FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Knihovna pro řízení šestinohého robota
Student:	Jitka Seménková
Vedoucí:	Ing. Miroslav Skrbek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Seznamte se s modelem šestinohého robota a proveďte rešerši způsobů řízení jeho pohybu. Vezměte v úvahu i chůzi v terénu s využitím dotykových senzorů na nohách. Navrhněte a implementujte programové vybavení, které umožní řídit pohyb robota v různých směrech, případně přenášení předmětů v klepetech. Na robotovi zprovozněte akcelerometr a kameru. Části, které bude možné využít i v jiných aplikacích uspořádejte do formy knihoven, které zastřešíte demo aplikací. Rozsah práce a užité programovací jazyky upřesněte po dohodě s vedoucím práce.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 10. ledna 2019

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Bakalářská práce

Knihovna pro řízení šestinohého robota

Jitka Seménková

Katedra číslicového návrhu
Vedoucí práce: Ing. Miroslav Skrbek, Ph.D.

16. května 2019

Poděkování

Děkuji svému vedoucímu práce Ing. Miroslavu Skrbkovi, Ph.D. za ochotu a trpělivost při konzultacích. Dále děkuji Bc. Petrovi Guľovi za technický servis robota a své rodině za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 16. května 2019 ……

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2019 Jitka Seménková. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Seménková, Jitka. *Knihovna pro řízení šestinohého robota.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Cílem práce je vytvořit programové vybavení pro šestinohého robota v podobě knihovny. Knihovna je určena pro model hexapodu A-Pod od firmy Lynxmotion, který je ovládán pomocí počítače Raspberry Pi a servo kontroleru SSC-32u od stejné firmy, jako je robot. Vytvořená knihovna podporuje chůzi robota po rovině i v terénu pomocí dotykových senzorů. Při plánování pohybu je použita inverzní kinematika. Výsledky této práce umožní dalším studentům na FIT ČVUT v Praze vytvářet složitější aplikace pro šestinohého robota. V příloze práce lze nalézt uživatelskou příručku a na přiloženém CD se nachází videa zachycující chůzi robota.

Klíčová slova šestinohý robot, hexapod, programové vybavení, knihovna pohybu, pohyb v terénu, dotykové senzory, klepeta, Raspberry Pi

Abstract

The aim of this thesis is to create a software library for six-legged robot. This library is created for the A-Pod hexapod by Lynxmotion controlled with Raspberry Pi and servo controller SSC-32u by Lynxmotion. This library enables the hexapod robot to walk on flat surface and over terrain as well using contact sensors. Inverse kinematic is used for planning the movement. This library can be used to create more complex applications for hexapod robot by other students of FIT CTU. Manual and CD with video of the robot walking is attached to this thesis.

Keywords six-legged robot, hexapod, software, movement library, movement over terrain, contact sensor, mandibles, Raspberry Pi

Obsah

Úvod **1**

1 Cíl práce **3**

2 Analýza **5**
2.1 Model hexapodu 5
2.2 Základní principy chůze 6
2.3 Chůze v terénu 8
2.4 Inverzní kinematika 9
2.5 Křivky pro plánování trajektorií 9
2.6 Existující projekty 11

3 Konkrétní hexapod **13**
3.1 Konstrukce 13
3.2 Elektronika 15

4 Návrh **21**
4.1 Návrh rozhraní knihovny 21
4.2 Řízení chůze 22
4.3 Odvození inverzní kinematiky 29

5 Implementace **33**
5.1 Programovací jazyk 33
5.2 Složení aplikace 33
5.3 Interface knihovny 34
5.4 Konfigurace robota 35
5.5 Průběh funkce pro chůzi 36
5.6 Chůze v terénu 36
5.7 Implementace inverzní kinematiky 37
5.8 Tvorba trajektorie těla a nohou 37

5.9 Synchronizace nohou 38
5.10 Komunikace se servo kontrolerem 38
5.11 Zprovoznění akcelerometru 39
5.12 Použití klepet 39
5.13 Demo aplikace 40

6 Testování **41**
6.1 Chůze po rovině 41
6.2 Chůze v terénu 42
6.3 Přenášení předmětů 43

Závěr **45**

Literatura **47**

A Uživatelská příručka **51**
A.1 Zapnutí robota 51
A.2 Dostupné funkce knihovny 52
A.3 Ukázka použití knihovny 54

B Zapojení akcelerometru **55**

C Seznam použitých zkratek **57**

D Obsah přiloženého CD **59**

Seznam obrázků

0.1 Šestinohý robot A-Pod od firmy Lynxmotion 1

2.1 Tvary hexapodů 6
2.2 Typy chůze 7
2.3 Typy křivek a jejich řídicí body 10

3.1 A-Pod od firmy Lynxmotion 13
3.2 Rozměry těla A-Podu 14
3.3 Rozměry nohy A-Podu 15
3.4 Schéma zapojení elektroniky 15
3.5 Umístění elektroniky 16
3.6 Indexy pinů, do kterých jsou zapojena serva a dotykové senzory . . 17
3.7 Struktura příkazu pro pohyb serva 18
3.8 Průběh PWM signálu 19
3.9 Dotykový senzor a jeho umístění 20

4.1 Schéma cyklů nohy 22
4.2 Schéma chůze po přímce 24
4.3 Schéma pro výpočet vzdálenosti a úhlu konce nohy od středu těla 25
4.4 Schéma krajní polohy nohy při otáčení na místě 26
4.5 Návrh Bézierovy křivky 27
4.6 Chůze po křivce: Vyjádření souřadnic půlkroku relativně k poloze těla po posledním půlkroku 28
4.7 Inverzní kinematika - půdorys 29
4.8 Inverzní kinematika - bokorys 30

6.1 Chůze robota po nakloněné rovině 42
6.2 Přenášení plechovky pomocí klepet 43

A.1 Popis zapojení baterie 51

B.1 Raspberry Pi - čísla pinů 56
B.2 Zapojení I^2C na RPi 56

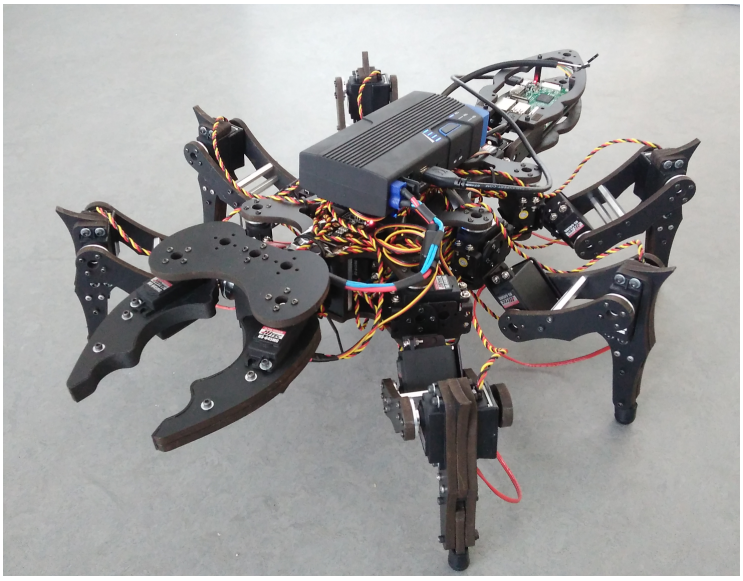
Seznam tabulek

3.1 Technické parametry Raspberry Pi 3B 16
3.2 Technické parametry Raspberry Pi kamery v1.3 19

B.1 Popis drátů pro připojení akcelerometru k RPi po rozhraní I^2C . . 55

Úvod

Robotika je v dnešní době velmi rozvíjené odvětví, které se postupně dostává do našich životů. Ať už je to v podobě domácích robotů pro úklid a vaření nebo velkých automatizovaných linek, které zastávají těžké monotonní práce. Své využití našli roboti i při plnění nebezpečných úkolů. Příkladem může být zneškodňování trhavin nebo otvírání podezřelých zavazadel.



Obrázek 0.1: Šestinohý robot A-Pod od firmy Lynxmotion

Pro práci na nepřístupných místech, jako jsou sutiny nebo cizí planety, se nejvíc hodí kráčiví roboti, kteří dokáží překonat větší překážky a nerovnosti terénu než koloví nebo pásoví roboti. Zejména vhodný je šestinohý robot – zvaný také hexapod – protože má ideální počet nohou, který při vhodné synchronizaci končetin zajišťuje dobrou stabilitu při rychlém i pomalém pohybu.

Navíc se dokáže pohybovat s jednou až dvěma vadnými končetinami, což je nejvíce oceňováno při úkolech, kdy není možné provést okamžitý servis robota. Díky svému potenciálu se šestinozí roboti těší stále většímu zájmu odborníků i veřejnosti.

Tématem mé bakalářské práce je vytvoření knihovny pohybu šestinohého robota, který se nedávno stal součástí vybavení Laboratoře inteligentních vestavných systémů na FIT ČVUT v Praze. Práce je rozdělena na teoretickou a praktickou část. Teoretická část analyzuje model hexapodu, princip jeho chůze a popisuje konkrétní hexapod, na který se práce zaměřuje. Praktická část obsahuje návrh, postup implementace knihovny a její otestování.

Výsledky této práce umožní studentům na mé fakultě vytvářet složitější aplikace pro hexapody. Přínos práce byl hlavním důvodem k volbě jejího tématu.

Kapitola 1

Cíl práce

Cílem práce je vytvořit programové vybavení pro šestinohého robota A-Pod od firmy Lynxmotion, které umožní jeho pohyb. Řešení bude zpracováno ve formě knihovny.

Cílem rešeršní částí je seznámení s modelem hexapodu a principem jeho chůze. Dále bude popsána kostra a elektronika konkrétního hexapodu, pro který je řešení vyvíjeno.

Cílem praktické části je navrhnout a implementovat knihovnu, která bude obsahovat funkce pro pohyb robota. Robot se bude umět pohybovat po rovině ve všech směrech a otáčet se na místě. Dále za pomoci dotykových senzorů na nohách a akcelerometru bude chodit v terénu. Řešení bude obsahovat část, která umožní pohyb klepet na hlavě a manipulaci s předměty jejich pomocí. Na robotovi bude zprovozněna kamera. Všechny funkce řešení budou otestovány a bude k nim vytvořena demo aplikace.

Kapitola 2

Analýza

2.1 Model hexapodu

Hexapod je šestinohý robot. Spolu s kvadrupenty a humanoidními roboty se řadí mezi kráčivé roboty. Oproti robotům na kolech mají menší potenciální rychlost na rovině, ale vynikají při pohybu v terénu, který je pro jezdící roboty špatně přístupný [1]. Další výhodou je, že méně narušují povrch, po kterém se pohybují.

2.1.1 Fyzické parametry hexapodů

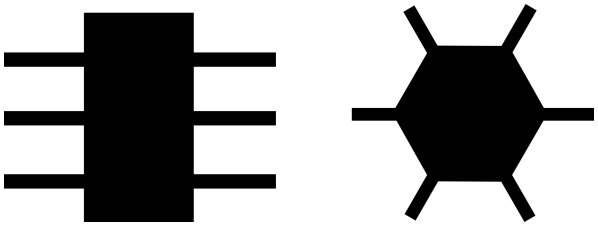
Podle tvaru se hexapody dělí na dvě základní skupiny:

1. Obdélníkové hexapody mají tělo tvaru obdélníku a na každé z delších protistran mají tři nohy. Je u nich snazší koordinace nohou při pohybu rovně, protože pohyby pro každou nohu mohou být identické. Avšak pro otáčení robota musí být implementován samostatný algoritmus. [2]

2. Hexagonální roboti mají kruhové nebo hexagonální tělo, ze kterého symetricky od středu těla paprskovitě vychází nohy. V základním postavení mají hexagonální roboti mnoho výhod. Dokážou se pohybovat do všech směrů stejným způsobem bez nutnosti otáčet se. Další velkou výhodou je větší stabilita a její snazší udržení při různých pohybech. [3]

Jiným parametrem, kterým se hexapody od sebe liší, je počet stupňů volnosti. „*Počet stupňů volnosti robota je nejmenší počet reálných parametrů potřebných k reprezentaci jeho konfigurace.*“ [4]

Každá z končetin robota je rozdělena na části, které jsou mezi sebou spojeny klouby. Existuje mnoho druhů kloubů. Lze je dělit podle druhu pohybu nebo podle počtu stupňů volnosti. Pro náš případ jsou důležitá serva, která mají rotační pohyb s jedním stupněm volnosti. Při popisu hexapodů se buď



Obrázek 2.1: Tvary hexapodů: obdélníkový (vlevo) a hexagonální

uvádí celkový součet počtu stupňů volnosti nebo je vyjádřen ve formě počtu stupňů volnosti na jednu nohu.

Aby byl umožněn pohyb nohy, musí mít alespoň jeden stupeň volnosti. V tomto případě se většinou jedná o spoj nohy a těla, který umožňuje nepřetržité otáčení o 360° [5]. Tito roboti jsou často označováni RHex.

Zároveň počet stupňů volnosti větší než šest je nadbytečný, protože s šesti stupni je možné konec nohy dát do jakékoli polohy (včetně koncového úhlu), která je v dosahu [6]. Nejčastěji se setkáme s hexapody, které mají dva nebo tři stupně volnosti na jednu nohu.

2.2 Základní principy chůze

2.2.1 Cyklus jedné nohy

Každá noha se většinou pohybuje v cyklu. Ten je možné rozdělit na dvě části - podpůrnou a přesouvací. V podpůrné fázi je noha na zemi, zajišťuje stabilitu robota a při pohybu se posouvá směrem proti pohybu. Robot se od ní odráží a tím se posouvá dopředu. Aby bylo možné provádět pohyb v podpůrné fázi, je třeba zařadit fázi přesouvací, kdy se noha zvedne do vzduchu a přesune se ve směru pohybu. Zde se položí na zem a přejde zpět do podpůrné fáze. [2]

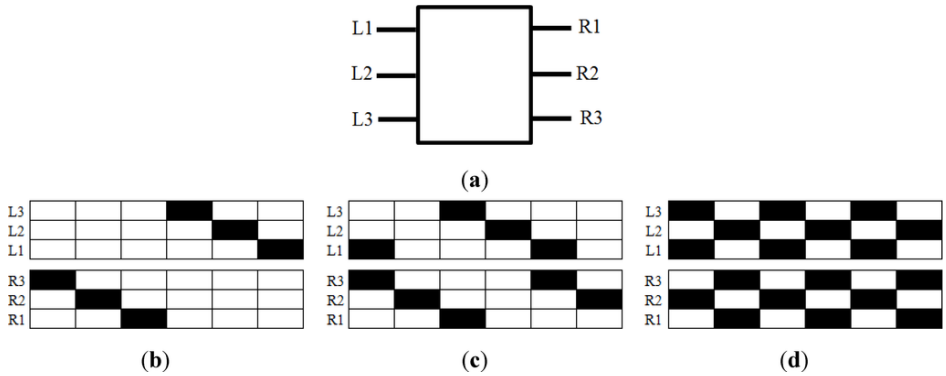
Poměr času podpůrné fáze ku celkovému času kroku nazýváme faktorem střídání (duty factor). Jeho hodnoty se pohybují od nuly do jedné. Faktor střídání omezuje maximální možnou rychlost, které je robot schopný dosáhnout. [2, 7]

2.2.2 Stabilita

Robot je staticky stabilní, jestliže se jeho těžiště nachází v mnohoúhelníku tvořeném jeho nohama, které jsou v kontaktu se zemí. Aby bylo možné dosáhnout statické stability, tak musí být na zemi alespoň tři nohy. Také platí, že čím blíže je těžiště ke středu mnohoúhelníku, tím je robot stabilnější. Hexapod využívá statické stability, protože k ní má díky šesti nohám ideální předpoklady. Hexapod může mít při chůzi snížený faktor střídání až na 0,5 a přitom má stále na zemi tři nohy, které stačí k dosažení statické stability. [2]

2.2.3 Typy chůze (synchronizace nohou)

Hexapod má díky šesti nohám velkou volnost při volbě synchronizace nohou. Jediným omezením je umístění alespoň tří nohou na zemi, aby neporušil podmínky statické stability. Následuje výčet a popis nejpoužívanějších typů chůze. Indexace nohou použitá při popisu synchronizací je znázorněna na obrázku 2.2. [5, 3]



Obrázek 2.2: Typy chůze a) indexace nohou pro účely popisu synchronizace nohou b) typ vlna c) typ vlnění d) typ tripod [5]

Tripod (3+3)

Při využití tohoto systému jsou nohy rozděleny do dvou skupin - L1L3R2 a L2R1R3. Nohy ve stejné skupině mají stejnou fázi cyklu chůze. Většinu doby je jedna skupina v podpůrné fázi a druhá v přesouvací. Když se přechází z fáze do fáze, existuje krátký okamžik, kdy se všechny nohy dotýkají země. Za jeden cyklus jsou udělány dva kroky. Faktor střídání je $\frac{1}{2}$.

Vlna (5+1)

Každá noha má jinou fázi. Vždy je ve vzduchu nejvýše jedna. Tím je docíleno maximální stability při pohybu. Tento způsob je vhodný do členitého terénu a na překonávání překážek. Je mnoho možností, jak mohou nohy následovat za sebou. Faktor střídání je $\frac{5}{6}$.

Alternující tripod

Jedná se o spojení vlny a tripodu. Rozdělení nohou je stejné jako v tripodu, avšak nohy nejsou zvedány synchronně, ale jsou fázově posunuty. Změnou délky překrytí přesouvacích fází je možné docílit větší stability na úkor maximální rychlosti nebo i opaku. Faktor střídání se pohybuje od $\frac{1}{2}$ do $\frac{5}{6}$.

4+2

Nohy jsou rozděleny do párů L1R1, L2R2, L3R3. Ve vzduchu je vždy jeden z párů. Začíná se od zadní dvojice. Faktor střídání je $\frac{2}{3}$.

Vlnění

Napodobuje chůzi čtyřnohých robotů, avšak díky většímu množství nohou je stále stabilní. Nohy jsou rozděleny do čtyř skupin - L1R3, R2, L3R1, L2. V přesouvací fázi je vždy jen jedna ze skupin a střídají se v uvedeném pořadí. V každou chvíli jsou tedy na zemi minimálně 4 nohy. Faktor střídání je $\frac{3}{4}$.

2.2.3.1 Porovnání typů chůze

Nejvyšší potenciální rychlost má tripod. Navíc je lehce implementovatelný, a proto je na rovné povrchy bez překážek jedním z nejlepších řešení. Oproti ostatním synchronizacím má však horší stabilitu, což ale vadí jen ve členitém terénu.

Další variantou je vlnění, které je o něco stabilnější, ale platí za to nižší rychlostí.

Lepším řešením může být alternující tripod, který se velice dobře dokáže interaktivně přizpůsobovat, protože kromě úpravy rychlosti kroku můžeme upravovat i délku překrytí přesouvacích fází. Proto se hodí do středně náročného terénu.

Pro velice nepravidelný terén nebo terén s velkými překážkami je dobrým řešením vlna, která má nejlepší stabilitu a její nízká rychlost v tomto prostředí není vadou.

2.3 Chůze v terénu

Chůze v terénu je mnohem náročnější než pohyb po rovině, proto se robot musí pohybovat pomaleji. Je zapotřebí, aby upravoval pohyby jednotlivých nohou podle tvaru terénu, po kterém jde. Abychom mohli úpravu provést, potřebujeme informace o okolí. Ty můžeme získat například jako předem připravenou 3D mapu [1], ale častějším způsobem je použití senzorů v průběhu chůze robota.

2.3.1 Změny vůči pohybu po rovině

Při pohybu po rovině stačí, aby se tělo posouvalo dopředu. To znamená, že nohy, které se dotýkají země, jdou dozadu. Tento způsob však není dostačující pro pohyb v terénu, protože tělo v tom případě zůstává ve stejné absolutní výšce. Což znamená, že pokud bychom robota nechali tímto způsobem jít nahoru po nakloněné rovině, tak po pár krocích narazí tělem do podložky. Tomu je potřeba předejít a pravidelně upravovat vzdálenost těla a podložky.

2.3.2 Poloha těla vůči zemi

Tělo může být nastaveno paralelně se zemí, nebo může být udržováno vodorovně, například pomocí akcelerometru. Způsob s paralelním postavením těla lze použít pouze na nakloněné rovině s malým úhlem a vyznačuje se tím, že se nohy přesouvají stejně, jako by byl hexapod na rovné zemi. Když je tělo vodorovně, zvládne udržet rovnováhu i na strmějších nepravidelných terénech. [3]

2.4 Inverzní kinematika

Inverzní kinematika je v robotice hojně využívaný matematický postup. Jedná se o opačný proces od dopředné kinematiky, který podle natočení serv jednoznačně vypočítá polohu koncového bodu. Tedy pomocí inverzní kinematiky najdeme natočení serv takové, aby byl koncový bod nohy na požadovaných souřadnicích. Inverzní kinematika v řadě případů není jednoznačná, tedy pro jedny souřadnice koncového bodu existuje více kombinací natočení serv, které je splňují. [4]

Základním přístupem inverzní kinematiky je geometrické vyjádření. Tento postup je aplikovatelný na systémy, které nemají velký počet stupňů volnosti. Při vyšším počtu začíná být příliš složitý.

Geometrický výpočet využívá především goniometrické funkce, Pythagorovu a Kosinovou větu. Odvození vzorců pro konkrétní hexapod je uvedeno v sekci 4.3.

2.5 Křivky pro plánování trajektorií

Pro plánování trajektorií robotů a jejich končetin se používají křivky, protože díky nim lze dosáhnout plynulejšího a tím i přirozenějšího pohybu. V této práci jsou využity kubické Bézierovy křivky pro plánování trajektorie těla jako celku.

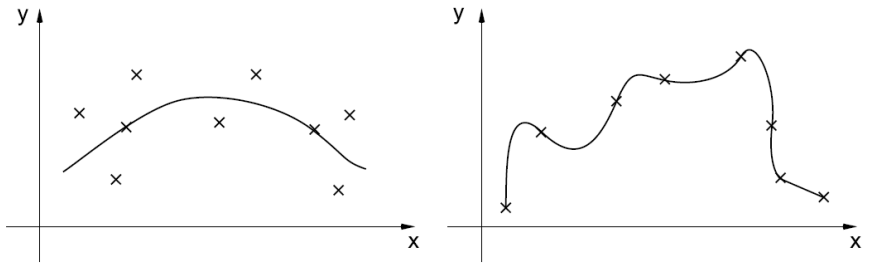
Křivku lze fyzikálně chápat jako dráhu pohybujícího se bodu, jehož souřadnice se mění v čase. V tomto případě je křivka vyjádřena parametricky v závislosti na parametru t (čase), který se většinou pohybuje v intervalu $< 0, 1 >$.

Rovnice 2.1 zobrazuje vyjádření tečného vektoru v bodě Q(t0). Vektor je určen derivací parametricky vyjádřené křivky po složkách.

$$\overrightarrow{q'}(t0) = (x'(t0), y'(t0), z'(t0)) = (\frac{dx(t0)}{dt}, \frac{dy(t0)}{dt}, \frac{dz(t0)}{dt}). \tag{2.1}$$

Při modelování křivek se využívají takzvané řídící body, kterými je ovlivňován průběh křivky. Existují dva základní typy křivek vzhledem k interpretaci řídících bodů:

- aproximační křivky jsou tvarovány na základě řídících bodů, ale nemusí jimi procházet.
- interpolační křivky přímo probíhají řídícími body,



Obrázek 2.3: Aproximační (vlevo) a interpolační křivka a jejich řídicí body [8]

Křivky můžeme na sebe postupně napojovat. Bod, ve kterém se dvě křivky spojují, nazýváme uzlem. Důležitým faktorem při spojování křivek je to, jak dobře na sebe navazují. Tuto informaci můžeme vyjádřit spojitostí.

Rovnice 2.1 až 2.4 byly převzaty z knihy [8]. Ze stejného zdroje byly čerpány i informace pro tuto kapitolu.

2.5.1 Bézierova křivka

Bézierova křivka se řadí mezi aproximační křivky, ale její zvláštností je, že prochází počátečním a koncovým řídícím bodem.

Rovnice 2.2 zobrazuje vyjádření Bézierovy křivky n-tého stupně. K určení křivky n-tého stupně je zapotřebí n+1 řídicích bodů (P_i). Pro výpočet křivky n-tého stupně je použit Bernsteinův polynom n-tého stupně (B_i^n), jehož vyjádření je v rovnici 2.3.

$$Q(t) = \sum_{i=0}^{n} P_i B_i^n(t), \tag{2.2}$$

$$B_i^n = \binom{n}{i} t^i (1-t)^{n-i}; t \in < 0, 1 >, i = 0, 1, ..., n. \tag{2.3}$$

V rovnici 2.4 je uveden maticový zápis výpočtu Bézierovy kubiky, který dále v práci využívám.

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{2.4}$$

.

2.6 Existující projekty

Existuje mnoho projektů, které se zaměřují na pohyb šestinohého robota. Každý z nich je ale uzpůsoben pro konkrétní fyzický model hexapodu, kód je implementován na danou elektroniku a většinou ho není možné použít na jiného robota. Dalším aspektem, ve kterém se implementace liší, je volba ovládání robota – zda je robot řízen pomocí ovladače či mobilu, nebo je alespoň do jisté míry autonomní. Dále jsou návrhy přizpůsobeny senzorům, které jsou na hexapodu přítomny.

Při provádění analýzy existujících řešení jsem nenalezla žádný projekt zaměřený na A-Pod s elektronikou, kterou mám k dispozici. Při odhlédnutí od elektroniky byl nejblíže projekt [9] s ovládáním ovladačem (bez zveřejněného kódu).

Článek [1] představuje robota, který chodí terénem s pomocí stereo kamery. Na základě dat z kamery hexapod upravuje výšku kroku, délku kroku a tvrdost došlapu, aby byl průchod terénem co nejhladší. Pokud se robot vyskytne v prostředí jako je vysoká tráva, kde není možné využít kameru, tak jako zpětnou vazbu z prostředí používá senzory v nohách.

Chůze hexagonálních hexapodů je hlavním tématem článku [3]. Autoři probírají různé typy chůzí, jejich parametry a použitelnost. Kromě základních typů chůzí představují i ty odolné vůči poruchám.

V projektu [10] hexapod chodí a tančí, přičemž je řízen pomocí mobilní aplikace. Hlavním parametrem funkcí pro pohyb robota je v této práci časový údaj, který specifikuje, po jak dlouhou dobu má hexapod pohyb vykonávat.

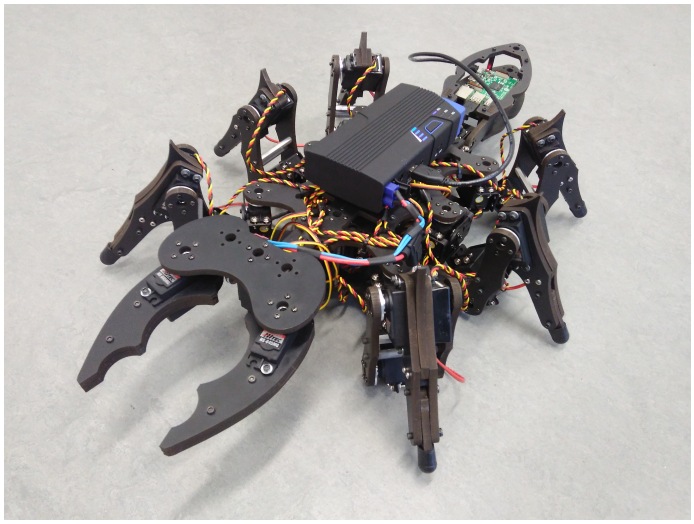
Phoenix code je označení projektu vytvářeného komunitou. [11] Projekt je velmi rozsáhlý. Původně byl napsán v programovacím jazyce Basic, ale v mezidobí byl již převeden do C/C++ na platformu Arduino. Kód v C/C++ je specificky vytvořený pro hexapody PhantomX – model MK-I a MK-II a řízení robota probíhá přes ovladač ArbotiX.

Kapitola 3

Konkrétní hexapod

V této kapitole popíši konkrétní hexapod, pro který knihovnu vyvíjím. Jedná se o model A-Pod od firmy Lynxmotion.

V první části se zaměřím na jeho kostru a mechanické vlastnosti. Ve druhé části popíši elektroniku, její parametry, možnosti a zapojení.



Obrázek 3.1: A-Pod od firmy Lynxmotion

3.1 Konstrukce

3.1.1 Tvar

Tento model nespadá ani do jedné ze základních skupin hexapodů vzhledem k tvaru těla, ale má blíže k obdélníkovým. Jeho tělo je tvaru obdélníku, jeho nohy jsou umístěné po stranách, ale přední a zadní pár musí být vytočen od

středního páru, aby bylo s nohama možné pohybovat. Toto je zohledněno už při montáži robota. Středová poloha serv zajišťujících horizontální pohyb na krajních párech nohou (serva nejblíže u těla) by měla být vytočena o 45° od středového páru.

3.1.2 Stupně volnosti

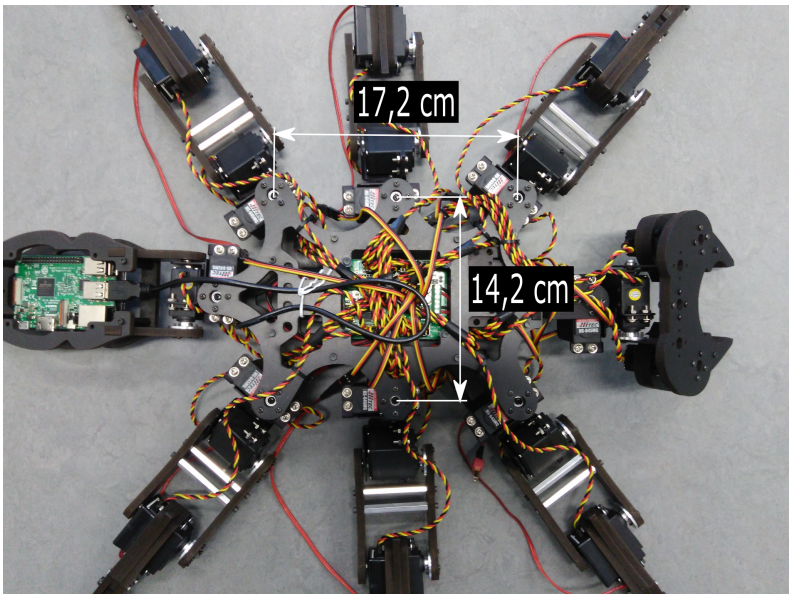
A-Pod má na každé noze tři serva a tím i stupně volnosti. Servo nejblíže u těla zajišťuje pohyb nohy horizontálně (dopředu a dozadu), zatímco další dvě serva se pohybují vertikálně (nahoru a dolů).

Na hlavě je celkem pět serv. Dvě ovládají zavírání a otevírání klepet. Zbylá zajišťují horizontální, vertikální a otočný pohyb celou hlavou.

V ocasu jsou dvě serva. Jedno se pohybuje vertikálně, druhé horizontálně.

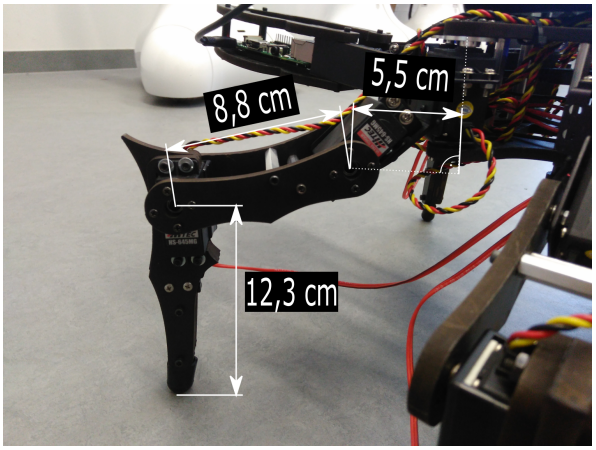
3.1.3 Rozměry

Rozměry těla potřebné pro výpočty v této práci jsou uvedeny na obrázku 3.2. Střed těla je umístěn v polovině vzdálenosti mezi středním párem nohou.



Obrázek 3.2: Rozměry těla A-Podu

Dalšími důležitými rozměry jsou délky článků nohy, které je možné vidět na obrázku 3.3.



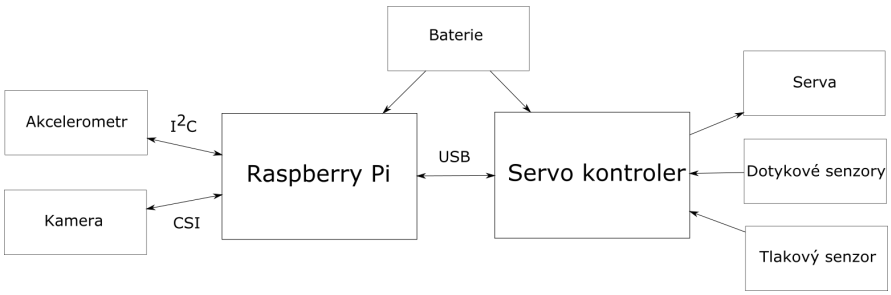
Obrázek 3.3: Rozměry nohy A-Podu

3.2 Elektronika

3.2.1 Propojení součástek

Schéma zapojení součástek je na obrázku 3.4 a její umístění na obrázku 3.5

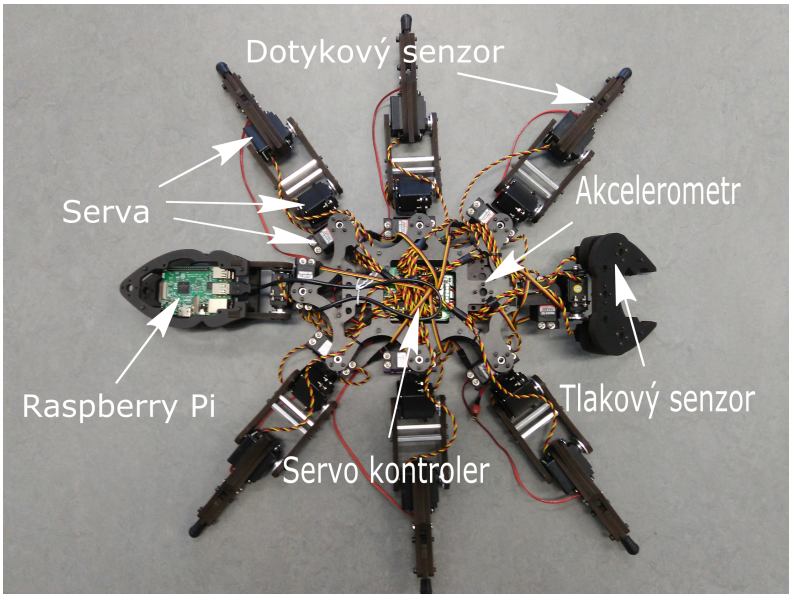
Hlavní řídicí jednotkou hexapodu je počítač Raspberry Pi umístěný v ocasu. Ten přes USB rozhraní posílá příkazy servo kontroleru SSC-32u od firmy Lynxmotion, který se stará o nastavování a udržování pozice všech serv. Obě desky jsou napájeny z baterie. Dotykové senzory na nohách a tlakový senzor v klepetech jsou zapojeny do servo kontroleru. K Raspberry Pi je připojen akcelerometr přes rozhraní I^2C a kamera přes rozhraní CSI.



Obrázek 3.4: Schéma zapojení elektroniky

3.2.2 Raspberry Pi

Hlavní řídicí jednotkou robota je Raspberry Pi (RPi) model 3B. Jedná se o malý počítač velikosti kreditní karty. Byl vyvinut společností Raspberry Pi Foundation pro výukové účely. [12] Cílem bylo zpřístupnit programování širší



Obrázek 3.5: Umístění elektroniky

skupině lidí. K tomu napomáhá nízká cena počítače a jeho univerzálnost. Jeho technické parametry jsou vypsány v tabulce 3.1.

Tabulka 3.1: Technické parametry Raspberry Pi 3B [12]

Procesor	Quad Core 1.2GHz Broadcom BCM2837 64bi
Paměť	1GB
Rozhraní	4x USB 2
	WiFi
	40 GPIO pinů - včetně I^2C
	CSI port pro kameru
	Ethernet
	HDMI

Na SD kartě je nainstalován operační systém Raspbian.

Při práci se k RPi připojujeme po WiFi vzdáleně přes protokol SSH. Bezdrátové připojení je důležité, aby se nenarušila mobilita robota.

Při řešení problémů je možné připojit k RPi monitor, klávesnici a myš a případně ethernetový kabel.

Do jednoho z USB portů je připojený servo kontroler, který se připojuje jako /dev/ttyUSB0 zařízení.

Kamera je zapojena do CSI portu, který se nachází vedle HDMI výstupu. Návod na zapojení lze nalézt na stránce [13].

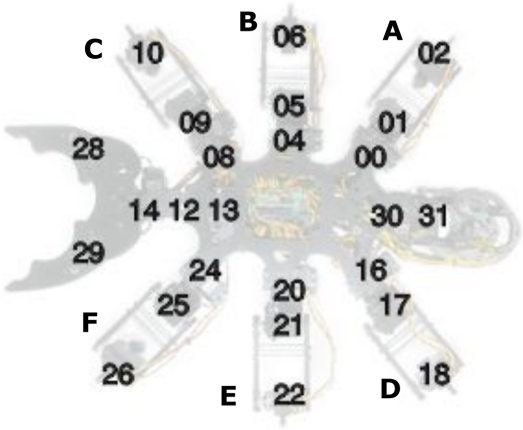
Akcelerometr je připojen přes I^2C GPIO piny.

Deska má dostatečný výpočetní výkon a paměť pro výpočty navržené v rámci práce.

3.2.3 Servo kontroler

V robotovi je použit servo kontroler SSC-32U USB Servo Controller Board od firmy Lynxmotion, jehož dokumentaci lze najít na stránce [14]. Základním úkolem kontroleru je ovládání serv pomocí PWM signálu (pulzně šířkové modulace). Dále jsou do něj zapojeny dotykové a tlakové senzory.

Deska dokáže ovládat až 32 serv současně a přijímat analogový signál z 8 zdrojů. Serva se zapojují do pinů 0-31 a číslo pinu odpovídá id serva, které je posíláno v rámci příkazů (obr. 3.6). Senzory se zapojují do pinů A-H.



Obrázek 3.6: Indexy pinů, do kterých jsou zapojená serva (čísla) a dotykové senzory (písmena) [15]

Kontroler není sám o sobě programovatelný, ale přijímá příkazy od RPi po USB rozhraní. Součástí firmwaru je i předpřipravený program umožňující chůzi hexapodu s 12 stupni volnosti (2 na nohu), takže pro A-Pod se 3 stupni volnosti není použitelný.

Nejdůležitějším příkazem kontroleru je příkaz pro změnu polohy serva. Jeho struktura je popsána na obrázku 3.7. Pro lepší přehlednost mohou být v příkazu použity mezery, tabulátory a odřádkování, protože jsou kontrolerem v tomto příkazu ignorovány.

Příkazy pro nastavování serv mohou být sdružovány. Díky tomu lze jedním příkazem nastavovat celou skupinu serv.

Pohyb lze časovat dvěma způsoby. Jednak je možné specifikovat, za jaký časový úsek má být pohyb proveden. Tato informace je sdílena mezi všechna serva při skupinovém nastavování. To znamená, že všechna serva začnou a skončí pohyb současně, přesně po určené době. Druhým způsobem je určení rychlosti změny šířky pulzu. Tento způsob ovlivňuje pouze servo bezprostředně

```
# <ch> P <pw> S <spd> T <time> <cr>
```

<ch> číslo pinu, ke kterému je servo připojeno (0 až 31), v decimální podobě

<pw> požadovaná šířka pulzu v mikrosekundách

<spd> rychlost změny šířky pulzu v mikrosekundách za sekundu

<time> čas, za který se servo dostane do požadované pozice v milisekundách

<cr> ukončovací znak příkazu v podobě carriage return

Obrázek 3.7: Struktura příkazu pro pohyb serva [14]

předcházející příkazu změny rychlosti. Oba způsoby se dají kombinovat. V tom případě se pohyb naplánuje tak, aby všechna serva začínala a končila pohyb současně. Rychlost změny šířky pulzu udává maximální rychlost pro dané servo, ale může být snížena, aby byla docílena časová synchronizace. Zadaná doba určuje minimální časový úsek, po který se serva budou pohybovat. Tato doba může být navýšena, pokud nějaké servo vyžaduje pomalou rychlost.

Při prvním poslání příkazu pohybu pro určité servo s sebou servo trhne daným směrem, protože kontroler nemá šanci zjistit jeho polohu, když je neaktivní.

Funkce Q se používá na zjištění dokončení posledního příkazu. Návratová hodnota „.“ znamená, že příkaz se již dokončil. „+“ značí, že příkaz ještě probíhá.

Funkce STOP slouží k okamžitému zastavení zadaného serva. Jedním příkazem se dá zastavit pouze jedno servo, není možné je sdružovat. Použití této funkce neovlivní návratovou hodnotu příkazu pro zjištění, jestli již bylo ukončeno provádění posledního pohybového příkazu, ani kdyby se funkcí zastavilo poslední servo.

Servo kontroler umožňuje základní kalibraci serv pomocí nastavení softwarového offsetu každému z nich. Offset se přičítá k požadované hodnotě šířky pulzu a výsledná hodnota se posílá do serva. Nastavení se provádí pomocí příkazu PO.

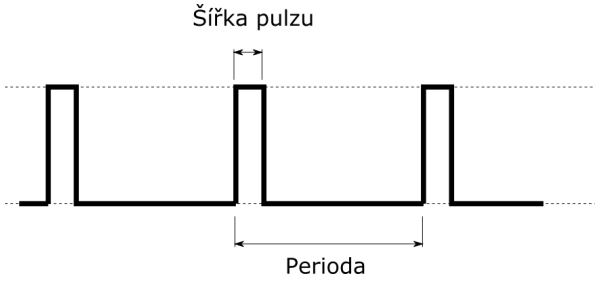
Díky příkazu QP se dá zjistit aktuální poloha serv. Funkce vrací 1 byte, který má hodnotu šířky signálu v rozlišení 10 µs. Je tedy možné získat informace pouze o aktivních servech.

Rozposlaný příkaz lze zrušit posláním <esc> znaku.

3.2.4 Serva

V robotovi jsou použita serva HS-645MG od firmy Hitec. Dokáží se otáčet v rozsahu zhruba 180°. Jedná se o analogová serva, která jsou řízena pulzně

šířkovou modulací (PWM) s periodou 20 ms (obr. 3.8). Reagují na šířku řídícího pulzu přibližně 600-2400 µs se středem 1500 µs.



Obrázek 3.8: Průběh PWM signálu

3.2.5 Raspberry Pi kamera

V hexapodu je použita oficiální RPi kamera verze 1.3. Kamera se zapojuje přímo do RPi přes CSI rozhraní.

Tabulka 3.2: Technické parametry Raspberry Pi kamery v1.3 [12]

rozlišení	5 Megapixelů
možnosti videa	1080p30, 720p60, 640×480p60/90
senzor	OmniVision OV5647
rozlišení senzoru	2592 × 1944 pixelů
formáty obrázků	JPEG (accelerated), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888
formáty videa	RAW h.264 (accelerated)

Kameru je možné používat buď přes příkazovou řádku pomocí aplikací od Raspberry Pi Foundation [16], nebo pro ni byla vytvořena knihovna pro Python, jejíž rozhraní je podrobně popsáno na stránce [13].

Pomocí kamery je možné natáčet videa i fotit v různých rozlišeních. Je možné nastavovat i pokročilé parametry jako je doba expozice.

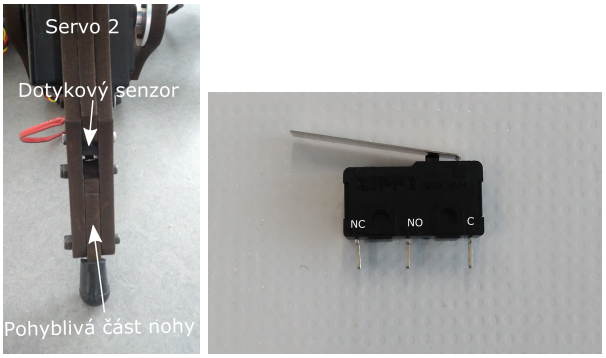
3.2.6 Akcelerometr

Do hexapodu byl v rámci práce nainstalován akcelerometr Me 3-Axis Accelerometer and Gyro Sensor od společnosti Makeblock. Základem desky je čip MPU-6050, který obsahuje akcelerometr, gyroskop a teploměr. Čip komunikuje přes sběrnici I^2C a má přiřazenu defaultní adresu 0x68. Desku je možné připojit přes piny nebo konektor RJ25 (popis zapojení je v příloze B). Požadované vstupní napětí je 5 V.

Pro komunikaci mezi RPi a čipem MPU-6050 existuje pro Python modul mpu6050-raspberrypi [17]. Jeho rozhraní umožňuje číst samostatně data z akcelerometru, gyroskopu i teploměru.

3.2.7 Dotyková čidla

Jako dotyková čidla jsou použity mechanické mikrospínače od firmy ZIPPY (obr. 3.9 vpravo). Na každé noze je umístěn jeden. Nachází se nad pohyblivou částí nohy (obr. 3.9 vlevo). Senzor se sepne, když noha stojí na zemi. Čidla jsou zapojena do analogových vstupů servo kontroleru s označením A-F.



Obrázek 3.9: Dotykový senzor (vpravo) a jeho umístění

3.2.8 Tlakový senzor

V klepetech je nainstalován Force-Sensing Resistor: 0.2 -Diameter Circle od firmy Interlink Electronics. Základem senzoru je proměnlivý odpor, jehož hodnota se snižuje při zvyšující se aplikované síle. Senzor je zapojen do analogového vstupu servo kontroleru s označením G.

3.2.9 Napájení

V robotovi je využita baterie Multifunction JUMP STARTER od firmy Whitenergy. Kapacita baterie je 13 600 mAh. Baterie má několik výstupů.

Napájení RPi je zajištěno přes USB výstup s 5 V.

Napájení logiky servo kontroleru i serv je vedeno z výstupu EC5. Baterie musí být nastavena na 12 V, které se pomocí měniče upraví na 6 V. Napájení je vedeno přes pojistku.

Kapitola 4

Návrh

V této kapitole popíši návrh programového vybavení pro pohyb hexapodu, jehož vytvoření bylo jedním z cílů práce. Při analýze existujících řešení jsem nenalezla takové, které by se dalo využít. Proto jsem navrhla vlastní. Obrázky v této kapitole jsou popsány anglicky kvůli lepšímu propojení na kód.

4.1 Návrh rozhraní knihovny

Při návrhu rozhraní (interface) knihovny jsem zohlednila, že robot bude řízen programem, nikoli ovladačem.

Hlavní funkcí knihovny je umožnit přesun z místa na místo na rovině i v terénu. Při tom se hexapod díky statické stabilitě může pohybovat variabilně rychle. Zároveň díky třem stupňům volnosti může hexapod chodit s různě zvednutým tělem při zachování základního postavení nohou.

Z těchto důvodů je rozhraní tvořeno následujícími funkcemi.

Funkce pro pohyb je rozdělena na dva druhy kvůli přehlednosti. Jedna pro chůzi po rovině a druhá pro terén. Funkce berou jako parametr vzdálenost ve složkách x a y, úhel, o který má být robot na konci pohybu otočen, a parametr upravující rychlost pohybu.

Pro začátek a konec programu a úsporu serv je přidána funkce, která hexapod převede do pozice v leže. Konce nohou se nedotýkají země a robot leží na nejnižší části těla – na pouzdrech prostředních serv nohy.

Funkce pro postavení robota zajistí přechod robota do výchozí polohy, kdy má všechny nohy na zemi a jeho tělo je v předem definované výšce. Výšku je možné změnit další funkcí.

Dále knihovna obsahuje funkci pro vycentrování hlavy i ocasu a pro ovládání klepet.

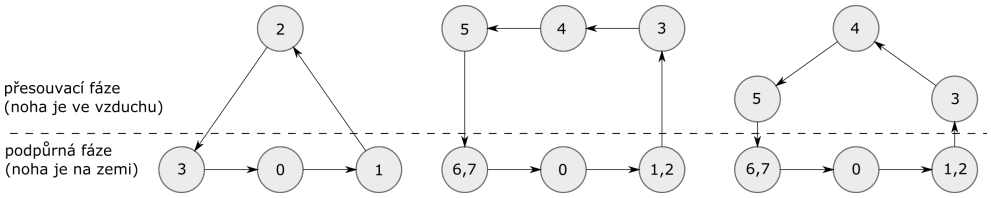
4.2 Řízení chůze

Knihovna zařizuje pohyb robota z místa na místo. Při volání funkce pro pohyb uživatel zadává parametry vzdálenosti a úhlu, podle kterých knihovna vybírá typ pohybu. Následně navrhuje dráhu, po které robot půjde, a vypočítává, jak se musí pohybovat nohy, aby se robot pohyboval po zvolené dráze. Vypočítané pozice nohou jsou knihovnou přiřazeny k jednotlivým fázím cyklu nohou, kterými se postupně prochází. V každé fázi cyklu jsou posílány příslušné příkazy pro serva, dokud robot nedorazí na místo.

4.2.1 Dráha nohou a jejich synchronizace

Zvolila jsem typ chůze tripod (viz kapitola 2.2.3), protože je díky své univerzálnosti základní dovedností hexapodů. Jeho nedostatek v podobě menší stability je pro základní chůzi v nepříliš členitém terénu zanedbatelný.

Každá noha se pohybuje v cyklu rozděleném do několika částí (stavů) (obr. 4.1). Stavy ve spodní linii obrázku jsou součástí podpůrné fáze cyklu nohy (noha je na zemi a její pohyby hýbou tělem), zbylé patří do přesouvací fáze (noha je ve vzduchu a přesouvá se na příhodné místo, pro začátek dalšího kroku). Základem každého cyklu je stav 0, kdy se noha nachází ve své výchozí pozici, jejíž x a y souřadnice jsou pro ni během programu neměnné. Stav 0 je středem podpůrné fáze dané nohy.



Obrázek 4.1: Cyklus nohy, základní (vlevo), terénní (uprostřed), vylepšený (vpravo)

Na obrázku 4.1 vlevo je zobrazen základní cyklus nohy pro chůzi po rovině. Základem je stav 0. Stavy 1 a 3 umožňují robotovi pohyb. Stav 2 umožňuje přesun vzduchem. Bohužel kvůli pohyblivému konci nohy, který se stlačuje zhruba o 3 mm, se noha začala mezi stavy 2 a 3 třít o zem a zastavila se dřív, než došla na určené místo, což způsobovalo nepřesnost chůze. Proto byl cyklus nahrazen.

Cyklus nohy při chůzi v terénu je zobrazen na obrázku 4.1 uprostřed. Oproti základní chůzi přibyly další dvě části cyklu, které zajišťují kolmost pohybu směrem dolů, takže ať je noha zastavena kdykoli během této části, nezmění se délka kroku.

Vylepšený cyklus pro chůzi po rovině, který je možné vidět na obrázku 4.1 vpravo, vznikl upravením terénního cyklu a nahradil základní variantu. Stavy 3 a 5 jsou v něm sníženy, protože stačí, aby se noha pohybovala kolmo k zemi jen těsně nad ní a ne při celém pohybu směrem dolů.

Kvůli typu chůze tripod, kdy mají nohy v různých skupinách fázový posun v cyklu o $\frac{1}{2}$, jsou do cyklů přidány nadbytečné stavy, aby se mohly inkrementovat fáze všech nohou najednou. Tedy například u terénního cyklu jsou skupiny nohou v jednu chvíli ve fázích 0 a 4, 1 a 5, 2 a 6, ..., 7 a 3. Všechny nohy se z pozice do pozice přesouvají pomocí skupinového příkazu pro serva (natočení serv je vypočítáno pomocí inverzní kinematiky v kapitole 4.3) za jednotný čas.

4.2.2 Trajektorie

Při pohybu robota z místa na místo je důležité určit dráhu, po které robot půjde. Následně je potřeba určit pohyby nohou v podpůrné fázi, aby se robot pohyboval po zvolené dráze.

Hexapod bude umět tři základní druhy pohybu:

- chůze po přímce ve všech směrech,
- otočení se na místě,
- chůze po křivce.

V každém z případů se tělo pohybuje jiným způsobem, což je nutné zohlednit při tvorbě trajektorie nohou, a je tedy třeba přistupovat k nim samostatně. V následující části práce rozeberu podrobněji všechny tři případy.

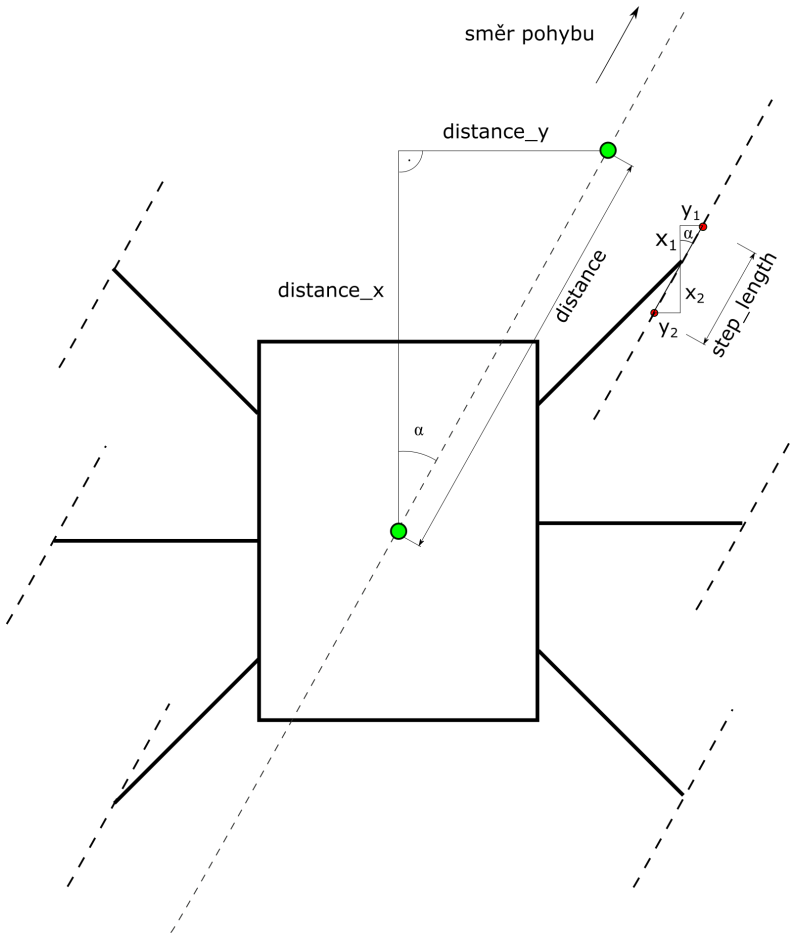
4.2.2.1 Chůze po přímce

Při chůzi po přímce se robot pohybuje přímo k cíli, aniž by se v průběhu otáčel. Aby se robot pohyboval daným způsobem, musí se jeho nohy v podpůrné fázi také pohybovat po přímkách rovnoběžných s trajektorií těla. Zároveň musí být délka kroku stejná pro všechny nohy, které jsou současně v podpůrné fázi.

Přímka, po které robot jde, je určena přeponou trojúhelníku (distance), jehož odvěsny jsou tvořeny vzdáleností ve směru x (distance_x) a y (distance_y) (obrázek 4.2). Přeponou je určen směr i vzdálenost, kterou má robot ujít.

Při určování délky kroku (step_length) jsem zvolila variantu, že všechny kroky jsou v rámci jednoho zavolání funkce stejné. Druhou variantou by byla fixní délka kroku při každém volání a jen poslední krok by byl upraven, aby součet délek kroků odpovídal požadované délce přesunu. Při výpočtu délky kroku je nejdříve určen počet kroků pomocí vydělení délky dráhy robota délkou ideálního kroku, která byla stanovena experimentálně. Délka jednoho kroku je vypočtena vydělením délky dráhy robota počtem kroků.

Polohy nohou v jednotlivých fázích kroku jsou určeny z délky kroku a úhlu (α), který svírá přepona směr chůze s osou x.



Obrázek 4.2: Schéma chůze po přímce

4.2.2.2 Otočení na místě

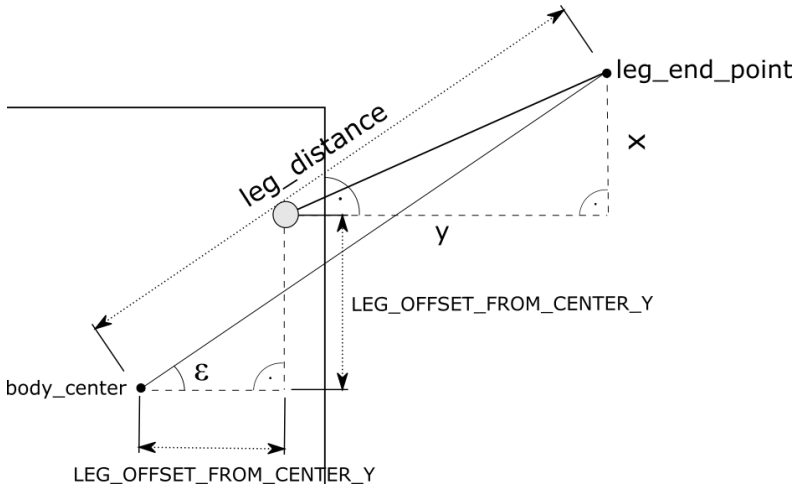
Pro docílení otočení robota na místě je nutné, aby se nohy pohybovaly po kružnici se středem ve středu těla. Nohy, které jsou současně v podpůrné fázi, se musí pohybovat se stejnou úhlovou rychlostí. Při zadání kladného úhlu se robot otáčí proti směru hodinových ručiček.

Základem je vypočítat vzdálenost (leg_distance) koncových bodů nohou (leg_end_point) od středu těla (body_center) v základní pozici a úhel (ϵ) tohoto spojení od kolmice na tělo. K výpočtu je třeba vzdálenost serva 0 od středu těla ve složkách x (LEG_OFFSET_FROM_CENTER_X) a y (LEG_OFFSET_FROM_CENTER_Y). K ní je po složkách přičtena sou-

řadnice koncového bodu nohy (vzdálenost konce nohy a serva 0 – x, y) a následně použita Pythagorova věta (obrázek 4.3).

Velikost úhlu, o který se robot otočí za jeden krok, je vypočtena obdobně jako délka kroku při chůzi po přímce.

Pro nalezení trajektorií nohou je aplikováno následující: noha se musí pohybovat po kružnici kolem středu těla, takže vzdálenost koncového bodu a středu těla je konstantní. Nohy mají mít stejnou úhlovou rychlost, proto se během jednoho kroku musí otočit o stejný úhel. Pozice nohou jsou vypočteny tak, že k úhlu základní pozice je přičtena/odečtena polovinu úhlu kroku (σ). Jsou vypočítány složky x a y vzdálenosti koncového bodu v krajních pozicích (leg_end_point_new) a odečtena po složkách vzdálenost základního postavení. Tím je určena vzdálenost krajních poloh od základního postavení nohy (x_2, y_2) (obrázek 4.4).

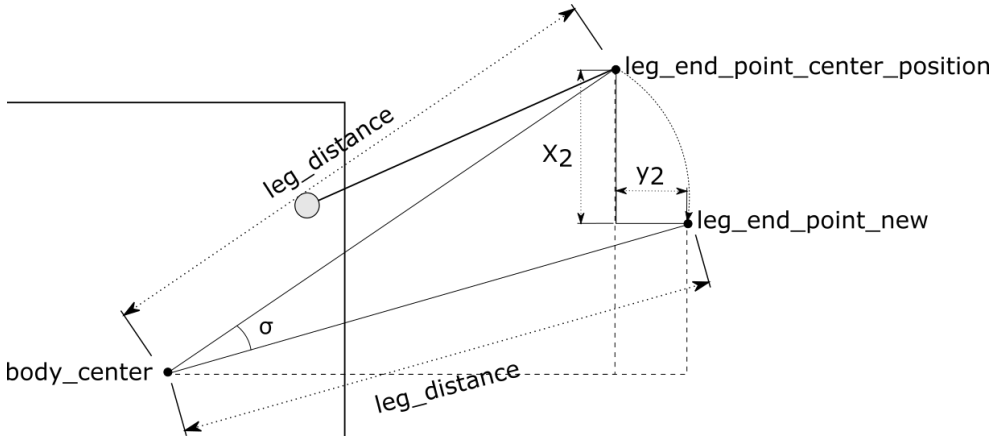


Obrázek 4.3: Schéma pro výpočet vzdálenosti a úhlu konce nohy od středu těla

4.2.2.3 Chůze po křivce

Chůzi po křivce lze využít při přesouvání robota mezi dvěma body, kdy ve výsledném bodě má být natočen jinak než v původním. Tento případ lze vyřešit i přesunem robota po přímce a následném otočení na místě, ale pohyb po křivce je ve většině případů přirozenější.

Vhodnou křivkou, po které se může robot pohybovat, je Bézierova křivka. Přestože patří mezi aproximační křivky, je u ní zaručeno, že prochází počátečním a koncovým řídícím bodem. Zároveň díky řídícím bodům sousedícím s koncovými body lze specifikovat natočení v koncových bodech. Vzhledem k těmto vlastnostem je možné navazovat více křivek na sebe. Pro základní



Obrázek 4.4: Schéma krajní polohy nohy při otáčení na místě

užití stačí kubická Bézierova křivka, která ale při určitých parametrech nezajišťuje přirozený pohyb hexapodu.

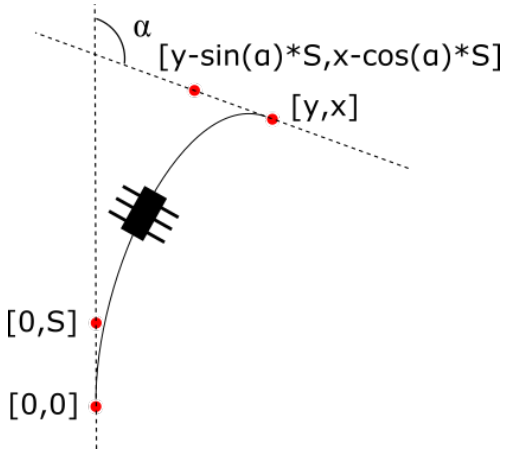
Koncové body křivky jsou určeny aktuální polohou robota, která je brána jako bod [0, 0], a zadanou vzdáleností určující bod [y, x]. Natočení robota v koncovém bodě je dáno zadaným úhlem. Ve směru natočení robota jsou ve vzdálenosti S (experimentálně zvolená hodnota, aby se robot ke směru přimykal dostatečně dlouho) od koncových bodů vytvořeny zbývající dva řídící body. Tím jsou zadány všechny čtyři body kubiky a lze jí pomocí změny parametru t (čas) procházet (obr. 4.5). V $t = 0$ se robot nachází v počátečním bodě, v $t = 1$ se robot nachází v koncovém bodě. V rovnici 4.1 je vidět dosazení řídících bodů do maticového vyjádření Bézierovy křivky.

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & S \\ y - \sin(\alpha) * S & x - \cos(\alpha) * S \\ y & x \end{bmatrix} \tag{4.1}$$

Zderivováním rovnic podle t a jeho následným dosazením je získáno natočení robota v podobě tečného vektoru v čase t.

Rovnice 4.2 a 4.3 převzaté z [18] ukazují použití transformační matice, která bod [x, y] přesune ve směru [h, k] a následně otočí o úhel α.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & h * \cos(\alpha) - k * \sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) & h * \cos(\alpha) + k * \sin(\alpha) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.2}$$



Obrázek 4.5: Návrh Bézierovy křivky

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & -h \\ -\sin(\alpha) & \cos(\alpha) & -k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{4.3}$$

Pro tvorbu trajektorie nohou navrhuji přes křivku procházet po půlkrocích robota. Parametr t je volen tak, aby rozdíl mezi současnou a následující pozicí byl přiměřený jednomu půlkroku (jeho délkou a otočením robota).

Pro výpočet polohy nohou v každém půlkroku je třeba nejdřív vypočítat délku půlkroku těla po složkách (s_y, s_x)a úhel (s_α), o který má být robot otočen. Obě informace jsou vyjádřeny relativně k poloze, ve které se nachází po posledním půlkroku (obr. 4.6). Výpočet je uveden v rovnicích 4.4 a 4.5

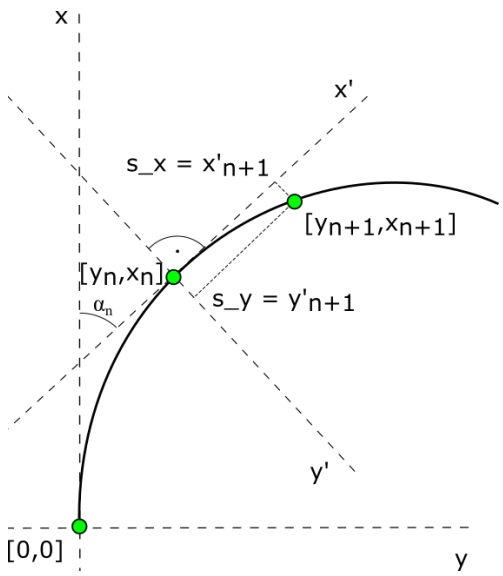
$$\begin{bmatrix} s_y \\ s_x \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(-\alpha_n) & -\sin(-\alpha_n) & -y_n * \cos(-\alpha_n) + x_n * \sin(-\alpha_n) \\ \sin(-\alpha_n) & \cos(-\alpha_n) & -y_n * \cos(-\alpha_n) - x_n * \sin(-\alpha_n) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{n+1} \\ x_{n+1} \\ 1 \end{bmatrix} \tag{4.4}$$

$$s_\alpha = \alpha_{n+1} - \alpha_n \tag{4.5}$$

Výpočet posunutí nohou při půlkroku se dělí na 2 varianty:

1. Noha je ve středové poloze [y_c, x_c] podpůrné fáze a přesouvá se do zadní polohy [y_2, x_2]. Tedy souřadnice středové polohy je potřeba transformovat způsobem uvedeným v rovnici 4.6.

2. Noha je v přední poloze [y_1, x_1] a přesouvá se do středové polohy [y_c, x_c] podpůrné fáze. Tedy souřadnice středové polohy jsou transformované a původní souřadnice jsou získány pomocí výpočtu v rovnici 4.7.



Obrázek 4.6: Chůze po křivce: Vyjádření souřadnic půlkroku relativně k poloze těla po posledním půlkroku

V případě výpočtu souřadnic levých nohou se před výpočtem musí u y souřadnice změnit znaménko, aby byly tyto souřadnice na správné straně osy (pro všechny nohy jsou za normálních okolností y souřadnice kladné). Po výpočtu se zase převedou zpět.

$$\begin{bmatrix} y_2 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(-s_\alpha) & -\sin(-s_\alpha) & -s_y * \cos(-s_\alpha) + s_x * \sin(-s_\alpha) \\ \sin(-s_\alpha) & \cos(-s_\alpha) & -s_y * \cos(-s_\alpha) - s_x * \sin(-s_\alpha) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_c \\ x_c \\ 1 \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} y_1 \\ x_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(-s_\alpha) & \sin(-s_\alpha) & s_y \\ -\sin(-s_\alpha) & \cos(-s_\alpha) & s_x \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_c \\ x_c \\ 1 \end{bmatrix} \tag{4.7}$$

V obou případech se od výsledných souřadnic odečte původní souřadnice středové polohy, aby se získal jejich offset.

4.2.3 Chůze v terénu

Při chůzi v terénu je třeba přizpůsobit cyklus nohy prostředí. K tomu slouží zpětná vazba z dotykových senzorů na nohách, které se sepnou, když je noha na zemi. Informace z nich používám při pokládání nohy na zem, kdy kontroluji stav senzoru. Když sepne, nohu zastavím. Pro zastavení serva existují dvě možnosti. Buď použít funkci Stop, kterou servo kontroler má. Nebo je možné zjistit od servo kontroleru aktuální polohu serv a na ně serva nastavit. Více

se mi osvědčila druhá varianta, protože při použití funkce Stop sebou serva trhaly nazpátek, že se noha ani nedotýkala podložky.

Ve chvíli, kdy se zastaví všechny nohy v pohybu dolů, se vypočtou nové offsety výšky těla pro tyto nohy. Výšky se využijí ve zbytku aktuálního průběhu cyklu nohou, aby bylo stále počítáno s tím, že noha má být zvednuta více než obvykle, aby správně přiléhala k zemi.

Následně proběhne upravení výšek všech nohou. Úprava se skládá ze dvou částí:

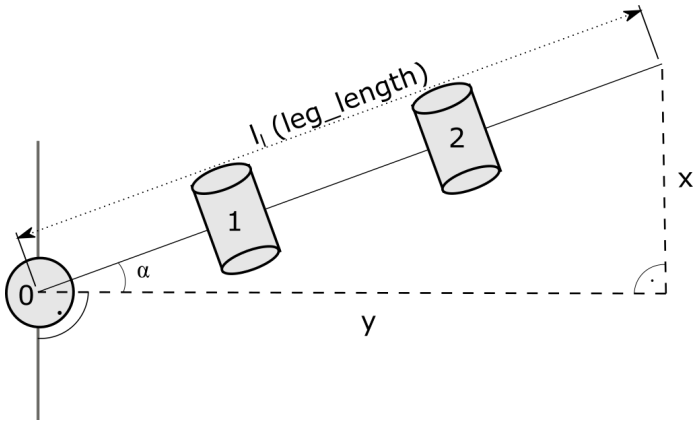
- Srovnání těla do vodorovné polohy. K tomu je využit akcelerometr, díky kterému se zjistí natočení těla v x a y složce. Následně se vypočte potřebná regulace výšky nohou, aby bylo tělo vodorovně.
- Vyrovnání těla do standardní výšky. Tato úprava je udělána přičtením rozdílu standardní výšky a aritmetického průměru výšek nohou k výškám nohou.

4.3 Odvození inverzní kinematiky

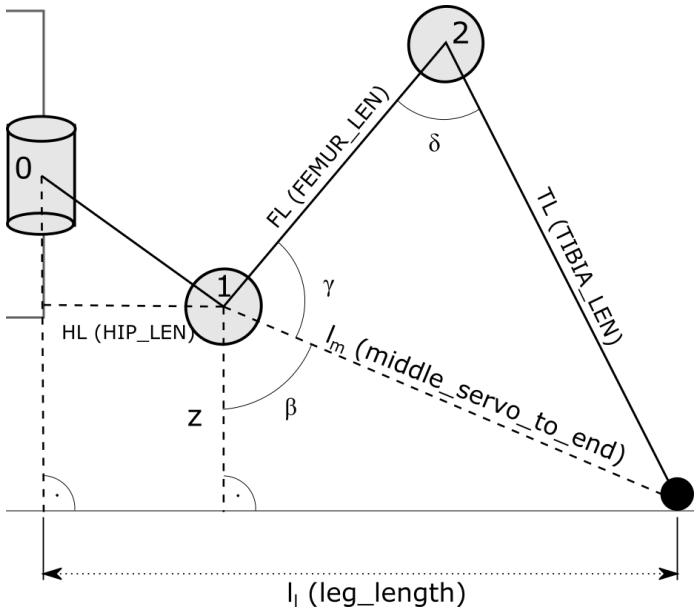
V této části práce popíši, jak jsem odvodila vzorce pro inverzní kinematiku nohy. Cílem je pro zadané souřadnice koncového bodu nohy najít odpovídající natočení serv.

Poté, co jsem návrh vytvořila sama a měla hotovou i implementaci, jsem při pokračování v práci narazila na několik stránek, které problematiku řeší stejně. Například: [19], [20], [21].

Obrázky 4.7 a 4.8 ukazují půdorys a bokorys daného problému pro jeho lepší pochopení. Šedé kruhy a válce na nich představují serva. Plné černé čáry jsou části končetiny. Konstanty jsou uvedeny velkými písmeny, zatímco proměnné malými.



Obrázek 4.7: Inverzní kinematika - půdorys



Obrázek 4.8: Inverzní kinematika - bokorys

Každá noha je ve svém vlastním prostoru a její inverzní kinematika je počítána nezávisle na poloze těla a ostatních nohách. Vzorce jsou pro všechny nohy stejné. Přestože jsou krajní páry nohou ve střední poloze serv vytočeny o 45° od středového páru, při výpočtech je uvažována základní poloha nohy kolmo k tělu.

Na začátku výpočtu jsou známy pouze konstanty v podobě délek článků nohy (`HIP_LEN`, `FEMUR_LEN`, `TIBIA_LEN`) a požadovaných souřadnic koncového bodu (`x`, `y`, `z`). `HIP_LEN` je vzdálenost nultého a prvního serva se zanedbáním výškového rozdílu, protože nulté servo umožňuje pohyb vodorovně, takže výškový rozdíl těchto dvou serv je neměnný.

Ze souřadnic `x` a `y` je vypočten úhel α (rovnice 4.8) a hodnota proměnné `leg_length` (rovnice 4.9) (obr. 4.7).

$$\alpha = \arctan\left(\frac{x}{y}\right) \tag{4.8}$$

$$l_l = \sqrt{x^2 + y^2} \tag{4.9}$$

Dále je určen úhel β (rovnice 4.10) a proměnná `middle_servo_to_end` (rovnice 4.11) (obr. 4.8)

$$\beta = \arctan\left(\frac{(l_l - HL)}{z}\right) \tag{4.10}$$

$$l_m = \sqrt{z^2 + (l_l - HL)^2} \tag{4.11}$$

Díky vypočtení `middle_servo_to_end` je známa délka všech stran v horním trojúhelníku na obrázku 4.8, a tedy lze pomocí Kosinové věty vypočítat úhel γ (rovnice 4.12) a δ (rovnice 4.13)

$$\gamma = \frac{l_m^2 + FL^2 - TL^2}{2 * l_m * FL} \tag{4.12}$$

$$\delta = \frac{FL^2 + TL^2 - l_m^2}{2 * FL * TL} \tag{4.13}$$

Nakonec jsou převedeny úhly na offsety úhlů od středních poloh jednotlivých serv.

$$servo_0_angle_offset = \alpha \tag{4.14}$$

$$servo_1_angle_offset = \beta + \gamma - 90 \tag{4.15}$$

$$servo_2_angle_offset = \delta - 90 \tag{4.16}$$

Kapitola 5

Implementace

Při implementaci jsem vytvořila kód v Pythonu určený pro RPi 3B. Pro správnou funkci programu musí platit, že další elektronika a její zapojení odpovídá popisu v kapitole 3.2.

5.1 Programovací jazyk

Pro implementaci jsem jako programovací jazyk zvolila Python. Je jedním z jazyků používaných při psaní programů pro roboty na fakultě, takže bude knihovna pro další studenty dobře využitelná. Pomalejší běh programu kvůli tomu, že je Python interpretovaný jazyk, není pro tuto aplikaci kritickou závadou. Naopak výhodou je rychlejší vývoj aplikace a existence následujících knihoven:

serial komunikace po USB [22]

math matematické výpočty [23]

numpy práce s poli a maticemi [24]

picamera ovládání Raspberry Pi kamery [13]

mpu6050 komunikace RPi a čipu MPU6050 na akcelerometru pomocí I^2C [17]

5.2 Složení aplikace

Program se skládá z několika tříd. Každá z nich se zaměřuje na některý z dílčích problémů.

Třída **Hexapod** slouží jako interface pro uživatele. Zároveň zaštiťuje všechny ostatní části a řídí jejich použití.

Třída **MovementTrajectoryManager** zajišťuje návrh dráhy těla jako celku a od toho odvozuje dráhy nohou relativně k tělu.

Třída **GaitManager** zajišťuje synchronizaci nohou.

Třída **Leg** slouží k výpočtu inverzní kinematiky a následnému převedení úhlů na příkazy pro servo kontroler.

Třída **Servo** reprezentuje serva a slouží k ukládání informací o nich.

Třída **CommunicationHandler** zajišťuje posílání příkazů servo kontroleru a přijímání zpětné vazby.

5.3 Interface knihovny

Knihovna nabízí následující funkce. Podrobnější popis lze nalézt v uživatelské příručce v příloze A.

walk(distance_x, distance_y, angle, speed=1.0)

Funkce přesune robota z místa na místo po rovině. Parametry `distance_x` a `distance_y` určují vzdálenost v centimetrech, o které se robot posune. Parametr `angle` určuje úhel v radianech, o který bude robot na konci chůze otočen. Parametr `speed` upraví základní rychlost kroku, jeho hodnota se pohybuje od 0,5 do 2.

walk_in_terrain(distance_x, distance_y, angle, speed=1.0)

Funkce přesune robota z místa na místo v terénu. Parametry `distance_x` a `distance_y` určují vzdálenost v centimetrech, o které se robot posune. Parametr `angle` určuje úhel v radianech, o který bude robot na konci chůze otočen. Parametr `speed` upraví základní rychlost kroku, jeho hodnota se pohybuje od 0,5 do 2.

lay_down()

Funkce robota přesune do polohy ležmo. Nohy jsou v základním postavení, ale pár centimetrů nad zemí.

stand_up(speed=3)

Funkce robota postaví. Parametr `speed` určuje čas ve vteřinách, za který se robot postaví.

center_head_tail()

Funkce vycentruje ocas a hlavu, včetně klepet.

set_body_height(body_height)

Funkce nastaví výšku těla.

open_mandibles()

Funkce otevře robotova klepeta.

close_mandibles()

Funkce zavře robotova klepeta.

grab_with_mandibles()

Funkce slouží k uchopování předmětů.

wait_for_all()

Funkce počká na dokončení probíhajícího příkazu těla i hlavy. Umožňuje tak jejich synchronizaci.

read_leg_touch_sensors(leg_ids)

Funkce vrátí list hodnot na dotykových senzorech na nohou. Parametr `leg_ids` je list indexů nohou, jejichž dotykové senzory mají být přečteny. Vrácený list odpovídá pořadí zadání nohou.

read_accelerometer()

Funkce vrátí hodnoty akcelerometru ve formě slovníku (osa: hodnota).

5.4 Konfigurace robota

Kvůli přenositelnosti kódu mezi dvěma hexapody stejného modelu a vybavenými stejnou elektronikou bylo třeba zajistit, aby bylo možné měnit konfiguraci parametrů specifických pro daný kus. Konfigurace je uložena v souboru hexapod_config.yml, který je načten při startu programu a data z něj jsou uložena do příslušných proměnných. Soubor je ve formátu YAML, což je strukturovaný souborový formát čitelný pro člověka. K jeho načtení do Pythonu je použita knihovna PyYAML.

Konfigurace obsahuje kalibraci serv, která se skládá ze dvou složek:

- Šířka pulzu, při které je servo ve střední poloze.
- Hodnota rozdílu šířky pulzu pro polohy, které mezi sebou svírají 90°. Při odečtení a přičtení této hodnoty k šířce pulzu ve střední hodnotě dostaneme rozsah šířky pulzu pro dané servo, ve kterém se servo pohybuje v rozmezí od -90° do +90°.

Hodnoty je třeba manuálně naměřit a zadat do souboru. Při vývoji knihovny jsem zjistila, že je potřeba kalibraci provádět pravidelně, protože serva se poměrně rychle rozladí.

Dále soubor obsahuje informace o číslech pinů, do kterých jsou zapojena jednotlivá serva a dotykové senzory. Informace jsou členěny podle příslušnosti k jednotlivým nohám.

5.5 Průběh funkce pro chůzi

Třída `Hexapod` obsahuje dvě funkce pro chůzi robota. Funkce `walk(distance_x, distance_y, angle, speed)` je určena pro chůzi na rovině a funkce `walk_in_terrain(distance_x, distance_y, angle, speed)` pro pohyb v terénu. Obě funkce mají stejné základní schéma.

Nejprve se parametrem `speed` upraví rychlost jednoho kroku. Parametrem lze základní hodnotu, která byla pro každou funkci zvolena po několika praktických testech, upravit na polovinu nebo až dvojnásobek.

Následně funkce na základě zadaných parametrů `x`, `y` a `angle` vybere, o jaký druh pohybu těla se bude jednat a zavolá příslušnou funkci ze třídy `MovementTrajectoryManager`, která jí vrátí informace o počtu kroků a potřebných polohách nohou.

Třída `GaitManager` se nastaví na požadovanou synchronizaci nohou.

Poté se iteruje přes počet kroků. V každém kroku probíhá cyklus, ve kterém se volá funkce `active_leg_position_function` ze třídy `GaitManager`, která vrátí list souřadnic, na které se mají přesunout jednotlivé nohy. Souřadnice se předají funkci `create_commands_from_positions`, která pro každou nohu zavolá funkci pro převedení pozice na příkazy pro servo kontroler, sloučí příkazy dohromady a vrátí je jako celek. Hotový příkaz pro servo kontroler je odeslán servo kontroleru přes třídu `CommunicationHandler`.

Po dokončení všech kroků funkce končí.

5.6 Chůze v terénu

Chůze v terénu probíhá stejně jako chůze po rovině. Třída `GaitManager` je přepnuta na terénní cyklus.

Pro zastavování nohou je využit následující postup. Funkce `active_legs_to_be_stopped_function` vrací indexy nohou, které mají být zastaveny při dotyku se zemí. Po odeslání příkazu pro pohyb serv je zavolána funkce `stop_on_touch_legs` a jsou jí předány indexy noh pro zastavení a čas, za který mají dorazit do koncového bodu. Ve funkci se spustí cyklus `while`. V cyklu se program dotazuje servo kontroleru na hodnoty dotykových senzorů zadaných nohou. Pokud se vrátí hodnota 255 (dotykový senzor je sepnutý), tak si program od servo kontroleru vyžádá aktuální pozici serv a pošle příkaz,

aby se nastavila na danou pozici, čímž se noha zastaví. Cyklus se přeruší, pokud jsou všechny zadané nohy zastaveny nebo pokud dojde čas jejich pohybu – nějaká z noh se nedostala na zem. Funkce vrací list indexů nohou, které nebyly zastaveny.

5.7 Implementace inverzní kinematiky

Implementace inverzní kinematiky se nachází ve funkci `get_servo_angles(x, y, z)` třídy `Leg`. Funkce vznikla přepsáním vzorečků z části 4.3 do Pythonu pomocí funkcí z knihovny math. Funkce vrací úhly jednotlivých serv.

Výše uvedená funkce je volána funkcí `get_servo_commands(x, y, z)` stejné třídy, která vrácené úhly převede na příkazy pro serva. Převod probíhá pomocí informací o kalibraci serv z konfiguračního souboru, které jsou uloženy v parametrech třídy `Servo`.

5.8 Tvorba trajektorie těla a nohou

Trajektorie těla je tvořena ve třídě `TrajectoryManager`. Třída má samostatné funkce pro každý druh pohybu.

5.8.1 Chůze po přímce

Trajektorie chůze po přímce je vytvořena ve funkci `walk_in_line(distance_x, distance_y, ideal_step_length)`. Je určena pro případ, kdy je funkcím pro chůzi zadána nenulová vzdálenost a nulový úhel.

Výpočet počtu kroků a vzdáleností krajních pozic podpůrné fáze od střední polohy podpůrné fáze pro jednotlivé nohy je proveden podle principu uvedeného v kapitole 4.2.2.1.

Funkce vrací počet kroků a čtyři listy souřadnic pro nohy (x_1, x_2, y_1, y_2), každý o délce 6. Souřadnice x_1 a y_1 jsou použity pro přední polohu podpůrné fáze cyklu, x_2 a y_2 jsou použity pro zadní polohu.

5.8.2 Otočení na místě

Trajektorie otočení na místě je vytvořena ve funkci `rotate_on_spot(angle, ideal_step_angle, leg_distances, leg_angles)`. Je určena pro případ, kdy je funkcím pro chůzi zadána nulová vzdálenost a nenulový úhel.

Výpočet počtu kroků a vzdáleností krajních pozic podpůrné fáze od střední polohy podpůrné fáze pro jednotlivé nohy je proveden podle principu uvedeného v kapitole 4.2.2.2.

Funkce vrací počet kroků a čtyři listy souřadnic pro nohy (x_1, x_2, y_1, y_2), každý o délce 6. Souřadnice x_1 a y_1 jsou použity pro přední polohu podpůrné fáze cyklu, x_2 a y_2 jsou použity pro zadní polohu.

5.8.3 Chůze po křivce

Trajektorie chůze po křivce je vytvořena ve funkci `walk_cubic_bezier_tripod(x, y, angle_to_be, starting_leg_group, leg_distances, leg_angles)`. Je určena pro případ, kdy je funkci pro chůzi po rovině zadána nenulová vzdálenost a nenulový úhel. V terénu je tato možnost převedena na chůzi po přímce a následné otočení na místě.

Výpočty jsou realizovány na základě návrhu v kapitole 4.2.2.3.

Na začátku jsou vytvořeny prázdné listy pozic po složkách (x_1, x_2, y_1, y_2). Tyto listy jsou v průběhu funkce naplněny listy se souřadnicemi nohou, odpovídající jednomu kroku. Délka vnějších listů se rovná počtu kroků, délka listů, ze kterých se skládají, je 6. Kvůli tomu, že skupiny nohou začínají ve fázích cyklu 0 a 4, tak se data k jednomu kroku skládají z půlkroku jedné skupiny (skupina se ze střední polohy posune do zadní polohy) a půlkroku druhé skupiny nohou (skupina se z přední polohy posune do střední polohy).

Funkce iteruje přes půlkroky. Při každém půlkroku vypočítá pozice pro nohy, které jsou zrovna v podpůrné fázi a uloží je po složkách do příslušných listů.

Funkce vrací vytvořené listy. Díky délce listů se přenáší informace o počtu kroků.

5.9 Synchronizace nohou

Synchronizaci nohou zajišťuje třída `GaitManager`. Třída funguje na principu konečného automatu. Pro každý typ cyklu nohy je implementován samostatný automat. Typ cyklu je možné vybrat pomocí funkce `set_gait(gait_id)`. Ve funkci se načtou do parametrů třídy hodnoty specifické pro daný cyklus a resetuje se fáze cyklu.

Funkce `active_leg_position_function` vrací pozice nohou v aktuální fázi cyklu. Parametrem funkce jsou offsety krajních poloh od střední polohy cyklu a výška těla. Funkcí `active_next_phase_function` se automat posune do dalšího stavu. Při tom se nastaví příznak, jestli byl tímto přechodem dokončen krok. To je důležité, protože každý typ cyklu nohy může krok dokončovat v jiné fázi.

5.10 Komunikace se servo kontrolerem

Komunikace se servo kontrolerem probíhá přes USB rozhraní. Zařizuje ji třída `CommunicationHandler` pomocí knihovny serial. Komunikace se skládá z odesílání příkazů a přijímání dat, které byly vyžádány předchozími příkazy.

5.10.1 Odesílání příkazů

Odesílání příkazů probíhá ve dvou variantách:

- Odeslání ihned bez ohledu na dokončení předcházejících příkazů. Tuto variantu používám pro odesílání dotazů na stav senzorů a na příkazy, které mají zastavit serva na místě, například při chůzi v terénu. Ani v jednom z případů není žádoucí, aby příkaz čekal na dokončení předcházejícího.

- Odeslání příkazu až ve chvíli, kdy skončil předešlý příkaz na určených skupinách těla. Skupiny jsou dvě - tělo a hlava+ocas. Při odesílání se zapamatuje čas, kdy může být odeslán další příkaz a při dalším zavolání se počká, dokud tento čas ještě nenastal.

5.10.2 Přijímání zpětné vazby

Data se přijímají po jednom bytu, protože odpovědi všech dotazových příkazů jsou jeden byte. Funkce `read_byte` přečte 1 byte z linky a vrátí ho v podobě čísla. Funkce `read_char` zavolá funkci `read_byte`, jejíž návratovou hodnotu převede na char a vrátí ho.

5.11 Zprovoznění akcelerometru

Knihovna mpu6050, která zajišťuje vyčítání dat z akcelerometru, je obalena vlastní třídou, aby uživatel nemusel znát a zadávat adresu zařízení. Je zachováno stejné rozhraní. Navíc je přidána funkce, která vrátí úhel odchylky v radiánech ve složkách x a y, vzorečky pro výpočet jsou převzaty z [25].

5.12 Použití klepet

Pro pohyb pouze klepet jsou implementovány tři funkce.

- `open_mandibles()` otevře klepeta.

- `close_mandibles()` zavře klepeta.

- `grab_with_mandibles()` slouží k uchopování předmětů. Funkce zavolá příkaz na zavření klepet. Pokud v průběhu uzavírání vzroste hodnota na tlakovém senzoru, tak klepeta zastaví pomocí funkce servo kontroleru STOP.

Pro jejich synchronizaci s příkazy pro tělo se používá funkce `wait_for_all()` třídy hexapod, která počká na dokončení probíhajících příkazů.

5.13 Demo aplikace

Vytvořená demo aplikace pro ukázku použití knihovny se pro lepší přehlednost skládá ze tří částí a demonstrace zprovoznění kamery. Soubory jsou vytvořeny tak, že při spuštění mají být ve stejné složce jako soubory knihovny. Příklady kódů demo aplikací jsou umístěny v uživatelské příručce v příloze A.

Ukázka chůze po rovině je umístěna v souboru demo_rovina.py. Robot začíná vleže, postaví se, přejde šikmo po přímce, otočí se namístě o 180° a jde zpět po křivce, kde si opět lehne.

Soubor demo_teren.py obsahuje demo aplikaci chůze v terénu. Robot na začátku leží, je mu nastavena jiná než základní výška těla a na tu se postaví. Dále robot přejde dopředu, otočí se o 180°, vrátí se zpět a lehne si.

Demo aplikace pro přenášení předmětu klepety hexapodu se nachází v souboru demo_prenaseni.py. Robot začíná vleže. Následně se postaví, otevře klepeta a přejde kousek dopředu. Dále sníží svoje tělo a uchopí předmět, poté se zvedne, popojde dopředu, sníží se, otevře klepeta, zvedne se, odcouvá a ulehne.

Pro demonstraci zprovoznění kamery na RPi slouží kód umístěný v souboru demo_kamera.py využívající knihovnu picamera.

Kapitola 6

Testování

Testování knihovny jsem provedla v reálném prostředí na robotovi modelu A-Pod od firmy Lynxmotion. Pokusy jsem prováděla v Laboratoři inteligentních vestavných systémů.

Videa se záběry chůze robota a přenášení předmětů pomocí klepet lze nalézt na přiloženém CD.

6.1 Chůze po rovině

Při testování chůze po rovině jsem samostatně testovala chůzi po přímce, otočení na místě i chůzi po křivce. Pro každou možnost jsem udělala sérii testů, při kterých jsem pozorovala a měřila, jak robot chodí.

Chůzi po přímce jsem testovala ve dvou úrovních. Jednak jsem vyzkoušela, že robot chodí, a to ve směru zadaném uživatelem. Robot se při všech pokusech pohyboval plynule definovaným směrem.

Druhý test ověřoval přesnost chůze vpřed a vzad. Měřením jsem zjistila, že neujde přesně vzdálenost, která mu byla zadána. Cestou dopředu robot vzdálenost přecházel, dozadu ji naopak neušel. Odchylka se různila podle zvolené délky kroku, ale pro jedno nastavení byla stále stejná. V návaznosti na tyto výsledky jsem hledala chybu v programu, ke které tato systematičnost navádí. Avšak nic jsem nenašla. Posílané příkazy pro serva jsou shodné pro chůzi dozadu i dopředu, jen se posílají v opačném pořadí. Jako možné příčiny odhaduji nepřesné smontování robota a nepřesnost serv a jejich nedostatečně dobrou kalibraci. Tu je obtížné provést a je třeba ji pravidelně opakovat. Dále se materiál, ze kterého je robot vyroben, výrazně ohýbá při zátěži. Tyto problémy se vyskytují i u podstatně kvalitnějších robotů.

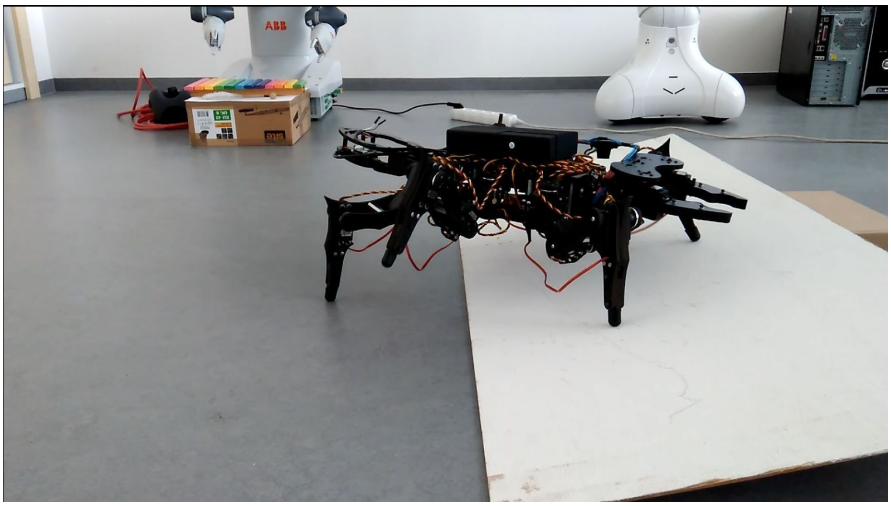
Při otáčení na místě se robot skutečně otočí a jeho střed těla zůstane na stejném místě. Stejně jako u chůze po přímce má robot problém s přetáčením, které je v obou směrech pohybu stejné.

Při testování chůze po křivce jsem zkoušela několik různých tvarů a jejich směrů. Ve všech případech se robot na pohled pohyboval správně po zadané křivce. Ve chvíli, kdy robot procházel ostřejší zatáčkou, jeho pohyb nebyl podle očekávání úplně plynulý a přirozený. Stejně jako u dalších druhů pohybu se robot přesunul do větší vzdálenosti, než měl zadáno, avšak se zachováním tvaru křivky.

6.2 Chůze v terénu

Chůze v terénu byla testována na dvou typech překážek. Jednou byly krabice o výšce 5 cm a druhou nakloněná rovina s úhlem 12°.

Krabice, které byly uloženy pouze v jedné vrstvě, dokázal robot překonat. Jediné problémy nastávaly, když došlápl na okraj krabice. Noha se při dotyku s krabicí zastavila, ale když na ni byla přenesena váha, tak se sklouzla přes hranu. To způsobilo rozhození robota, než dokončil krok, ale poté bylo tělo opět vyrovnáno díky použití akcelerometru.



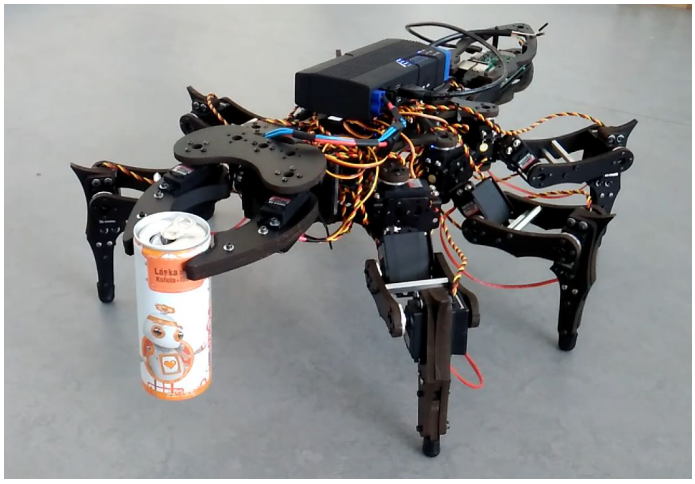
Obrázek 6.1: Chůze robota po nakloněné rovině

V případě umístění dvou pater krabic na sebe v podobě schodů měl robot s průchodem problémy. Algoritmus, který vyrovnává tělo, aby průměrná odchylka výšky nohou od stanovené výšky těla byla nulová, neumožnil takto rychlé stoupání.

Nakloněná rovina byla pro robota schůdnější než schody, avšak její náklon byl na hranici jeho schopností.

6.3 Přenášení předmětů

Testování přenášení předmětů probíhalo pomocí aplikace pro přenášení předmětů nacházející se v souboru demo_prenaseni.py. K testování byly použity různě těžké předměty různých tvarů – prázdná plechovka, plná plechovka 0.33 l, plná láhev 0.5 l, krabička od RPi a kartonová krabice 5x12x17,cm.



Obrázek 6.2: Přenášení plechovky pomocí klepet

Všechny předměty, až na kartonovou krabici, se alespoň jednou podařilo úspěšně přenést. Obecně dokázal robot přenášet lépe kulaté předměty, protože pro ně má uzpůsobená klepeta.

Prázdnou plechovku dokázal robot zvednout, ať už ji chytil do jakékoli části klepet. Plná plechovka mu většinou vyklouzla, když se s ní zvedal, ale když ji uchopil částí klepet nejblíže hlavě, tak ji dokázal odnést. Láhev byla zařazena kvůli své váze, protože její tvar neumožňoval, aby robotovi vyklouzla. Robot dokázal láhev přenést bez viditelných obtíží při chůzi. Krabička od RPi měla příhodný rozměr a robot ji uchopoval oběma výběžky na klepetech, takže díky její nízké hmotnosti neměl hexapod s přenesením problém. Kartonová krabice byla moc velká a nerovnoměrně zatížená, robot ji nedokázal pořádně uchopit, a proto se protočila a vypadla mu.

Závěr

Cílem bakalářské práce bylo navrhnout a vytvořit programové vybavení pro pohyb šestinohého robota po rovině i v terénu, které by mělo podobu knihovny. Implementace byla zaměřena na model A-Pod od firmy Lynxmotion a jeho elektroniku v podobě Raspberry Pi 3B a servo kontroleru SSC-32U. Dále bylo cílem přenášení předmětů pomocí klepet na hlavě, zprovoznění akcelerometru i kamery a vytvoření demo aplikace.

Při analýze existujících řešení jsem nenalezla žádné vyhovující, proto jsem navrhla vlastní. Knihovna je napsána v programovacím jazyce Python, který je široce využíván pro programování robotů na fakultě a mé řešení bude moci být dále rozvíjeno. V neposlední řadě jsem jej zvolila kvůli jeho užitečným knihovnám.

Pro pohyb robota jsem navrhla a implementovala tři typy pohybu: chůzi po přímce, otočení na místě a chůzi po křivce. Na každý z nich jsem vytvořila samostatný algoritmus pro tvorbu trajektorie těla a nohou. Pro synchronizaci nohou jsem implementovala typ chůze tripod. Při chůzi v terénu jsem využila dotyková čidla na nohou a akcelerometr. Programové vybavení obsahuje i ovládání klepet a díky tlakovému senzoru umožňuje uchopování a přenášení předmětů.

Pro ovládání kamery jsem našla knihovnu a její zprovoznění demonstruji pomocí jedné z demo aplikací. Další tři demo aplikace ukazují použití mnou vytvořené knihovny pro pohyb po rovině, v terénu a pro přenášení předmětů. Tím jsem splnila cíle bakalářské práce.

V rámci průběžného testování jsem zjistila, že robot nechodí přesně. V návaznosti na to jsem testovala svůj kód, ale chybu jsem nenašla. Domnívám se, že chyba nastává kvůli špatnému smontování spojeném s nekvalitou materiálu, který se při zátěži výrazně ohýbá, a kvůli nepřesnosti serv. Tyto problémy postihují i jiné, kvalitnější roboty.

Výsledky mé práce umožní dalším studentům na FIT ČVUT v Praze vytvářet složitější aplikace. Projekt může být v budoucnu rozšířen přidáním dalšího typu chůze nebo vylepšením pohybu v terénu.

Literatura

[1] Bjelonic, M.; další: Weaver: Hexapod robot for autonomous navigation on unstructured terrain. *Journal of Field Robotics*, ročník 35, č. 7, 2018: s. 1063–1079, ISSN 1556-4959.

[2] Preumont, A.; Alexandre, P.; Ghuys, D.: Gait analysis and implementation of a six leg walking machine. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, IEEE, 1991, s. 941–945.

[3] Wang, Z. Y.; Ding, X. L.; Rovetta, A.: Analysis of typical locomotion of a symmetric hexapod robot. *Robotica*, ročník 28, č. 6, 2010: s. 893–907, ISSN 0263-5747.

[4] Lynch, K. M.; Park, F. C.: *Modern robotics: mechanics, planning, and control*. Cambridge, United Kingdom: Cambridge University Press, 2017, ISBN 978-1-107-15630-2.

[5] Tedeschi, F.; Carbone, G.: Design Issues for Hexapod Walking Robots. *Robotics*, ročník 3, č. 2, 2014: s. 181–206, ISSN 2218-6581.

[6] Lopez-Franco, C.; další: A soft computing approach for inverse kinematics of robot manipulators. *Engineering Applications of Artificial Intelligence*, ročník 74, 2018: s. 104–120, ISSN 0952-1976.

[7] Ting, L. H.; Blickhan, R.; Full, R. J.: Dynamic and static stability in hexapedal runners. *Journal of Experimental Biology*, ročník 197, č. 1, 1994: s. 251–269, ISSN 0022-0949.

[8] Žára, J.; Beneš, B.; Felkel, P.: *Moderní počítačová grafika*. Praha: Computer Press, 1998, ISBN 8072260499, 9788072260492.

[9] A-pod, an ant inspired hexapod [online]. vBulletin Solutions, Inc., ©2019, [cit. 2019-5-9]. Dostupné z: http://forums.trossenrobotics.com/showthread.php?2762-A-pod-an-ant-inspired-hexapod

[10] STEMI hexapod v2 [online]. GitHub, Inc., ©2019, [cit. 2019-4-18]. Dostupné z: https://github.com/stemi-education/stemi-hexapod

[11] PhantomX Hexapod Phoenix Code [online]. Trossen Robotics, [cit. 2019-4-18]. Dostupné z: https://learn.trossenrobotics.com/10-interbotix/crawlers/phantomx-hexapod/68-phoenix-code.html

[12] Raspberry Pi 3 Model B [online]. Raspberry Pi Foundation, [cit. 2019-4-20]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[13] Jones, D.: *Picamera 1.13 Documentation [online]*. ©2013-2016, [cit. 2019-4-21]. Dostupné z: https://picamera.readthedocs.io/en/release-1.13/index.html

[14] Lynxmotion: *Lynxmotion SSC-32U USB Servo Controller Board - Electronics Guide V1.2 [online]*. červenec 2017, [cit. 2019-3-29]. Dostupné z: http://www.lynxmotion.com/images/data/lynxmotion_ssc-32u_usb_user_guide.pdf

[15] A-Pod Bot Board II / BASIC Atom Pro Tutorial [online]. Lynxmotion, [cit. 2019-5-11]. Dostupné z: http://www.lynxmotion.com/images/html/build184.htm

[16] Raspberry Pi Foundation: *Raspberry Pi Camera Module - Software [online]*. listopad 2018, [cit. 2019-4-21]. Dostupné z: https://www.raspberrypi.org/documentation/raspbian/applications/camera.md

[17] mpu6050 [online]. GitHub, Inc., ©2019, [cit. 2019-4-21]. Dostupné z: https://github.com/Tijndagamer/mpu6050

[18] Geometric Transformations [online]. [cit. 2019-5-11]. Dostupné z: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/geometry/geo-tran.html

[19] Liang, O.: Inverse Kinematics Implementation for Hexapod Robots and Quadruped Robots [online]. ©2013-2018, [cit. 2019-5-4]. Dostupné z: https://oscarliang.com/inverse-kinematics-implementation-hexapod-robots/

[20] Inverse Kinematic equations for Lynxmotion 3DOF legs [online]. RobotShop inc., ©2018, [cit. 2019-5-4]. Dostupné z: https:

//www.robotshop.com/community/forum/t/inverse-kinematic-equations-for-lynxmotion-3dof-legs/21336/3

[21] KDL Inverse Kinematics for a Hexapod [online]. ROS Answers, [cit. 2019-5-4]. Dostupné z: https://answers.ros.org/question/317025/kdl-inverse-kinematics-for-a-hexapod/

[22] Liechti, C.: pySerial knihovna [online]. ©2001-2016, [cit. 2019-5-10]. Dostupné z: https://pyserial.readthedocs.io/en/latest/pyserial.html

[23] The Python Standard Library - math [online]. Python Software Foundation, ©2001-2019, [cit. 2019-5-10]. Dostupné z: https://docs.python.org/3/library/math.html

[24] NumPy v1.16 Manual [online]. The SciPy community, ©2008-2019, [cit. 2019-5-10]. Dostupné z: https://docs.scipy.org/doc/numpy/

[25] Accelerometers [online]. HobbyTronics Ltd , ©2019, [cit. 2019-5-11]. Dostupné z: http://www.hobbytronics.co.uk/accelerometer-info

Příloha **A**

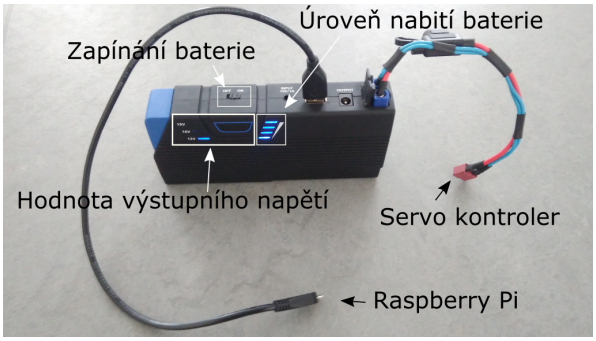
Uživatelská příručka

Příručka popisuje rozhraní a použití knihovny pro pohyb hexapodu vytvořené v rámci bakalářské práce. Knihovna se skládá ze tří souborů: hexapod.py, accelerometer.py a hexapod_config.yml, kde všechny musí být umístěny v jedné složce.

A.1 Zapnutí robota

Robota umístěte na zem nebo stojan z krabic. V obou případech musí mít robot volné nohy.

Zapněte baterii (obr. A.1). Zkontrolujte nabití baterie (dva a méně čárek je nedostačující) a nastavte výstupní napětí na 12 V. Ještě když kontrolky svítí, připojte micro USB do Raspberry Pi, které se tím zapne.



Obrázek A.1: Popis zapojení baterie

Připojte se k Raspberry Pi po SSH.

Až budete mít připravený program, zapojte napájení do servo kontroleru a spusťte program.

Serva není dobré nechávat dlouho zapnutá k důvodu přehřívání, proto robota v pauzách mezi programy nechávejte v ležící poloze, ale nejlépe je vypněte vypojením napájení servo kontroleru.

A.2 Dostupné funkce knihovny

Knihovna je napsána v Pythonu. Pro použití ve vlastním programu, který je umístěn ve stejné složce jako soubory knihovny, importujte modul následovně:

```
from hexapod import Hexapod
```

Knihovna disponuje následujícími funkcemi:

walk(distance_x, distance_y, angle, speed=1.0)

Funkce přesune robota z místa na místo po rovině. Parametry `distance_x` a `distance_y` určují vzdálenost v centimetrech, o které se robot posune. Parametr `angle` určuje úhel v radianech, o který bude robot na konci chůze otočen. Parametr `speed` upraví základní rychlost kroku, jeho hodnota se pohybuje od 0,5 do 2.

Robot umí na rovině chodit třemi způsoby. Chůze po přímce, která se provede při zadání nenulové vzdálenosti a nulového úhlu. Otočení na místě, když je zadána nulová vzdálenost a nenulový úhel. Chůze po kubické Bézierově křivce, když je zadána nenulová vzdálenost i nenulový úhel.

walk_in_terrain(distance_x, distance_y, angle, speed=1.0)

Funkce přesune robota z místa na místo v terénu. Parametry `distance_x` a `distance_y` určují vzdálenost v centimetrech, o které se robot posune. Parametr `angle` určuje úhel v radianech, o který bude robot na konci chůze otočen. Parametr `speed` upraví základní rychlost kroku, jeho hodnota se pohybuje od 0,5 do 2.

Robot umí v terénu chodit dvěma způsoby. Chůze po přímce, která se provede při zadání nenulové vzdálenosti a nulového úhlu. Otočení na místě, když je zadána nulová vzdálenost a nenulový úhel. Pokud je zadána nenulová vzdálenost i úhel, tak robot na místo dojde po přímce a tam se otočí.

lay_down()

Funkce robota přesune do polohy ležmo během 3 vteřin. Nohy jsou v základním postavení, ale pár centimetrů nad zemí.

Tato funkce je doporučena pro začátek skriptů, kdy robot nemá zpevněná serva. Nebo naopak chce uživatel robota vypnout.

stand_up(speed=3)

Funkce robota postaví. Parametr `speed` určuje čas ve vteřinách, za který se robot postaví, pokud se nezadá, tak jsou defaultní 3 vteřiny.

center_head_tail()

Funkce vycentruje ocas a hlavu, včetně klepet.

set_body_height(body_height)

Funkce nastaví výšku těla, ale nepřevede na ni robota. Je doporučeno po této funkci zavolat funkci `stand_up`.

open_mandibles()

Funkce otevře robotova klepeta během 2 vteřin.

close_mandibles()

Funkce zavře robotova klepeta během 2 vteřin.

grab_with_mandibles()

Funkce slouží k uchopování předmětů. Funkce zavolá příkaz na zavření klepet. Pokud v průběhu uzavírání vzroste hodnota na tlakovém senzoru, tak klepeta zastaví.

wait_for_all()

Funkce počká na dokončení probíhajícího příkazu těla i hlavy. Umožňuje tak jejich synchronizaci.

read_leg_touch_sensors(leg_ids)

Funkce vrátí list hodnot na dotykových senzorech na nohou. Parametr `leg_ids` je list indexů nohou, jejichž dotykové senzory mají být přečteny. Vrácený list odpovídá pořadí zadání nohou. Vrácené hodnoty jsou 0 nebo 255.

read_accelerometer()

Funkce vrátí hodnoty akcelerometru ve formě slovníku (osa: hodnota).

A.3 Ukázka použití knihovny

V této kapitole se nachází ukázky základního použití knihovny. Více příkladů je v demo aplikacích na přiloženém CD.

```
from hexapod import Hexapod
import math

hexapod = Hexapod()

hexapod.lay_down()
hexapod.center_head_tail()
hexapod.stand_up(2)
hexapod.walk(90, 90, 0, 1.5)
hexapod.walk(0, 0, math.pi, 1)
hexapod.walk(90, 90, -math.pi/2, 0.9)
hexapod.walk(0, 0, -math.pi/2, 1.7)
hexapod.stand_up(1)
hexapod.lay_down()
```

Listing 1: Ukázka kódu pro chůzi po rovině

```
from hexapod import Hexapod

hexapod = Hexapod()

hexapod.lay_down()
hexapod.center_head_tail()

hexapod.stand_up(1)
hexapod.open_mandibles()

hexapod.walk(10, 0, 0, 1)
hexapod.set_body_height(7)
hexapod.stand_up(1)

hexapod.wait_for_all()
hexapod.grab_with_mandibles()
hexapod.wait_for_all()

hexapod.set_body_height(12)
hexapod.stand_up(1)
```

Listing 2: Ukázka kódu pro zvednutí předmětu

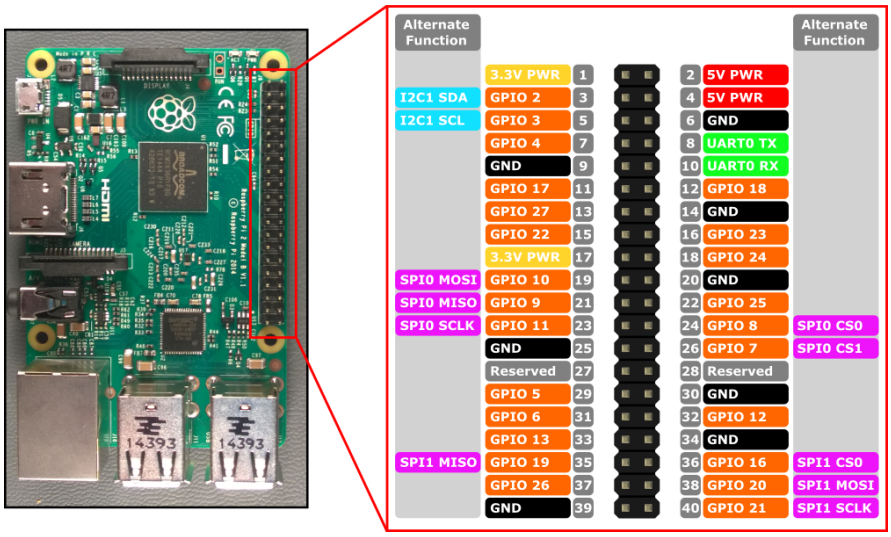
Příloha B

Zapojení akcelerometru

Akcelerometr je k RPi připojen přes rozhraní I^2C. Drát má na straně RPi piny, na straně akcelerometru konektor RJ25.

Tabulka B.1: Popis drátů pro připojení akcelerometru k RPi po rozhraní I^2C

Číslo RPi pinu	Barva kabelu	Funkce
5	modrá	SCL
3	žlutá	SDA
6	zelená	GND
4	červená	VCC
-	černá	ADO
-	bílá	INT



Obrázek B.1: Raspberry Pi - čísla pinů [https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi]

Obrázek B.2: Zapojení I^{2}C na RPi

Příloha C

Seznam použitých zkratek

RPi Raspberry Pi

USB Universal Serial Bus

I^{2}C Inter-Integrated Circuit

SSH Secure Shell

UART Universal Asynchronous Receiver/Transmitter

PWM Pulse-width modulation (Pulsně šířková modulace)

CSI Camera Serial Interface

GPIO General-purpose input/output

Příloha D

Obsah přiloženého CD

```
readme.txt ..................................... stručný popis obsahu CD
src
  impl ..................................... zdrojové kódy implementace
  doc ..................................... dokumentace zdrojového kódu
  thesis ....................... zdrojová forma práce ve formátu LaTeX
text ..................................................... text práce
  thesis.pdf ............................ text práce ve formátu PDF
videos ............................................ videa chůze robota
```