: Schéma úhlového rozvržení motorů na podvozku . . . .	41
Obrázek 26: Schéma velikostí vektorů X a Y podle směru pohybu . .	41
Obrázek 27: Úhly motoru 2 pro vysvětlení vzorce rychlosti . . . . .	42

## 1 Úvod a cíl práce

Správný návrh podvozku se skládá z mnoha složitých rozdělených částí, které je potřeba sloučit do jediného funkčního celku. Pro vytvoření této bakalářské práce bylo použito mnoho zdrojů různorodých témat. Pohybujeme se v oblastech od konfigurací systémů, přes samotný návrh implementace algoritmů, samotné programování, až po samotnou konstrukci podvozku.

V této práci jde o syntézu různorodých implementačních částí. Programují se zde dva jednodeskové počítače, které se musí navzájem chápat a musí vědět, co po sobě chtejí. Zároveň musí tyto instrukce zvládnout vykonat v reálném prostoru a čase, na správně sestavené konstrukci, která mechanický pohyb vydrží a bude zároveň ochranou pro celou tuto sestavu.

V této implementaci se využívají programovací jazyky Python a C++. Python již přichází jako předinstalovaný na Raspberry Pi a je jeho oficiálním programovacím jazykem. Arduino má své vlastní vývojové prostředí, které je napsané v jazycích C a C++. Toto vývojové prostředí podporuje programování v obou těchto jazycích a programátor v něm musí dodržovat speciální pravidla pro dodržování struktury kódu. Toto vývojové prostředí si může člověk představit jako klasické okno aplikace, do kterého píše. Oba jsou to jazyky ze skupiny vysokoúrovňových programovacích jazyků, které dokáží poskytnout kvalitní a rychlé zpracování algoritmů v krátkém čase.

Při řízení kol bude využíváno matematických vzorců, které budou určovat jejich rychlosť a směr. Protože každé kolo má jinou polohu, bude každé využívat lehce pozměněný vzorec. Takto použité vzorce budou v daný okamžik udržovat kola v pohybu synchronizovaně.

Pro dosažení cíle a správných výsledků bude zapotřebí používat v algoritmech knihovny, které nám pomohou řešit problémy různého druhu. Jedná se o funkcionality, které výrazným způsobem usnadňují práci nebo nejsou deskami v základu podporovány. Jednou z nich může být například knihovna řešící vzájemnou komunikaci jednodeskových počítačů.

Cílem práce je sestrojit podvozek, který se umí na základě přijatých dat rozhodnout, co bude jeho vykonávaná činnost.

V této práci jde o předem definované hodnoty, které jsou vyhodnocovány a posílány, to ovšem jen přispívá správnému návrhu i implementaci tohoto podvozku.

Po úspěšném dokončení této práce a správném řízení celého stroje bude podvozek připraven pro různé nástavby a rozšíření. Může být použit pro různý automatizovaný pohyb v prostoru, popřípadě může být vybaven technikou, která dokáže vyhodnocovat obrazová data a tato technika může celý tento podvozek na základě vyhodnocených dat ovládat.

## 2 Přehled literatury

Nejdůležitějším faktorem při sestavování podvozku je vždy jeho uplatnění, popřípadě v jakých prostorách se bude takový podvozek pohybovat. To ovšem není nikdy dopředu jasné.

Pokud jej víme, pomůže nám rozhodnout, kolik kol bude podvozek mít, jakého typu kola budou a jak budou na podvozku uspořádány. Při nahlédnutí do pramenů, které se problematikou konstrukce zabývají, nalezneme stránku, která popisuje typy kol a jejich uspořádání i popis způsobu, jak mohou fungovat (SUPERDROID ROBOTS, 2019). Je zde ukázáno, jak takové typy kol vypadají. Stránka jednoznačným způsobem popisuje vektorovými diagramy, jakým způsobem se podvozek pohybuje s určitým typem kol a je jisté, že podvozek bude nutno řídit pomocí určitých matematických výpočtů, které budou měnit směr otáčení kol, jejich rychlosť v závislosti na směru jízdy, a přitom každé kolo musí mít schopnost se otáčet jinou rychlosťí, také v závislosti na směru jízdy. Řeč je o dvou typech kol. Kolech "mecanumských" (Raděj, 2015) a kolech holonomických. Každá kola se používají na jiný typ podvozku. Mnohem obecnější a univerzálnější by bylo použít kola holonomická, která jsou umístěna do obvodu. Používají se pro trojúhelníkové nebo kulaté tvary.

Je důležité, jakým způsobem a jak data budou posílána a přijímána pro správné upravení konfigurace a dodatečná nastavení jednodeskových počítačů tak, aby si navzájem mohly bezchybně předávat data. Jakým způsobem bude komunikace jednodeskových počítačů probíhat popisuje Oscar Liang (Liang, 2013). Připojení přes USB kabel nebo sériové připojení nám bude zabírat naše USB vstupy na jednodeskovém počítači, které budeme zcela určitě potřebovat pro jiné účely, například kontrolovat počítač pomocí klávesnice a myši nebo jiné vstupní zařízení. Nejlepším způsobem bude použít komunikaci přes I2C sběrnici, kterou Liang popisuje. Hlavními výhodami je například možnost připojení až 128 zařízení k počítači Raspberry Pi. Připojení většího množství zařízení se v budoucnu pro podvozek velice hodí. Je zde popsán také celý proces konfigurace obou mikropočítačů a upravení nastavení na obou z nich do té fáze, kdy je propojíme dráty a budeme schopni otestovat reálnou funkčnost jejich vzájemné komunikace na jednoduchém kusu dat. Liang v této práci dokázal, že Raspberry Pi data odesílá, Arduino data zpracuje, vypíše na obrazovku a pošle zpět na Raspberry Pi, které si je vyžádalo.

Motory, které budou na podvozku použity, jsou krokové. Přesné vlastnosti motorů nalezneme na stránkách výrobce (Microcon, Dvoufázové krokové motory). Krokové motory nám mohou zaručit určitou míru přesnosti pohybu celého podvozku. Záleží samozřejmě na počtu kroků celé otáčky,

kterými krokový motor disponuje. Čím více jich je, tím přesnější náš podvozek může být. Není zcela důležité, aby se podvozek pohyboval na milimetry přesně, ale do budoucna se středně pokročilé krokové motory mohou pro automatizované procesy velmi hodit. Zapojení jednotlivých párů vodičů krokového motoru je také popsán v dokumentu výrobce (Microcon, Návod k instalaci).

Matematika používaná pro otáčení kol je složitější a je popisována výpočty pomocí matic (Angel, 2014), násobením matic a jejich úpravami, kde je použita Gaussova metoda pro inverzi matic. Zde se ve výpočtech snažíme získat výslednou hodnotu tří sil, které reprezentují rychlosť jednotlivých kol. Druhý způsob výpočtů (Swart, 2014) je výpočet vektorů X a Y pro daný směr pohybu. Tyto vektory se poté jen používají jako proměnné pro každý vzorec k motoru. Tento druhý způsob vypadá jako elegantnější způsob řešení a zároveň efektivnější, znamenající ušetření výpočetního času před samotným pohybem všech kol. Swartův všesměrový robot přijímá R/C signály ze standardního rádiového ovládacího zařízení jako jsou vysílačky pro modely atd. Arduino na podvozku přijímá signálové pulzy, které převádí na PWM signály pro motory.

Jak bude vypadat základní konstrukce podvozku, jaký bude jeho tvar a jak se celé provedení poskládá do jednoho, záleží také na prostředí, kde se bude podvozek využívat. Jeden typ je popsán v práci Návrhu modelování a řízení všesměrového mobilního robota (Raděj, 2015). Je zde podotknuto, že podvozek je nejdůležitější částí jakéhokoliv robota. Při výběru materiálu pro jeho sestavu musí být brán důraz na jeho pevnost, protože je použit jako základna na další součástky. Autor se zde rozhodl pro desku z materiálu PVC. Materiál PVC je velice dobrá volba, protože tento materiál skutečně disponuje nejen pevností, ale také nízkou hmotností. Jak sám autor dodal: „Atributy jako tvar, materiál a rozměr desky byly voleny s ohledem na co nejnižší hmotnost, aby robot mohl být co možná nejhbitější. To by mělo mít příznivý dopad na řízení robota“. Deska jako základna podvozku je zde kulatá a motory s koly jsou připevněné k desce na spodní části. Pro dráty jsou v desce vytvořené otvory, kterými se protáhnou nahoru. Autor své matematické principy testoval a vyhodnocoval v softwaru Matlab. Svůj podvozek sestrojil ze stejnosměrných motorů, které ovšem měly své chyby, například nepřesná změna orientace podvozku při změně rychlosti motorů. Proto se rozhodl v nejbližší době přidat zpětnovazební řízení pro její udržování.

Všesměrová kola těchto menších rozměrů mají průměr pět centimetrů a konkrétně v této zmíněné práci jen jednu řadu osmi válečků po obvodu kola. Tyto válečky nejsou nijak moc vystouplé, tím pádem dokáže podvozek překonávat překážky o výšce maximálně půl centimetru. Z této informace lze vyvodit, že takový podvozek není použitelný na obyčejnou cestu. Ne-směly by na ní být kamínky nebo štěrk, popadané větvě a jiné poválující

se předměty, které by znemožnily pohyb sestavy.

Všesměrový podvozek menších rozměrů by měl tedy správně nalézt své využití v omezených, uzavřených prostorách nebo prostorách, které budou pro něj speciálně upraveny a přizpůsobeny pro jeho pohyb a manévrování.

### 3 O všesměrovém podvozku

Všesměrový podvozek ve světě robotiky představuje schopnost pohybovat se jakýmkoliv směrem a nezávisle se otáčet kolem své osy. Podvozek s těmito vlastnostmi se nazývá "holonomický". Holonomické nemusí být právě vozidlo, může to být jakýkoliv objekt, který může volně vyrazit jakýmkoliv směrem bez nutnosti před tímto výkonem jakkoliv manévrovat. Také by se dalo jednoduše říci, že pokud může podvozek vyrazit kterýmkoliv z 360ti stupňů, dá se o něm říci, že je holonomický, tedy všesměrový.

Obecně není stanoveno, kolik kol musí všesměrový podvozek mít. Každé kolo na podvozku je poháněno samostatně jedním motorem. Díky možnosti pohybovat se jakýmkoliv směrem se umí v rychlém čase přesunovat přímo z jednoho místa na druhé. Nemusí se totiž otáčet. Z toho plyne, že takovýto podvozek má 3 stupně volnosti. Jezdí dopředu a dozadu, doleva a doprava a rotuje kolem své vlastní osy. Výhoda tohoto podvozku spočívá ve velice dobré flexibilitě na malém prostoru. Nepotřebuje na svoje manévrování velkou plochu.

#### 3.1 Typy všesměrových kol

Pro všesměrové podvozky se používají dva typy kol, které se liší svou konstrukcí i svým použitím. Mají také jiný princip otáčení, který definuje jejich styl jízdy.

Častěji se v robotických projektech používají kola Mecanumská, také známé jako Švédská kola nebo Ilonova kola, která jsou tak pojmenována podle svého vynálezce, Švéda Bengta Erlanda Ilona. Jsou to kolečka bez pneumatik. Namísto pneumatik mají tato kola po svém obvodu řadu pogumovaných soudečků, které jsou šikmo připojeny po celém obvodu jejich ráfku. Každý z těchto soudků má osu otáčení  $45^\circ$  k rovině kola a  $45^\circ$  k ose nápravy.



Obr. 1: Mecanumské kolo

Zdroj: <https://opencircuit.shop/Product/4-inch-100mm-Mecanum-Wheel-Right-Bearing>

Druhým typem kol, které se pro robotické projekty používají, jsou tzv. Omni kola. Omni kola jsou podobná kolům Mecanum. Mají malé disky (nazývanými válečky) po obvodu, které jsou kolmé ke směru otáčení. Účinkem je, že kolo může být poháněno plnou silou, ale bude se také velmi snadno posouvat bočně. Vizuálně tedy řečeno se může volně "klouzat" bokem. Tato kola se také často používají v holonomických pohonnéch systémech. Své jméno Omni dostala podle anglického slova Omnidirectional (všesměrový). Kolům mecanumským by se dalo říkat Omni také, protože významem splňují to stejné, ale pro jejich rozlišení jim bylo slovo Mecanum zachováno.



Obr. 2: Omni kolo

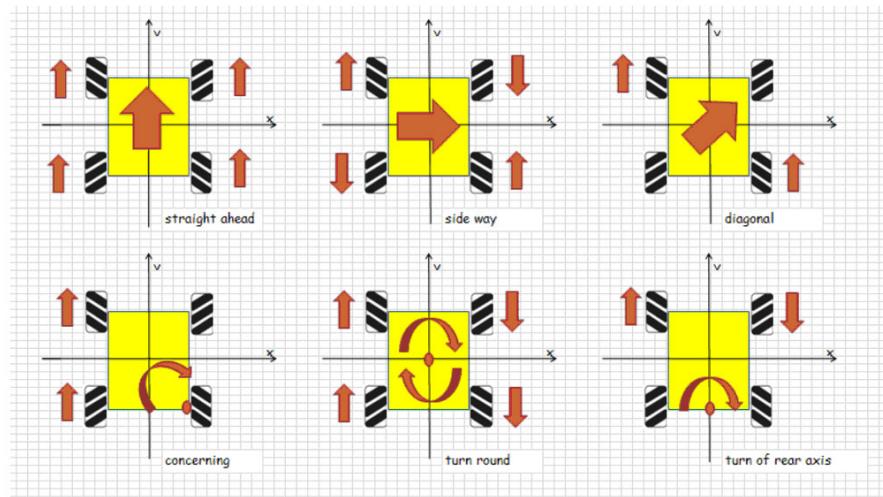
Zdroj: <https://www.vexrobotics.com/omni-wheels.html>

### 3.2 Pohybování podvozku

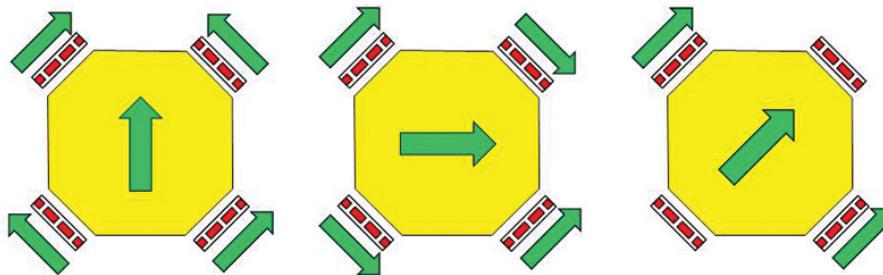
Díky té vlastnosti, že každé kolo je poháněno jednotlivými motory individuálně, s nimi můžeme otáčet různě a naprostě nezávisle. Určitými kombinacemi pohybů jednotlivých kol můžeme s podvozkem pohybovat směrem, kterým potřebujeme. Tyto kombinace pohybů se liší podle typů koleček, které na podvozku používáme a také podle jejich rozmístění na podvozku.

U mecanumských kol bývají kolečka rozmístěna po stranách za sebou, jako na klasickém automobilu. U tohoto typu kol podvozek jezdí do strany tím způsobem, že se kolečka točí proti sobě. V této chvíli se síla vzájemně působících kol vyrovnává, ovšem válečky, natočené po obvodu koleček, které jsou pootočeny pod úhlem 45 stupňů, zajišťují tento volný pohyb do strany. Pokud by jedno kolo působilo více než to druhé proti němu, podvozek by jel více po směru rychlejšího kola, tedy šikmo.

U Omni kol je logika ještě o kousek složitější. Zde se totiž pohyby jednotlivých kol sčítají, ovšem vydedukovat výsledný pohyb je mnohem těžší pro naši představivost. Je to způsobeno tím, že kola jsou poskládána do uzavřeného tvaru, jako třeba obdélník, trojúhelník atd.



Obr. 3: Jízda mecanumského podvozku podle směru otáčení  
Zdroj: (RoboteQ, Driving Mecanum Wheels)



Obr. 4: Jízda podvozku s Omni koly podle směru otáčení  
Zdroj: <https://www.servomagazine.com/magazine/article/a-look-at-holonomic-locomotion>

### 3.3 Matematické přístupy k řízení podvozku

První způsob výpočtu pohybu je vypočtení výsledných sil pomocí matice. (Angel, 2014). Tento postup předpokládá, že podvozek je rovnostranný trojúhelník, kde všechny tři kola jsou umístěna přesně uprostřed každé strany trojúhelníku. Kola jsou tedy od sebe po 120 stupních. Pro každý motor jsou nejdříve nalezeny jeho vektory x a y. Po nalezených vektorech všech motorů jsou čísla zapsána do matice o rozměrech 3x3. Převede se na inverzní matici pomocí Gaussovy metody. Nyní pokud podvozek pojede dopředu, vynásobí se  $(0, 1, 0)$  a dostaneme výslednou sílu pro každý motor, kterou poté používáme pro pohyb jednotlivých kol.

Výpočet jízdy se dá také provést vypočtením vektorů X a Y z úhlu jízdy, který známe a použijí se ve vzorcích. Pro každý motor existuje upravený

vzorec, kde se tyto vektory násobí úhlovými konstantami. Buď se navzájem sčítají, nebo odečítají. Pomocí vzorců se vypočte účinnost jednotlivých motorů, kterou poté využijeme. Vypočtené účinnosti potom můžeme upravovat podle naší libosti, abychom náš podvozek ovládali správnými hodnotami.

$$\text{We need to obtain the } x \text{ and } y \text{ components of } \vec{f}_i \\ \text{so...}$$

$$\begin{aligned} a_x &= \sin(\alpha_1 + \frac{\pi}{2}) \cdot f_1 + \sin(\alpha_2 + \frac{\pi}{2}) \cdot f_2 + \sin(\alpha_3 + \frac{\pi}{2}) \cdot f_3 \\ a_y &= \cos(\alpha_1 + \frac{\pi}{2}) \cdot f_1 + \cos(\alpha_2 + \frac{\pi}{2}) \cdot f_2 + \cos(\alpha_3 + \frac{\pi}{2}) \cdot f_3 \\ w &= f_1 + f_2 + f_3 \end{aligned}$$

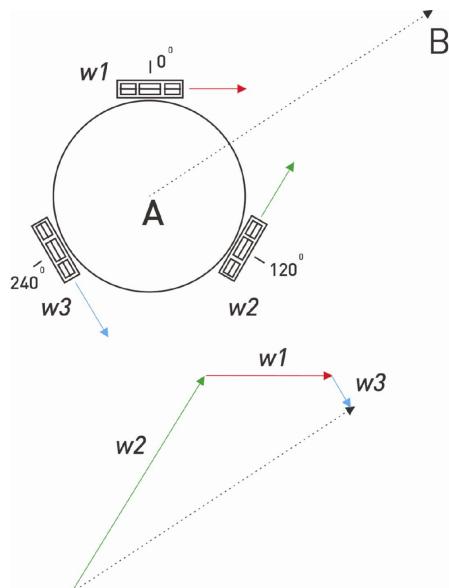
We can write this like a matrix

$$\begin{pmatrix} a_x \\ a_y \\ w \end{pmatrix} = \begin{pmatrix} \sin(\alpha_1 + \frac{\pi}{2}) & \sin(\alpha_2 + \frac{\pi}{2}) & \sin(\alpha_3 + \frac{\pi}{2}) \\ \cos(\alpha_1 + \frac{\pi}{2}) & \cos(\alpha_2 + \frac{\pi}{2}) & \cos(\alpha_3 + \frac{\pi}{2}) \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

As we know the vector  $(a_x, a_y, w)$  we need to obtain  $(f_1, f_2, f_3)$  so...

Obr. 5: Výpočet motorů pomocí úhlových konstant a Gaussovy metody

Zdroj: (Angel, 2014)



Obr. 6: Výpočet motorů pomocí vektorů X a Y

Zdroj: (Swart, 2014)

## 4 Stavba podvozku a součásti

V této kapitole je popsán celý postup skládání podvozku až do funkčního celku. Je zde uveden přehled jednotlivých součástí a zároveň některé jejich hlavní informace a vlastnosti.

### 4.1 Typ kol

Vybrání typu všesměrových kol je nejzásadnějším rozhodnutím celé práce, protože se od tohoto kroku budou vyvíjet všechny ostatní. Pro tuto práci byly vybrány klasická Omni kola. Každé toto kolo je složeno ze dvou všesměrových koleček, které jsou na sebe pevně připevněné a lehce pootočené, tak aby válečky byly rozmištěny po celém obvodu kola. Kola jsou značky Small Hammer. Vnější průměr kol je 58 milimetrů a vnitřní průměr kola pro uchycení je 13 milimetrů.



Obr. 7: Omni kolo použité na podvozku

### 4.2 Motory

Jaký typ motorů bude podvozek využívat je pro práci důležité podobně jako kola. Typů motorů je v dnešní době velké množství. První se vybírá druh motoru, tedy zda bude stejnosměrný, kartáčový, bezkartáčový (také nazýván jako BLDC nebo outrunnerový) nebo krokový. Po výběru motoru se vybírají parametry, které budou pro naši práci dostatečné. Jako uvedený příkladu není potřebné vybírat motor bezkartáčový. Tyto motory otáčejí celým svým venkovním obalem. Mají klasicky více pólu než ostatní elektromotory, což má za následek nižší rychlosť otáčení. Na druhou stranu mají zase díky tomu mnohem větší točivý moment. To je dělá skvělou volbou při použití jako pohonu na vrtule letadel nebo lodní šrouby. V našem

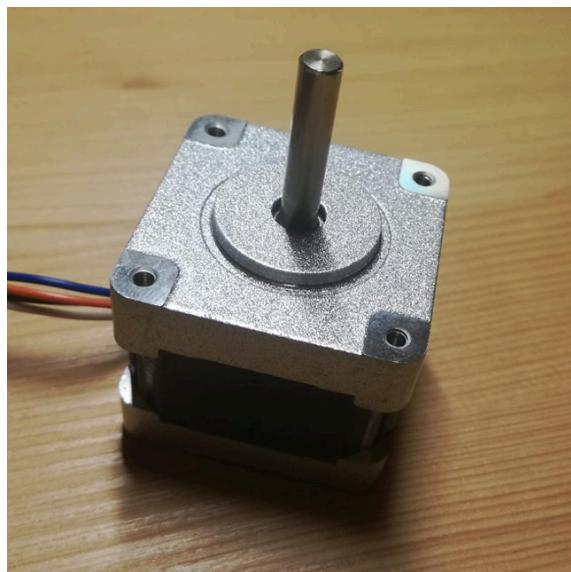
případě silný točivý moment na třech kolech lehkého podvozku zcela ne-potřebujeme, tedy přihlédneme k jinému.



Obr. 8: Outrunnerový motor (BLDC)

Zdroj: <https://arduino-shop.cz/arduino/1633-1000-kv-bezkartacovy-outrunnerovy-motor-a2212-13t.html>

Motory použité na tento podvozek jsou značky Microcon. Jsou to dvoufázové krokové motory, které mají 200 kroků. Znamená to, že motor se pro udělání celé otáčky musí pootočit dvěstěkrát. Jeden krok tohoto motoru je 1,8 stupňů s přibližnou odchylkou danou výrobcem +-0,1 stupňů. Tyto motory nejsou ani příliš velké. Délka motoru bez hřídele je 32 milimetrů, šířka 39 milimetrů a jeho hmotnost je 200 gramů (Microcon, Dvoufázové krokové motory).



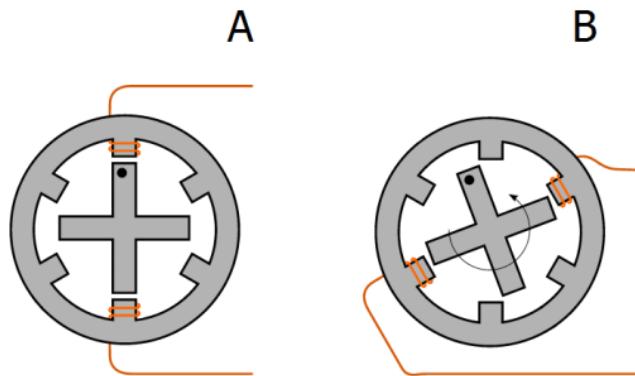
Obr. 9: Použitý krokový motor Microcon SX16-0402N

### Princip krokového motoru

Krokový motor je synchronní motor (rotor se točí stejnou rychlostí jako točivé magnetické pole ve statoru). Točivé magnetické pole ale není vytvářeno střídavým proudem, ale postupným zapínáním jednotlivých cívek statoru.

Stator motoru se skládá z několika dvojic cívek (obvykle 4 dvojice), které mohou být různě zapojeny (vyvedeny obě strany cívky, dvě a dvě cívky spojeny jednou stranou vinutí, všechny cívky se společnou jednou stranou, sériově, paralelně, ...). Rotor je váleček buď z magneticky měkkého, nebo tvrdého materiálu s vyniklými póly.

Popis funkce viz. obr. 10: Poloha A - Motor je v první poloze, protože proud tekoucí cívkami způsobuje magnetický tok, který prochází místem s nejnižším magnetickým odporem – rotem. Ostatními cívkami neprotéká žádný proud. Poloha B - Přepnutím aktivní cívky se vytvoří magnetický tok na jiném místě. Rotor se tedy natocí tak, aby kladl co nejnižší magnetický odpor, tedy o  $60^\circ$  doleva. (Nyms, Krokový motor - princip)

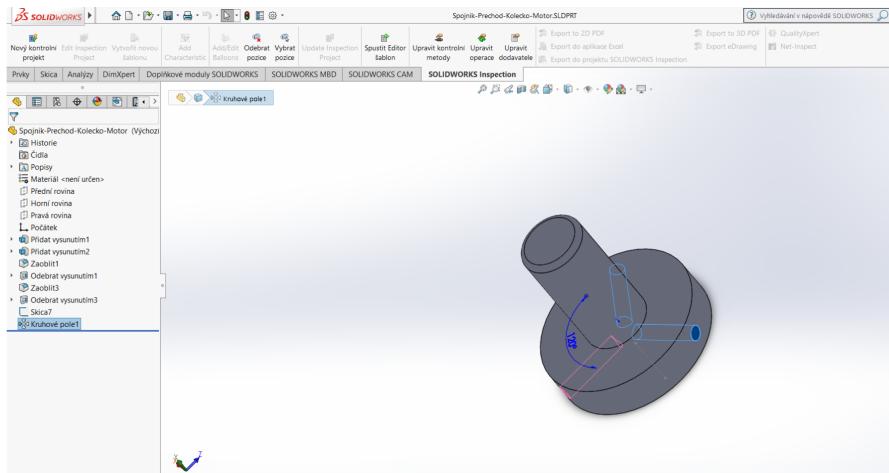


Obr. 10: Princip otáčení rotoru přepínáním cívek  
Zdroj: (Nyms, Krokový motor - princip)

### 4.3 Upevnění kol k motorům

Protože do všešměrových kol jsou předem dostupné po zakoupení pouze plastové špunty s křížovým otvorem pro lego součástky, je potřeba vymyslet jiný způsob uchycení, který bude na hřídel motoru pasovat. Systém uchycení je tedy vymodelovaný na 3D tiskárně. Pro modelování byl využitý 3D modelovací program SolidWorks. Špunty tvarem připomínají šrouby. Jejich tělo pro nasunutí do kola je namodelováno téměř o stejném průměru jako vnitřní průměr kola, aby se do kola vešly, ale za použití trochy síly. Po obvodu jejich hlavičky jsou symetricky udělány tři otvory, kterými

se zašroubují ploché šroubky k hřídeli motoru, aby se nám celá sestava vzájemně neprotácela.



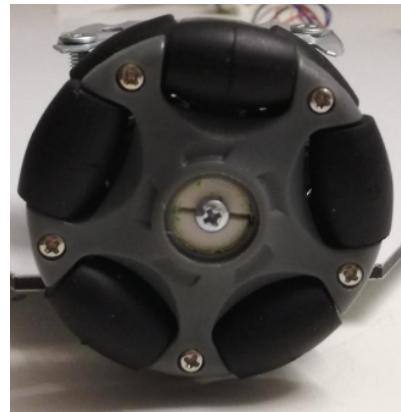
Obr. 11: Model uchytávacího špantu kol v SolidWorks



Obr. 12: Vytisknutý uchytávací špunt mezi kolem a motorem

Problém, na který se narazilo při prvním složení kolečka se špuntem byl ten, že špunty jsou sice jen o nepatrnný kousek menší v průměru, ale i bez příliš velkého vynaloženého úsilí se protáčí. Tuto komplikaci je nutno eliminovat. Protáčení koleček by způsobovalo pouze problémy, ne-přesnosti, ale především neovladatelnost podvozku jako celku.

Zepředu těla všech špuntů, tedy na "špičce", byla přímo uprostřed pilkou přičně udělána asi 5 milimetrů rýha. Kolem těla každého špantu je omotána páska. Špunt se nasadil na kolo, a poté se do špičky špantu zašrouboval samořezný vrut na plasty.

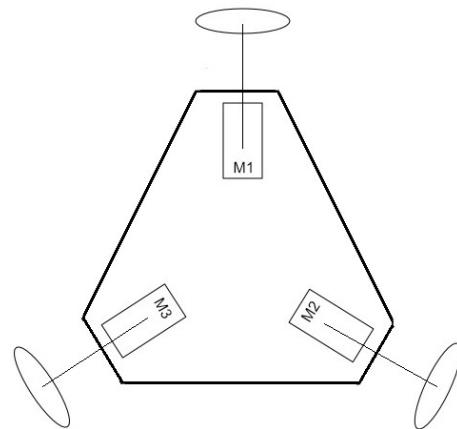


Obr. 13: Špunt přichycený v kole

#### 4.4 Návrh desky podvozku

Část navržení největší komponenty celého podvozku je velice analytická a je potřeba u ní zvážit některé faktory. Ovšem je to část také velmi kreativní, protože si s ní můžeme vyhrát dle vlastní libosti. Při prozkoumávání jsou všude kolem nejrůznější tvary a modely. Někteří dělají ikonický kruhový podvozek, jiní dělají metrové dřevěné trojúhelníkové podvozky na silniční pojezdění.

Rozhodnutí tvaru desky padlo na rovnostranný trojúhelník s ořezanými rohy pro tvarovou elegantnost a originalitu. Na tyto oříznuté rohy přijdou připevněny jednotlivé motory s koly. Materiál, ze kterého je deska vyrobena se nazývá Alubond. Je bílé barvy. Každá strana má délku 27 centimetrů a samotná výška desky je 3 milimetry. Uřezané rohy jsou dostatečně dlouhé, aby se na jejich délku vešel celý motor i s patřičnou rezervou pro případné potřeby upevnění motorů k desce.



Obr. 14: Schéma podvozku



Obr. 15: Načrtlý podvozek na alubondové desce

## 4.5 Raspberry Pi

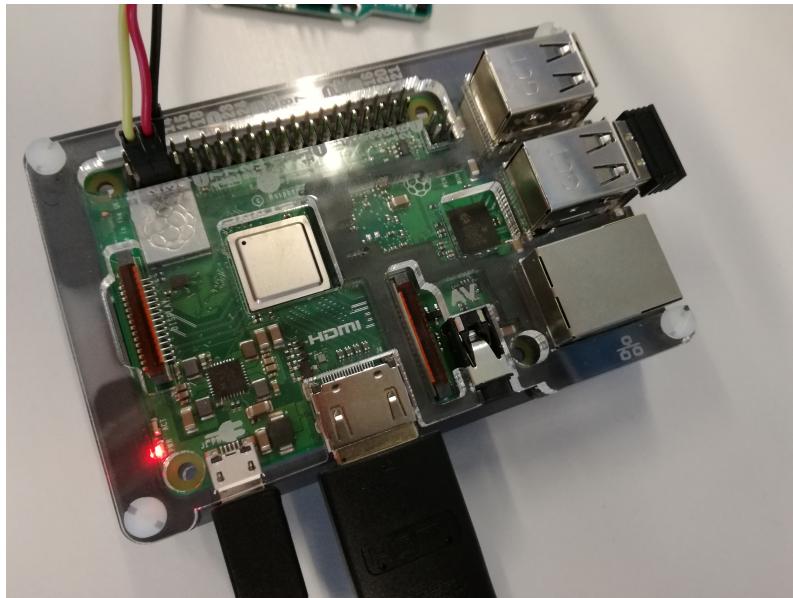
Jedná se o malý jednodeskový počítač s deskou plošných spojů o velikosti zhruba platební karty. Vyvinula jej britská nadace Raspberry Pi Foundation roku 2012 s cílem podpořit ve školství výuku informatiky a nechat obeznámit studenty s tím, jak mohou kontrolovat nejrůznější zařízení za pomocí počítače (např. mikrovlnnou troubu, automatickou pračku atd.).

Je to jednočipový počítač, který dokážeme srovnávat jako oponenta se slabším stolním počítačem. V podstatě splňuje veškeré jeho vlastnosti. Raspberry má konektor HDMI pro zapojení displeje, můžeme k němu připojit různá periferní zařízení díky jeho několika USB vstupům, jako například myš, klávesnice nebo flash disk pro uchovávání našich dat.

Pro desky se používají mikroprocesory ARM. Svými parametry se dají porovnávat s běžnými smartphony.

### Operační systémy

Můžeme na něm mít nainstalované různé operační systémy. Základním operačním systémem, který se uvnitř nachází od prvopočátku, je Raspbian. Je to linuxová distribuce Debian upravená pro chod na Raspberry Pi platformy, která má i vizuální změny. Podporuje také operační systém NetBSD, což je open source operační systém vycházející ze systému Unix. Také operační systém Windows 10 IoT Core se ujal pozice jakožto podporovaný na této platformě. Jedná se o vývojovou platformu pro Internet věcí (IoT). Tato edice operačního systému převzala jádro Windows 10. Její výhoda i nevýhoda spočívá v absenci přívětivého grafického rozhraní pro uživatele a ovládá se v systému pomocí příkazového řádku Powershell.



Obr. 16: Použitý model Raspberry Pi

Zdroj: (Raspberry Pi 3 Model B+)

### Možnosti využití

Práce v operačním systému není nijak náročná a zvládne ji i začátečník. Co se týče klasického využívání, tak Raspberry má na své desce řadu pinů (kontaktů). Jsou to vývody z desky, kterými se může k zařízením připojovat a ovládat je nebo s nimi komunikovat. V této fázi by spousta dalších desek jako Arduino svým využitím skončila. Na rozdíl od nich Raspberry můžeme využívat pro vývoj dalších aplikací, lze jej využít jako multimediální zařízení pro přehrávání hudby a videí. Má zabudovaný Ethernet port, díky němuž můžeme přistupovat k internetu. Operační systém Raspbian má v základu předinstalovanou řadu aplikací, jako je Webový prohlížeč, přehrávač hudby i programy pro programování a vytváření aplikací.

Pro svoje široké spektrum využití se dá použít pro nespočet velice zajímavých projektů. Můžeme si na něm přehrát hudbu z paměti nebo ji streamovat ze vzdáleného uložiště. Také na něm lze sledovat filmy a seriály. Můžeme si z něj udělat server ve svém domově. Buď pro vlastní účely, jako např. ukládání a zálohování, nebo na něm provozovat internetové služby či aplikace a poskytovat je veřejnému internetu. Také jej můžeme používat pro automatizaci domácnosti, jako je ovládání topení nebo třeba pro monitorování.

### Generace Raspberry Pi

Generací tohoto jednodeskového počítače bylo vyvinuto již několik. První původní Raspberry Pi vyšlo již v roce 2013 a s každou novou řadou výrobce obohacuje tyto desky o nové funkcionality a zvýšení parametrů desek. Každá řada se dělí na Model A a model B, kde model B je vylepšeným a opraveným ekvivalentem modelu A. Následují generace Raspberry Pi 2 z roku 2015, poté Raspberry Pi 3 z roku 2016, jako první osazený 64bitovým procesorem.

Dnes nejnovější generací, vydanou v červnu 2019, je Raspberry Pi 4 model B, které se prodávají nyní již oproti Raspberry Pi 3 ve třech různých variantách s různou velikostí operační paměti RAM (4GB, 2GB, 1GB). Napájení pomocí micro-USB bylo nahrazeno napájením pomocí USB-C. Jeden HDMI port byl nahrazen dvěma micro-HDMI porty a Raspberry Pi 4 nyní podporuje i monitory s rozlišením 4K. Dva USB 2.0 porty byly nahrazeny USB 3.0 porty. (Raspberry Pi)

V této práci je použito Raspberry Pi 3 model B+. Je to léta vyzkoušený produkt a za svou dobu prošel mnoha úpravami a vylepšeními směřujícími k velké stabilitě. Tento model má 1.4GHz 64bitový 4jádrový procesor ARMv8. Stejně jako Model 2 je vybaven 1 GB operační paměti RAM. Kromě 64bitového procesoru má také integrované moduly Wi-Fi a Bluetooth 4.2, 300 Mb/s Ethernet a PoE konektor.

### 4.6 Arduino

Arduino je malý jednodeskový počítač založený na mikrokontrolerech ATmega od firmy Atmel. Svým návrhem se snaží podobným způsobem jako Raspberry podpořit výuku informatiky ve školách a seznámit studenty s tím, jak jsou pomocí počítačů řízena zařízení. Nejedná se tedy o počítač ve smyslu stolního počítače nebo chytrého telefonu. Nelze proto k němu snadno přímo připojit monitor ani klávesnici či myš, ale je připraven na připojení LED diod, displeje z tekutých krystalů, servomotorů, senzorů, osvětlení atd.

První Arduino vyšlo roku 2005. Arduino ale není zamýšleno jako obdobu počítače, tak jako Raspberry. Program, který bude na Arduinu probíhat, je vyvíjen zvlášť (na PC nebo notebooku). Po napsání algoritmu je do Arduina nahrán. Tento nahraný program se potom v Arduinu spustí. Obsahuje smyčku, nekonečný cyklus, který se provádí neustále dokola. Může snímat, zjišťovat stav a na ten potom provádí reakce. Díky své nenáročnosti disponuje Arduino nízkou spotřebou. Operuje na 5V, proto je doporučeno od výrobce napětí vstupu 7-12V. Z tohoto hlediska je možné napájet Arduino malou baterií a hodí se například pro řízení dronů, robotů atd.



Obr. 17: Arduino Uno Rev3

Zdroj: (Arduino Uno Rev3)

### Typy Arduina

Dnes je typů Arduin asi jedenáct. Nejznámějšími typy Arduina jsou Arduino Nano a Arduino Uno. Arduino Nano je ze všech nejmenší, má trochu menší paměť, jiné rozhraní USB, ale jinak je práce stejná. Arduino Uno je větší, má klasické USB a velikostí jsou si velmi podobné s Raspberry Pi.

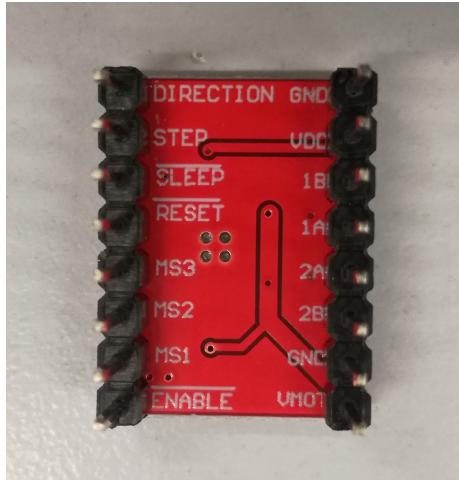
(Wikipedia, Arduino)

V této práci je použito Arduino Uno Rev3, které můžete vidět na obr. 17. Je to léta vyzkoušený produkt a za svou dobu prošel mnoha úpravami a vylepšeními.

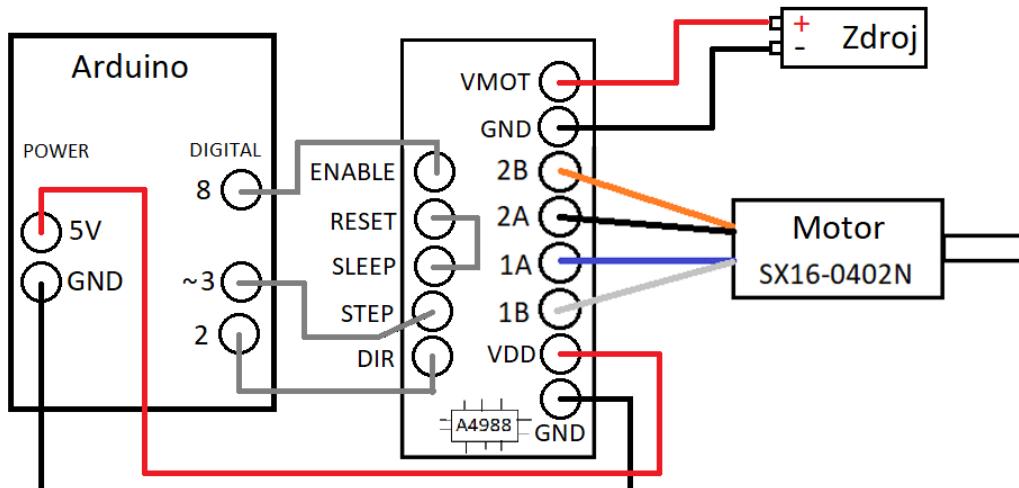
Tento model má 1.2GHz 64bitový 4jádrový procesor ARMv8. Stejně jako Model 2 je vybaven 1 GB operační paměti. Kromě 64bitového procesoru má také integrované Wi-Fi a Bluetooth moduly.

## 4.7 Propojení Arduina s motory

K propojení je použita komponenta Motor driver A4988. Tato deska od firmy Allegro pro bipolární krokové motory se vyznačuje nastavitelným omezením proudu, ochranou proti přepětí a přehráttí a pěti různými rozlišeními kroků motorů (až na 1/16 kroku). Pracuje od 8 V do 35 V a může dodávat až přibližně 1 A na fázi bez chladiče nebo umělého proudění vzduchu (je dimenzován pro 2 A na vinutí s dostatečným dodatečným chlazením). (Pololu, A4988 Stepper Motor Driver Carrier)



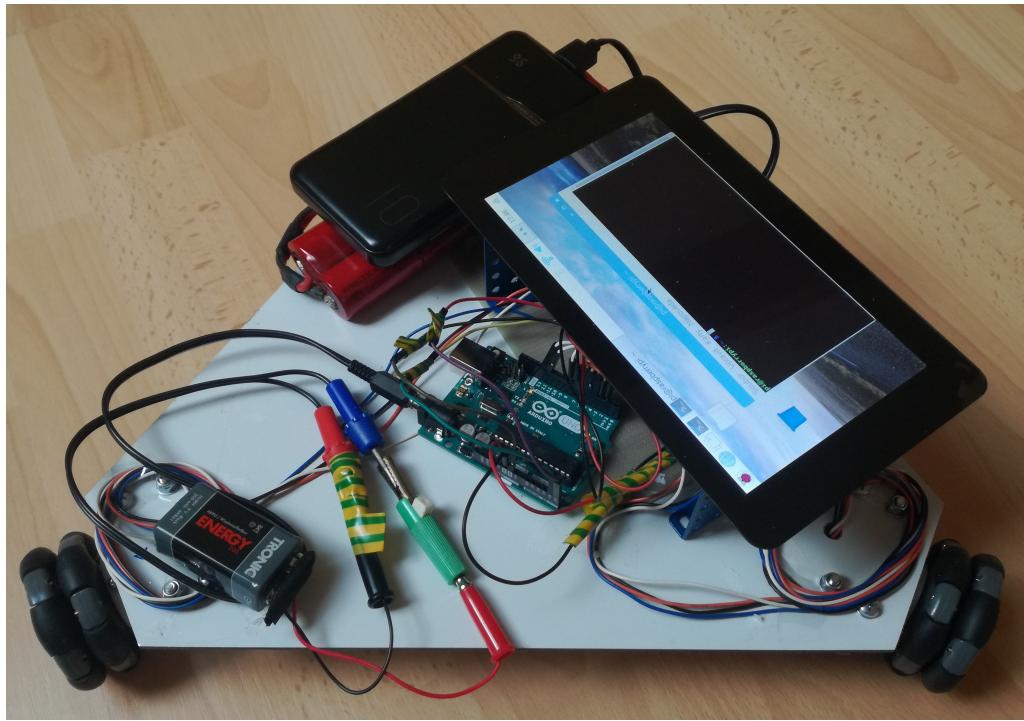
Obr. 18: Motor driver A4988 pro krokové motory



Obr. 19: Zapojení motor driveru A4988 s Arduinem a krokovým motorem

Jednotlivé páry vodičů motoru se připojí do správných kontaktů A a B na driveru. Driver je napájen zdrojem pro motory (8-35V) na pinech VMOT a GND a také Arduinem pro logické operace (3-5,5V) na pinech VDD,GND. Arduino posílá signály do pinu STEP pro krokování motoru a DIR pro ovládání směru otáčení motoru. Enable slouží pro přerušení napájení do motoru. Pin Sleep na motor driveru umožňuje šetření baterie v tzv. "režimu spánku". V tomto režimu je motor driver v podstatě nečinný. Pokud pin není zapojený, driver bude v tomto režimu neustále. Pin Reset je připojen právě do tohoto pinu pro neustálé udržování zapnutého driveru, protože šetřit baterii není potřeba.

### Výsledná konstrukce



Obr. 20: Výsledná konstrukce podvozku

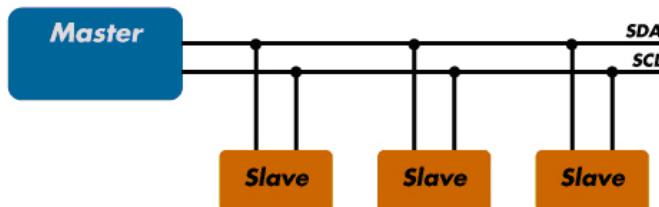
## 5 Komunikace Raspberry Pi a Arduina

Jsou k dispozici dvě možnosti, jak propojení provést. Nejjednodušším propojením Arduina a Raspberry je sériová komunikace. Druhou je připojení přes I2C sběrnici.

Při použití sériové komunikace můžeme propojit desky dvěma způsoby. Buď přímo USB kabelem z USB portu Raspberry Pi do USB portu Arduina nebo přes GPIO piny desek. Při použití USB kabelu se nám bude obtížně pracovat. Budeme muset program nahrát a poté USB kably přepojovat, což je celkem nešikovné. I2C sběrnice využívá propojování desek pomocí kabelů přes GPIO piny. Pro její nastavení je potřeba provést více práce a nastavování, ovšem po zprovoznění přináší mnohem více výhod.

### 5.1 I2C sběrnice

Pro tento podvozek byl vybrán právě tento typ komunikace. Sběrnice I2C používá společný hodinový signál (data se tedy přenáší synchronně), ovšem komunikace je poloduplexní, což znamená, že v jeden okamžik může existovat pouze jedno vysílající zařízení a libovolný počet zařízení (většinou však jen jedno), která data přijímají. Komunikace může probíhat tam i nazpět. Můžeme tedy říci, že tato sběrnice je sériová také. Sběrnice používá komunikaci typu master-slave, tedy nadřízený a podřízený. Tímto způsobem se určuje, které zařízení signály vydává, a které je přijímá, popřípadě na ně odpovídá. U sběrnice I2C také není použit výběr zařízení typu podřízený pomocí zvláštních signálů, protože každému uzlu může být přiřazena jednoznačná adresa, kterou my konkrétně definujeme přímo v napsaných programech. Kromě elektrických charakteristik je totiž ve specifikaci přesně stanoven i základní komunikační protokol. Obecně je možné říci, že I2C je sice složitější, ale zato flexibilnější sériovou externí sběrnicí, která se velmi často používá i pro komunikaci na delší vzdálosti (řádově metry).

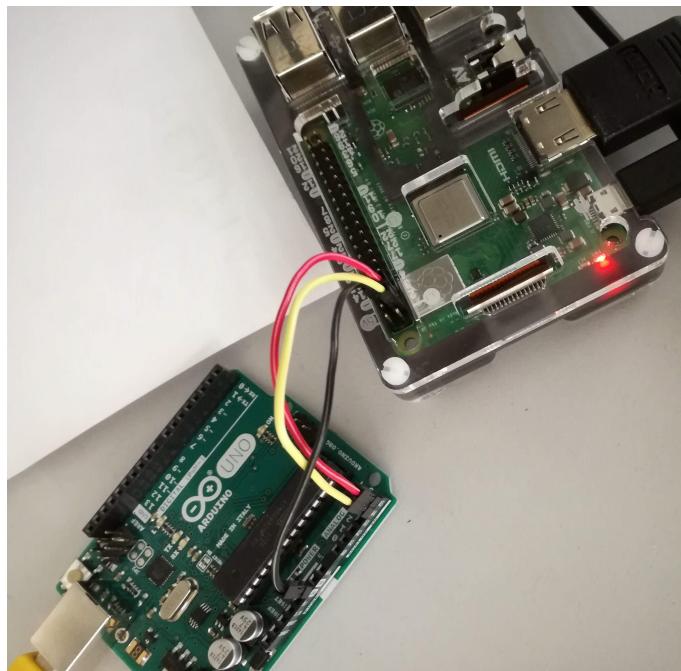


Obr. 21: Schéma I2C sběrnice při komunikaci  
Zdroj: (Tišnovský, 2009)

Na obr. 21 je zobrazen princip propojení několika uzlů pomocí sběrnice I2C. V naší implementaci Master představuje Raspberry Pi a Slave Arduino, které bude požadavky přijímat. Sběrnice je tvořena dvojicí signálových vodičů. První signálový vodič slouží pro obousměrný přenos dat (SDA – serial data), druhým vodičem pak zařízení typu master posílá všem ostatním zařízením hodinové signály (SCK – serial clock). (Tišnovský, 2009)

### Zapojení Raspberry a Arduina po I2C sběrnici

Při propojování desek je potřeba opatrnosti. Nepracují na stejném napětí a špatné zapojení vede ke zničení (nejspíše Raspberry Pi, protože pracuje na nižší hladině). Při správném zapojení by nám nemělo vadit, že zařízení nepracují na stejně hladině, protože naše zapojení je deklarováno master-slave, Arduino nemá připojené žádné pull-up rezistory, a proto pracuje i na 3,3V. Kabel z Raspberry pinu 3 (SDA) půjde do A4 (SDA) pinu na Arduinu a kabel z Raspberry pinu 5 (SCL) do A5 (SCL)(Liang, 2013). Jak již bylo řečeno, to jsou vodiče, které podle I2C standardu přenáší data a hodinové signály. Poslední důležitou částí je zapojit uzemnění mezi deskami (GND). Na obr. 22 je výsledné zapojení ukázáno.

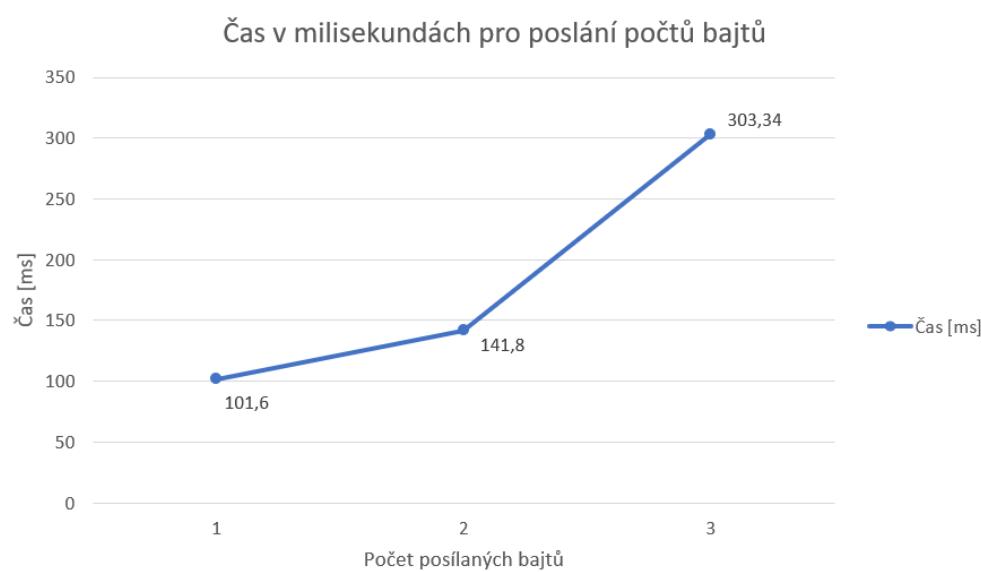


Obr. 22: Testovací zapojení I2C sběrnice

## 5.2 Způsob komunikace

Pro přesnou komunikaci a výměnu dat je nutné mít domluvený formát dat, který se bude mezi Raspberry Pi a Arduinem posílat. Pro všechny možné příkazy musí vyjít dostatek místa pro data, ale zároveň čím více je dat, tím pomalejší je přenos. Rozhodnutí tedy padlo na přenos několika bytů. Jednotlivé skupiny bitů v těchto bytech budou reprezentovat hodnotu, kterou požadujeme.

Byla provedena měření na přenos jednoho, dvou a tří bytů v určitém počtu opakování. Tyto byty obsahovaly naše požadovaná data. Posílání dat jedním bytem je málo, všechna data se nám na tak malý prostor nevejdou. Výsledek posílání dvěma byty se zvýšil časově od jednoho jen o 20 sekund, což ještě není špatné. Na 500 opakování je zpoždění 4 setiny vteřiny na každý přenos, což je čas, který je v reálném světě při komunikaci postradatelný. Výsledek posílání tří bytů po sběrnici ovšem nemile překvapil. Očekávání bylo, že čas se zvedne lineárně, tak jako mezi jedním a dvěma byty. Čas výsledku tohoto průběhu je více než dvojnásobný. Proběhly i pokusy snížit čekání na Raspberry Pi, aby se instrukce odesíaly rychleji, ale sběrnice v průběhu přenosu selhala a vyhodila chybovou hlášku, protože přenos nestíhala. Kdyby výsledek přenosu tří bytů byl lineární nebo i trošku více, rozhodnutí by bylo těžké, možná i přiklánějící se přenosu tří bytů. Do tří bytů by se nám vlezly všechny potřebné hodnoty s opravdu vysokou přesností. Bylo by místo pro možná rozšíření v budoucnu. Graf na obr. 23 s výsledky ukazuje, že přenos dvou bytů je nejfektivnější variantou. Je rychlá a pro odesílaná data dostačující.



Obr. 23: Test rychlosti počtu bajtů při posílání na I2C sběrnici v milisekundách

Ovšem s výběrem omezeného množství bytů při komunikaci přichází i řada problémů, které se musí řešit. Nevezou se do bytů všechna čísla instrukce v původních hodnotách. Byte je číslo, které může nabývat pouze hodnot od 0 do 255. Obsahuje 8 bitů. Nemůžeme tedy přenášet informaci o úhlu, který může nabývat hodnoty až do 360 stupňů jen tak v jednom bytu. Prvním rozhodnutím bylo, že se 360 stupňů konstantou upraví na hodnotu nepřesahující maximum 255, abychom ve dvou bytech ušetřili místo. Z důvodu přebytku množství informací a nedostatku místa je nutné byty zkombinovat. Konec jednoho a začátek druhého bytu bude obsahovat právě tuto upravenou hodnotu úhlu, který bude poté opět převeden na Arduinu stejným způsobem nazpět.

### 5.3 Konfigurace a nastavení přenosu I2C

Arduino žádná dodatečná nastavení nepotřebuje. Již vlastní svoji knihovnu pro komunikaci (Wire Library), kterou má již v sobě nainstalovanou. Stačí ji tedy pouze použít v kódu. Zato Raspberry Pi bez dodatečných nastavení přes I2C komunikovat nemůže. Má v původní konfiguraci spoustu věcí nedostupných nebo vypnuty. Naší implementaci předchází řada nastavení, než jej můžeme začít používat:

- Musí se I2C sběrnice smazat z blacklistu konfigurací.
- Do souboru /etc/modules se musí vepsat modul i2c-dev.
- Nainstalovat přes terminál balíček i2c-tools.
- Přidat přístup k I2C sběrnici systémovému uživateli pi.
- Pomocí linuxového instalátoru balíčků apt-get install nainstalujeme knihovnu pro jazyk Python (python-smbus). Ta nám umožní ke sběrnici v kódu na Raspberry přistupovat.

Poslední věc pro možnost používání I2C rozhraní je nyní už pouze zkontolovat, zda je jako interface (rozhraní) k používání povoleno. K tomu použijeme raspi-config. Je to konfigurační nástroj pro Raspberry Pi, který byl původně napsán Alexem Bradburym. Tento nástroj je specifický právě pro operační systémy Raspbian. (raspi-config) Po zadání příkazu raspi-config v terminálu se dostaneme do nejrůznějších nastavení Raspberry Pi. Naše hledaná záložka je Interfaces. Zde je právě možnost I2C Interface, na kterou klikneme a Raspberry se nás zeptá, zda chceme tento interface povolit. Po této krocích Raspberry zrestartujeme. (Liang, 2013)

## 5.4 Struktura přenášených bytů

Přenášíme dva byty. Tyto dva byty jsou po vytvoření jejich struktury uloženy do pole a poslány na Arduino. Tyto dva byty mají následující strukturu jednotlivých bitů: První byte:

- Bity 1-3: Kód primitiva prováděné instrukce. Podle tohoto kódu se Arduino rozhoduje, jaký typ pohybu se provádí. Např. „Jet“, „Rotovat“, „Zabrzdit“, „Vypnout motory“, „Nastavit rychlosť“.
- Bity 4-8: První část parametru úhlu prováděné instrukce. Úhel je příliš velký. Z 360 stupňů je konstantou upraven na hodnotu nepřesahující 255, která se poté na Arduinu stejnou konstantou vrátí nazpět, abychom se opět pohybovali v celých 360 stupních.

Druhý byte:

- Bity 1-3: Druhá část parametru úhlu prováděné instrukce.
- Bity 4-7: Hodnota úrovně rychlosti. Dosahuje čísel 0-16. Je to konstanta určená k násobení původních rychlostí motoru. Při maximální hodnotě se kola mohou točit nejvýše 240 kroků za sekundu.
- Bit 8: Tzv. „Bezpečnostní bit“ nebo paritní bit. Pokud je po provedení přenosu 1, data nebyla během přenosu poškozena. Pokud bude 0, data mohla být modulována nebo poškozena a příkaz se neproveze.

### Implementace přenosu na Raspberry Pi

Na Raspberry i Arduinu se musí nastavit stejná I2C adresa (v našem případě 0x04), přes kterou spolu budou komunikovat a vytvořit objekty pro komunikaci po I2C. Na Raspberry Pi nastavíme I2C adresu Arduina a nadeklarujeme objekt pro komunikaci (SMBus):

```
ARDUINO_ADDRESS = 0x04
bus = smbus.SMBus(1)
```

Nyní se na Raspberry oba byty s daty ve správném formátu uloží do pole. Do metody knihovny smbus pro I2C přenos se vkládají tři parametry. I2C adresa Arduina, offset neboli posunutí dat od začátku přenosu a samotná data (v tomto případě naše byty). Data nikam posouvat nepotřebujeme, takže prostřední parametr bude 0.

```
firstBit = '11100000'
secondBit = '00000011'

data = [firstBit, secondBit]
```

```
bus.write_block_data(ARDUINO_ADDRESS, 0, data)
```

### Implementace přenosu na Arduinu

Na Arduinu importujeme knihovnu pro komunikaci po I2C a musíme definovat, že je Slave, neboli podřízený, a k němu nastavíme stejnou adresu 0x04. Poté je v programu arduina povinná část, metoda `setup()`, kde se deklarují proměnné při samém spuštění programu. Nastavíme rychlosť sériového přenosu dat v bitech za sekundu, aktivujeme Slave adresu kvůli viditelnosti z Raspberry Pi a nastavíme výchozí metodu, která se spustí, když Arduino přijme data ze sběrnice.

```
#include <Wire.h>
#define SLAVE_ADDRESS 0x04

void setup() {
    Serial.begin(9600); // start serial for data transmission
    Wire.begin(SLAVE_ADDRESS); // initialize i2c as slave
    Wire.onReceive(receiveData); // define default callback for data
}
```

Arduino je nyní kompletně připraveno pro přijetí dat a poté jejich zpracování v námi definované funkci dle uvážení.

V této chvíli Raspberry úspěšně odesílá data, Arduino je přijímá a zpracovává.

## 5.5 Získání hodnot instrukcí z bytů na Arduinu

Naše hodnoty, čísla, přenášíme v binární (dvojkové) soustavě. Binární číslo je reprezentováno posloupností binárních hodnot, tedy jedniček a nul. Takto jsou reprezentovány všechny hodnoty ve výpočetní technice.

Bitová operace (anglicky bitwise operation) je operace, která aplikuje určitou logickou operaci ne na konkrétní bit, ale na celou skupinu bitů. Obyčejně na celé číslo, což může být byte složený z osmi bitů atd. Všechny procesory v dnešní době jsou na takové operace připraveny. Operace bitového posuvu rozhodně nejsou pomalejší, než kdybychom prováděli operaci sčítání nebo odečítání.

Dalšími bitovými operacemi, které jsou dostupné jako strojové instrukce, jsou operace bitových posuvů a rotací. Posuv o jeden bit doleva lze interpretovat jako znásobení operandu číslem dvě, posuv o jeden bit doprava jako vydelení dvěma. Posuvy o více bitů pak jako násobení nebo dělení příslušnou mocninou čísla dvě. (Bitová operace) Posuvy o jeden bit jsou mnohem rychlejší než operace násobení a dělení. Posuvy o větší

počet bitů se na některých procesorech provádějí opakovaným posuvem o jeden bit. Co se ovšem týče výpočtové náročnosti, naše číselné posuvy jsou mnohem rychlejší než transformace přes dlouhé řetězce (tedy že by čísla byla vyjádřena jako text).

## 5.6 Bitové operace použité na Arduinu

V každém bytu je potřeba zjistit pouze určité skupiny bitů. V tomto případě třeba první tři, nebo prostřední 4 v druhém bytu. Nejrychlejší, a přitom stále jednoduchá operace pro zjištění těchto pár hodnot je operace bitového posuvu.

### Bitový posuv (bitshift)

Při provádění bitového posuvu se vezme celé číslo jako operand a aplikuje se na něj operátor „`<<`“ (posuv vlevo) nebo „`>>`“ (posuv vpravo) a za něj operand, o kolik míst se všechny bity posunou. Existují však různé druhy posuvu podle toho, jakými hodnotami vyplňuje krajní pozice. Aritmetický posuv používá operátory „`<<`“ a „`>>`“. Respektuje kódování čísla, které je posouváno, a proto má i aritmetický význam - posuv o jednu pozici vlevo odpovídá násobení dvěma a posuv vpravo dělení. Opakováním aritmetickým posuvem lze tedy velmi efektivně násobit a dělit mocninami čísla 2.

Příklad: Je potřeba zjistit první 3 bity zleva prvního bytu pro zjištění našeho kódu primitiva prováděné instrukce. Byte vypadá následovně: „11100101“. Arduino tento byte reprezentuje jako číslo. Tento byte je hodnota 229, ale není potřeba ji znát, orientujeme se binárně. Potřebujeme znát 3 jedničky zleva, které v bytu jsou. Pro zjištění prvních tří bitů se provede operace „`229 >> 5`“. Celé číslo posunulo všechny své bity o 5 míst doprava. Nyní byte vypadá takto: „00000111“. Můžeme si povšimnout, že skutečně při posunu vznikají zleva doplňováním nuly. Nyní jsme získali hodnotu našeho primitiva, podle kterého se budeme rozhodovat, jaký pohyb se bude provádět. V našem projektu jsme se rozhodli, že kód „111“ (v desítkové soustavě číslo 7) bude přímá jízda. V našem kódu tedy máme na začátku výběru instrukce mezi jinými také podmíinku, zda budeme provádět právě instrukci číslo 7.

### Bitový součet

Bitový součet využívá operátor „`|`“. Také využívá dva operandy, ale tentokrát jsou jimi obyčejně naše dvě vybraná porovnávaná čísla. Bere dva bitové vzory o stejně délce a vytváří nový bitový vzor, jehož hodnota závisí

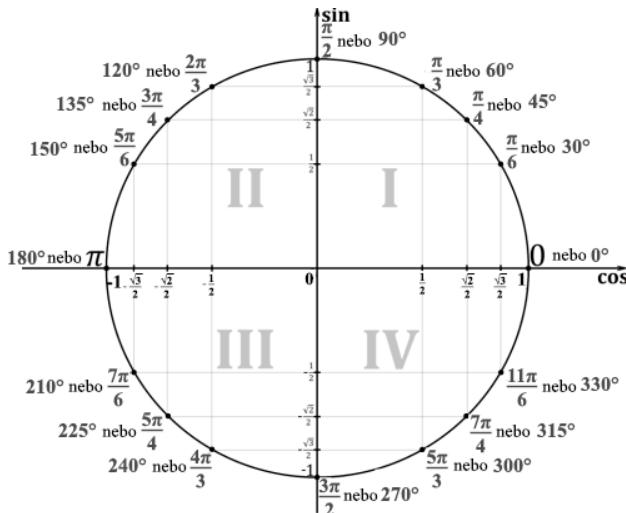
na hodnotě vstupních bitových vzorů. Porovnává postupně jeden po druhém příslušné bity (první bit prvního vzoru s prvním bitem druhého vzoru, druhý bit prvního vzoru s druhým bitem druhého vzoru atd.) a provádí s každým párem logickou operaci OR (součet). Příklad: Potřebuji spojit posledních 5 bitů prvního bytu s prvními třemi bity druhého bytu a získat tak úhel směru jízdy. Vezmeme si první byte z minulého příkladu: "11100101". Bitový posuv o tři doleva nám vrátí byte, který bude mít na prvních pěti bitech naši první část bitů, poslední tři budou vždy nuly. Tedy z "11100101" se stane "00101000" (40). Druhý byte bude například "00101101" (45). Bitový posuv o pět doprava nám vrátí byte, který bude mít na posledních třech bitech naši druhou část bitů, prvních pět bitů budou vždy nuly. Tedy "00000001" (1). Nyní proběhne součet prvního a druhého upraveného bytu a dostaneme náš správný hledaný úhel od 0 do 255. Proběhne tedy operace "40 | 1", neboli (40+1). Je to jako obyčejný součet pod sebou  $00101000 + 00000001$ . Tento součet upravené hodnoty musíme definovanou konstantou v programu vrátit zpět do maxima 360 stupňů. (HORDĚJČUK, Bitové operace)

## 6 Implementace řízení podvozku

Matematika je již prováděna na Arduinu. Raspberry se stará o předávání instrukcí a Arduino je zpracovává.

Jakým způsobem se podvozek bude chovat závisí na typu instrukce, kterou po něm budeme požadovat, tedy jaký typ instrukce od Raspberry Pi po I2C sběrnici Arduino přijme. Pro různé typy instrukcí se tedy vykonává jiná posloupnost výpočtů, podle kterých se poté motory chovají. Naši jízdu podvozku si vysvětlíme na přímé jízdě.

Po přijmutí dat k tomuto typu instrukce víme před začátkem všech výpočtů dvě věci, které Arduino obdrželo a je potřeba je ve výpočtu využít. První informace je úhel, který po přepočtení má hodnotu 0-360 stupňů. Druhá informace je úroveň rychlosti, kterou podvozek pojede. Je důležité podotknout, že naše úhly směru jízdy se počítají jako na jednotkové kružnici v goniometrii. Když jedeme doprava, máme zadáno 0 stupňů, když jedeme dopředu, máme zadáno 90 stupňů atd. Rychlosť máme uváděnou v číslech 0 až 15 a maximální rychlosť je 240 kroků za sekundu.

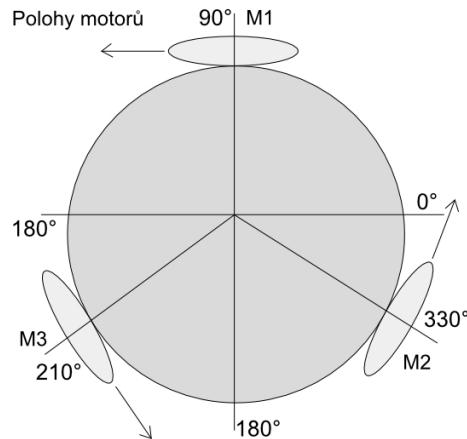


Obr. 24: Jednotková kružnice využitá pro směr jízdy

Zdroj: ([Matematika.hys.cz](http://Matematika.hys.cz), Trigonometrie)

### Výpočet přímé jízdy

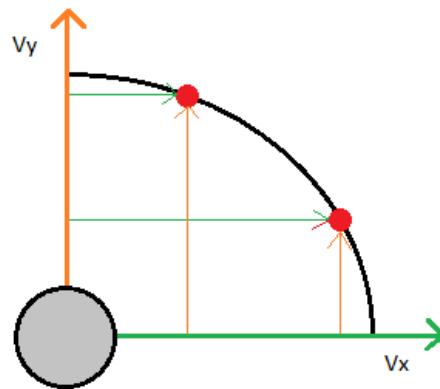
Pro představu rozmístění kol na podvozku slouží schematický obr. 25. Určili jsme si, že motor číslo 1 je vepředu v úhlulu 90 stupňů, druhý po směru hodinových ručiček na úhlulu 330 stupňů a třetí na úhlulu 210 stupňů. Motory se pohybují vpřed v protisměru hodinových ručiček.



Obr. 25: Schéma úhlového rozvržení motorů na podvozku

1. Zjistíme velikosti vektoru pohybu na ose X a ose Y pro daný úhel jízdy. Z našeho schématu na obr. 26 je tedy možné vyvodit, že pokud pojedeme doprava přímo po ose X, náš vektor X bude 1 (cosinus 0 stupňů je roven 1) a vektor Y bude 0 (sinus 0 stupňů je roven 0).

```
double vectorX = cos(uhel);
double vectorY = sin(uhel);
```



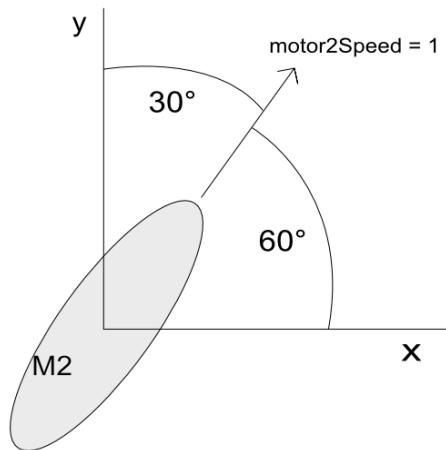
Obr. 26: Schéma velikostí vektorů X a Y podle směru pohybu

2. Zjistíme rychlosti otáčení pro jednotlivé motory. Motor1 se pohybuje pouze na ose X, tedy počítání s komponentou  $V_y$  nemá význam. Motor2 i Motor3 se pohybují v obou osách, vzorec je tedy o něco složitější. Když se na podvozek budeme dívat shora, kladné výsledné hodnoty otáčí koly proti směru hodinových ručiček, zatímco záporné po směru hodinových ručiček.

```
double motor1Speed = (-1 * vectorX) * speedModifier;
double motor2Speed = (0.5 * vectorX + sqrt(3)/2 * vectorY) *
speedModifier;
double motor3Speed = (0.5 * vectorX - sqrt(3)/2 * vectorY) *
speedModifier;
```

Motor1 je dle schématu přesně nahoře. Ten se pohybuje pouze po ose X. Při kladném vektoru X (jedeme vpravo) se vynásobí rychlosť konstantou -1 a motor se točí dozadu, při záporném vektoru X vynásobený -1 se bude motor točit vpřed. Vzorce pro motor2 a motor3 nemají náhodné konstanty pro násobení vektorů, ale mají svůj důvod. Vysvětlíme si konstrukci vzorce například na motoru 2.

Prohlédněme si obr. 27. Důležité je, že kolo kolmě ke středu podvozku nesvírá žádných 45 stupňů s osami X a Y, jak se na první pohled může zdát. Nejde zde o symetričnost.



Obr. 27: Úhly motoru 2 pro vysvětlení vzorce rychlosti

Okamžik, kdy kolo pojede skutečně plnou rychlostí (tedy 1 nebo -1), je jen jeden pod správným úhlem, který je 60 stupňů nebo jeho záporný opak  $240^\circ$ , což je přičtených 60 stupňů na opačném konci osy X podle jednotkové kružnice. VectorX se počítá cosinusem, tedy  $\cos(60^\circ) = 0,5$ . Používáme tedy k násobení proměnné vectorX tuto konstantu. To stejné provádíme pro osu Y. VectorY se počítá sinusem, tedy  $\sin(60^\circ) = \frac{\sqrt{3}}{2}$ . Zde také používáme tuto konstantu pro násobení proměnné vektoru. Při záporných hodnotách vektorů se vzorce chovají stejně. Jde o obrazně situace. Všechny hodnoty budou pouze záporné a kolo pojede opačným směrem. (Swart, 2014)

3. Nastavíme rychlosti všech motorů na Arduinu.

```
motor1.setSpeed(motor1Speed);
```

```
motor2.setSpeed(motor2Speed) ;  
motor3.setSpeed(motor3Speed) ;
```

#### 4. Roztočíme motory

```
motor1.runSpeed() ;  
motor2.runSpeed() ;  
motor3.runSpeed() ;
```

## 6.1 Ovládání motorů

Celou funkční část pohybu má na starosti Arduino. Knihovna, kterou jsou motory ovládány, se jmenuje AccelStepper library for Arduino. Poskytuje objektově orientovaný přístup pro 2, 3 nebo 4 pinové krokové motory nebo motor drivery. Dokáže ovládat více krokových motorů ve stejný okamžik s nezávislými rychlostmi. Podporuje akceleraci a zpomalování. Podporuje také velmi pomalé rychlosti a je rozšířitelná o další vlastní přidané hodnoty vývojáře. (Airspayce.com, AccelStepper library for Arduino)

U klasické Stepper knihovny pro Arduino, která byla původně použita, bylo zjištěno, že nedokáže ovládat více motorů ve stejnou chvíli. Proto bylo třeba od této knihovny upustit a nalézt jiné alternativní řešení.

Knihovna AccelStepper podporuje celou řadu funkcí pro práci s motory. Není třeba je všechny jmenovat. Některé již byly v předchozích kapitolách ukázány. Práce s prvním motorem (oba další jsou používány úplně stejně) v programu je následující:

#### 1. Vytvoření konstanty pro piny motorů ovládající směr a rychlosť

```
const int M1_DIR_PIN = 2; const int M1_STEP_PIN = 3;
```

#### 2. Vytvoření instancí motorů třídy AccelStepper s definovanými piny. Jednička na začátku v parametrech konstruktoru informuje knihovnu AccelStepper, že komunikujeme s motor driverem a ne přímo s motorem. Knihovna se bude podle toho chovat.

```
AccelStepper motor1 = AccelStepper(1, M1_STEP_PIN, M1_DIR_PIN) ;
```

#### 3. Nastavení maximální hranice kroků motoru za sekundu. Nad 240 kroků za sekundu se motor začne zadrhávat, protože naše rychlosť provedení cyklů v kódu značně převyšuje mechanické krokování motoru. Tento jev se projevuje vibrováním (nebo chvěním) motorů, kterým tato rychlosť nad limit byla nastavena. Při těchto nežádoucích vibracích se kola točí minimálním způsobem a podvozek tak manévruje chaoticky.

```
motor1.setMaxSpeed(240) ;
```

4. Nadefinování ENABLE pinu motor driveru pro každý motor. Poté se při použití metod disableOutputs() a enableOutputs() zapínají nebo vypínají motory. Při zapnutí metodou disableOutputs je motor "zablokován" (zabrzděný). Je to způsobeno tím, že je napájený, a protože se nestřídají póly, motor si drží svoji pozici.

```
motor1.setEnablePin(8)
```

5. Nastavení vypočítané rychlosti na motoru. provede se vždy jen jednou před samotným provedením jízdy.

```
motor1.setSpeed(motor1Speed);
```

6. Provedeme krok motoru. Tento příkaz se musí dít v cyklu. Metoda runSpeed() udělá pouze krok. Aby to vypadalo jako plynulé točení kolem, musíme ji volat dostatečně často a dostatečně dlouho, abychom dosáhli požadované vzdálenosti nebo časového intervalu.

```
motor1.runSpeed();
```

Díky maximálním nastaveným hranicím otáčení není nutno se bát, že by se podvozek choval nespolehlivě.

Podvozek při jízdě vydává lehce vibrační zvuk, což je způsobeno chodem krovových motorů, které jsou přišroubovány ocelovými pásky k desce podvozku. Může to být také způsobeno tím, že algoritmus nepohání krovové motory vždy nejoptimálnější frekvencí vztažené k dané rychlosti krovování, vyvolávající lehký vzdor od motorů (chtěly by se ještě točit o kousek pomaleji nebo naopak rychleji). Na odvozování takové optimální frekvence ve výpočtech není čas. Potřebovalo by spoustu reálného testování.

I při využití zaokrouhlování stupňů před přenosem a poté nazpět po něm mezi mikropočítači nevznikají odchylky. Rozdíly jsou v desetinách čísel, ale ne natolik, aby stupně přeskakovaly. Úhly jsou násobeny velice přesnou a stejnou konstantou na obou zařízeních.

## 7 Závěr

Po naprogramování matematických vzorců pro výpočet rychlostí motorů od Dirka Swarta probíhala dlouhá doba testování. Výsledné rychlosti motorů nebyly správné. Cílem bylo, aby se výsledek pohyboval v desetinné oblasti od 0 do 1 včetně pro správné vynásobení rychlostí konstantou. Ze zdroje zlomek  $\sqrt{3}/2$  není v radiánech  $60^\circ$ , ale  $70^\circ$ . Po výpočtu nám motory při maximální rychlosti překročí rychlosť až na 1,22, což není myslitelné. Může se tím překračovat maximální hranice rychlosti. Maximum musí být pro konzistenci 1. Správně jsou vzorce upraveny na  $\sqrt{3}/2$ , tedy  $60^\circ$ , jak odpovídá jednotkové kružnici. Také bylo nutné vyměnit znaménka + a - ve vzorcích motorů 2 a 3, protože výsledky měly opačnou hodnotu a motory se točily opačným směrem.

Jistý problém byl také s připojením Arduina k pinům ENABLE na motor driverech. Tento pin při průtoku proudem nebo bez něj zapíná nebo vypíná výstupy k motoru. Logické z pohledu věci je, že pokud se pin nastaví na HIGH (tedy jej sepneme), tak výstupy k motoru povolí. Po určité době řešení tohoto problému a všech vyzkoušených variantách změn v kódu bez funkčních motorů byly funkce na zapnutí a vypnutí ENABLE pinu na Arduinu vzájemně vyměněny a motor začal fungovat. Po detailním nahlédnutí do technického manuálu motor driveru na oficiálním webu je ta funkcionality skutečně uvedena. Z takových situací plyne ponaučení, že čist technické dokumentace není vždy špatný nápad.

Podvozek jezdí plynulými přechody a kola se neprotáčí. Je to způsobeno také zvolením nižší rychlosti otáček.

Tato práce mi dala možnost poznat dva nové programovací jazyky přímo v praktickém užití. Poznal jsem v široké míře skutečná využití jak Raspberry Pi, tak Arduina. Prošel jsem mnoha návrhy zapojení elektrických obvodů a diskusemi na téma, kdy to bude fungovat a kdy ne. Získal jsem informace, jaký zdroj energie na podvozek použít a jaký odběr pro motory je potřeba pokrýt. Dozvěděl jsem se, jak fungují krokové motory, jaká jsou jejich uplatnění a na jaké situace se naopak nehodí.

Podvozek je připravený na implementaci pomocí senzorů a čidel. Jeho deska je již nyní plná součástek, bylo by tedy potřeba podvozku vymyslet konstrukci pro další patro. Senzory nebo čidla by mohla snímat hodnoty okolí, jako je vzdálenost, podle které může podvozek zastavovat nebo měnit směr, aby nenarazil. Kola nejsou kompletně schována pod podvozkem, takže mohou být při nárazu zranitelná nebo se budou točit při kontaktu jinak než bychom potřebovali. Byla také myšlenka, že by se na podvozku mohl přidělat ještě jeden velice malý samostatný krokový motor, na kterém bude senzor vzdálenosti. Motor by se točil jednu otáčku doleva a potom otáčku nazpět, což může připomínat takový malý radar. Kontroloval by tak všechny směry kolem podvozku.

## 8 Literatura

SUPERDROID ROBOTS *Vectoring robots with Omni or Mecanum wheels* [online]. 2019 [cit. 2019-12-08]. Dostupné z: <https://www.superdroidrobots.com/shop/custom.aspx/vectoring-robots/44/>.

LIANG, O. *Raspberry Pi and Arduino Connected Using I2C* OscarLiang.com [online]. 2013 [cit. 2019-12-08]. Dostupné z: <https://oscarliang.com/raspberry-pi-arduino-connected-i2c/>.

FERRARI, MARIO, GIULIO FERRARI A DAVID ASTOLFO *Building robots with Lego Mindstorms NXT*. Updated ed. Oxford: Elsevier Science [distributor], c2007. ISBN 978-1-59749-152-5.

RADĚJ, JAN *Návrh, modelování a řízení všešměrového mobilního podvozku* Plzeň, 2015. Bakalářská práce. Západočeská univerzita v Plzni. Vedoucí práce Ing. Miroslav Flídr, Ph.D.

SWART, DIRK *R/C Omniwheel Robot* Make:Community [online]. 2014 [cit. 2019-12-08]. Dostupné z: <https://makezine.com/projects/make-40/kiwi/>.

ANGEL, MIGUEL *HOWTO Build your own 3-Wheels Holonomic Robot using LEGO The Technic Gear* [online]. 2014 [cit. 2019-12-08]. Dostupné z: <http://thetechnicgear.com/2014/04/howto-build-3-wheels-holonomic-robot-using-lego/>.

MICROCON *Hybridní dvoufázové krokové motory řady SX* In: Microcon [online]. [cit. 2019-12-08]. Dostupné z: [http://jan.kostial.sk/data/uploads/docs/cnc/microcon.cz\\_krokove\\_motory\\_sx\\_13-20.pdf](http://jan.kostial.sk/data/uploads/docs/cnc/microcon.cz_krokove_motory_sx_13-20.pdf).

MICROCON *Krokový motor SX16-0402N Návod k instalaci zapojení vynutí* Microcon [online]. [cit. 2019-12-08]. Dostupné z: <http://www.huco.cz/zapojenivinuti2012web/zapojenivinutipdf2012/SX16-0402N.pdf>.

ROBOEQ, OFFICE *Driving Mecanum Wheels Omnidirectional Robots* Roboteq.com [online]. [cit. 2020-03-07]. Dostupné z: <https://www.roboteq.com/index.php/applications/applications-blog/entry/driving-mecanum-wheels-omnidirectional-robots>.

*Krokový motor – princip* Mylms [online]. mylms.com, 2012 [cit. 2020-03-11]. Dostupné z: <https://www.mylms.cz/krokovy-motor-princip/>.

NĚMEC, LUKÁŠ *Optimální plánování pohybu robota se všešměrovým*

- podvozkiem* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.
- Raspberry Pi* Wikipedia the free encyclopedia [online]. 2019 [cit. 2020-03-12]. Dostupné z: [cs.wikipedia.org/wiki/Raspberry\\_Pi/](https://cs.wikipedia.org/wiki/Raspberry_Pi/).
- Raspberry Pi 3 Model B+* Mylms [online]. RPishop.cz [online]. 2018 [cit. 2020-03-13]. Dostupné z: <https://rpishop.cz/raspberry-pi-3b/896-raspberry-pi-3-model-b-plus-64-bit-1gb-ram-713179640259.html>.
- Arduino* Wikipedia the free encyclopedia [online]. 2020 [cit. 2020-03-13]. Dostupné z: [cs.wikipedia.org/wiki/Arduino](https://cs.wikipedia.org/wiki/Arduino).
- Arduino Uno Rev3* Mylms [online]. Arduino [online]. Arduino [cit. 2020-03-13]. Dostupné z: <https://store.arduino.cc/arduino-uno-rev3>.
- TIŠNOVSKÝ, PAVEL *Komunikace po sériové sběrnici I2C* Root.cz [online]. 8. 1. 2009 [cit. 2020-03-13]. Dostupné z: <https://www.root.cz/clanky/komunikace-po-seriove-sbernicii-sup2supc/>.
- raspi-config* RaspberryPi [online]. [cit. 2020-03-13]. Dostupné z: <https://www.raspberrypi.org/documentation/configuration/raspi-config.md>.
- Bitová operace* Wikipedia the free encyclopedia [online]. 2017 [cit. 2020-03-14]. Dostupné z: [https://cs.wikipedia.org/wiki/Bitov%C3%A1\\_operace](https://cs.wikipedia.org/wiki/Bitov%C3%A1_operace).
- HORDĚJČUK, VOJTĚCH *Binární čísla a bitové operace* Voho.eu [online]. [cit. 2020-03-14]. Dostupné z: Dostupné z: <http://voho.eu/wiki/bit/>.
- Jednotková kružnice* Matematika.hys.cz [online]. [cit. 2020-03-19]. Dostupné z: <http://matematika.hys.cz/trigonometrie.php>.
- A4988 Stepper Motor Driver Carrier* Pololu [online]. [cit. 2020-03-20]. Dostupné z: <https://www.pololu.com/product/1182>.
- AccelStepper library for Arduino* Airspayce.com [online]. [cit. 2020-03-21]. Dostupné z: <http://www.airspayce.com/mikem/arduino/AccelStepper/>.
- NOVÁK, PETR. *Mobilní roboty: pohony, senzory, řízení*. Praha: BEN - technická literatura. ISBN 80-7300-141-1.