Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Diplomová práce

Robotická platforma s více končetinami

Autor práce: Bc. Martin Faktor
Vedoucí práce: Ing. Petr Weissar, Ph.D.

Plzeň 2015

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin FAKTOR**
Osobní číslo:	**E13N0121P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Dopravní elektroinženýrství a autoelektronika**
Název tématu:	**Robotická platforma s více končetinami**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

Navrhněte a realizujte robotickou platformu založenou na principu kráčení několika končetin.

1. Zvolte vhodnou mechanickou konstrukci.
2. Navrhněte a realizujte řidicí elektroniku.
3. Uvažujte možnost dálkového ovládání uživatelsky nebo z počítače, příp. vhodnou formu autonomního chování.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**30 - 40 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Petr Weissar, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**15. října 2014**
Termín odevzdání diplomové práce:	**11. května 2015**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2014

Abstrakt

Předkládaná diplomová práce se zabývá kompletním návrhem a realizací malého modelu robotické platformy s více končetinami, která je ovládána smartphonem s operačním systémem Android přes rozhraní Bluetooth. První část práce je zaměřena na mechanickou konstrukci. Řeší se zde topologie, konstrukce, výroba jednotlivých dílu a výběr akčních členů. Druhá část práce se zaobírá matematickým popisem celého robota a implementací chůze. V další části práce je rozebráno elektrické vybavení robota i s popisem firmwaru pro řídicí jednotku. Poslední část práce popisuje realizaci ovládání, které je implementováno jako softwarový joystick na smartphonu.

Klíčová slova

Hexapod, robotická platforma, kinematika, STM32, Google Android, Bluetooth

Abstract

Faktor, Martin. *Robotic platform with multiple legs* [*Robotická platforma s více končetinami*]. Pilsen, 2015. Master thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Petr Weissar

---

This diploma thesis deals with complete design and implementation of small model of a robotic platform with multiple legs. The robotic platform is controlled by the smartphone with operating system Android via Bluetooth interface. First part of this thesis is focused on the mechanical construction. This part contains topology solutions of the body, construction, production of the parts and selection of the actuators. Second part of the thesis deals with the mathematical description of the robotic platform and describes the implementation of walking. The following part of the thesis describes the electrical equipment of the robot with firmware control unit. Last part of the thesis is focused on implementation of the control, which is realised by a software joystick on smartphone.

Keywords

Hexapod, robotic platform, kinematics, STM32, Google Android, Bluetooth

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 4. května 2015

Bc. Martin Faktor

.....

Podpis

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

ADT	Android Development Tools.
AHB	Advanced High-performance Bus. Sběrnice.
APB1	Advanced Peripheral Bus. Sběrnice
API	Application programming interface.
ARM	Advanced RISC Machine. Architektura procesoru.
BSRR	Bit Set/Reset Registr. Registr pro nastavování výstupních portů.
CAD	Computer-aided design. Počítačem podporované navrhování.
CRC-8-CCITT	Generující polynom.
D2PAK	Double Decawatt Package.
DC-DC Step-down	Stejnosměrný snižující měnič napětí.
FLASH	Nevolatilní elektricky programovatelná paměť s náhodným přístupem.
FPGA	Field-programmable gate array. Programovatelné hradlové pole.
GPOI	General-purpose input/output. Obecně použitelné vstupní/výstupní porty.
IDE	Integrated development environment.
JTAG	Joint Test Action Group. Debug rozhraní.
LED	Light-emitting diode. Svítivá dioda.
MOSFET	Metal–oxide–semiconductor field-effect transistor. Polem řízený tranzistor.
PC	Personal computer. Osobní počítač.
PLA	Polylactic acid.
PVC	Polyvinyl chloride.
RISC	Reduced instruction set computing. Procesor s redukovanou instrukční sadou.
SDK	Software development kit.
SPP	Serial Port Profile.
SRAM	Volatilní statická paměť s náhodným přístupem.
SWD	Serial Wire Debug. Debug rozhraní.
SysTick	Systémový časovač.
Timer/Counter	Čítač/časovač.
U_{DS}	Drain-to-Source Voltage.

U_{GS}	Gate-to-Source Voltage.
USART	Universal synchronous/asynchronous receiver/transmitter. Synchronní a asynchronní sériové rozhraní.
UUID	Universally unique identifier.
XML	Extensible Markup Language.

1

Úvod

V dnešním světě jsou robotické platformy založené na kráčení několika končetin běžně používané. Příkladem takovéto platformy může být komerčně dostupný produkt *John Deere Walking Tractor* (obr. 1.1) používaný pro lesní práce ve špatně přístupném terénu. Dalším příkladem této platformy je robotická mula *LS3 Mule* (obr. 1.1) vyvíjena americkou armádou pro transport těžkého nákladu a vybavení ve špatně přístupném terénu. Výhodou robotické platformy s končetinami oproti kolovým mechanismům je nesrovnatelně větší průchodnost terénem a menší následky devastace krajiny. Mezi nevýhody těchto platforem může patřit menší rychlost pohybu a větší energetická náročnost a složitost celého řízení.

Cílem této diplomové práce je celistvý návrh a realizace malého modelu kráčející robotické platformy s více končetinami. Jelikož se jedná o kompletní řešení, je v práci popsána mechanická konstrukce těla, výroba jednotlivých dílů, matematický popis celého robota, hardwarová část elektronického vybavení, softwarové vybavení robota a realizace ovládání, které je implementováno jako softwarový joystick na smartphone.

(a) John Deere Walking Tractor |Převzato z [1]|

(b) LS3 Mule |Převzato z [2]|

Obr. 1.1: Příklad robotických platforem s více končetinami

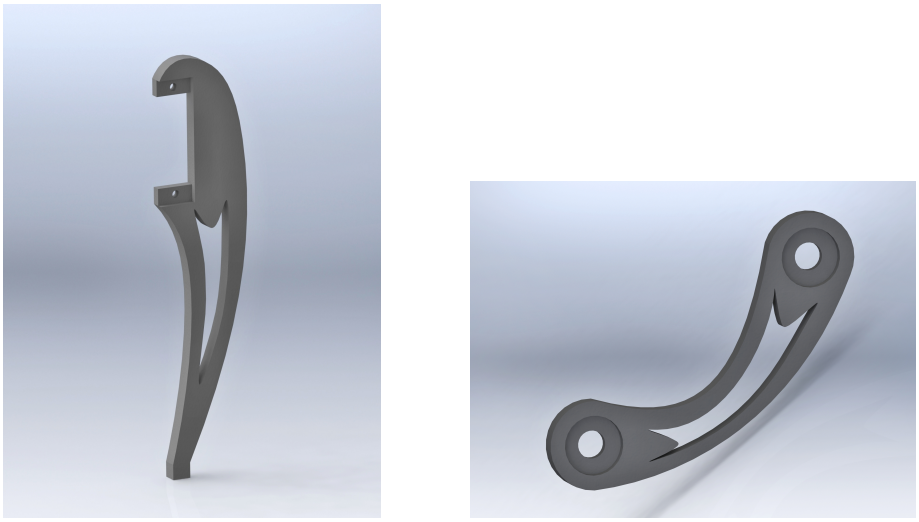
2

Mechanická konstrukce

Pod pojmem robotická platforma s více končetinami si lez představit konstrukce, které pro svůj pohyb využívají od jedné končetiny, až po desítky končetin. Nejpoužívanější platformy obsahují čtyři nebo šest končetin. Pro tuto diplomovou práci byla vybrána šestinohá varianta platformy, která se označuje jako hexapod. Název byl zděděn po živočiších z kmene členovců a podkmene šestinozí (Hexapoda).

2.1 Návrh mechanických částí

Celý robot se skládá ze šesti končetin, kde tři jsou levé a tři pravé. Rozdíl mezi nimi je jen v zrcadlení, a proto zde budou uvedeny díly jen na jednu končetinu. Další dva díly tvoří horní a spodní část těla a posledním dílem jsou akční členy, které taktéž tvoří konstrukční prvky, a pro bylo nutné je namodelovat pro ověření správných rozměrů dílů, které na ně navazují. Celý návrh byl proveden v 3D CAD softwaru SolidWorks 2013 [4].

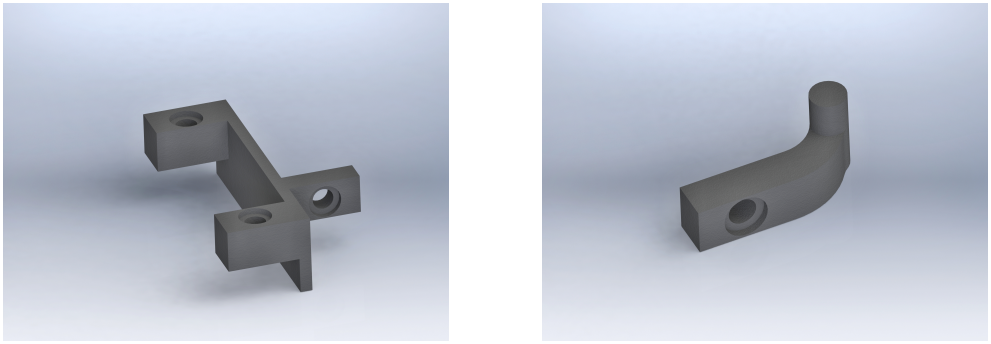


(a) Holenní kost (b) Stehenní kost

Obr. 2.1: Části končetiny

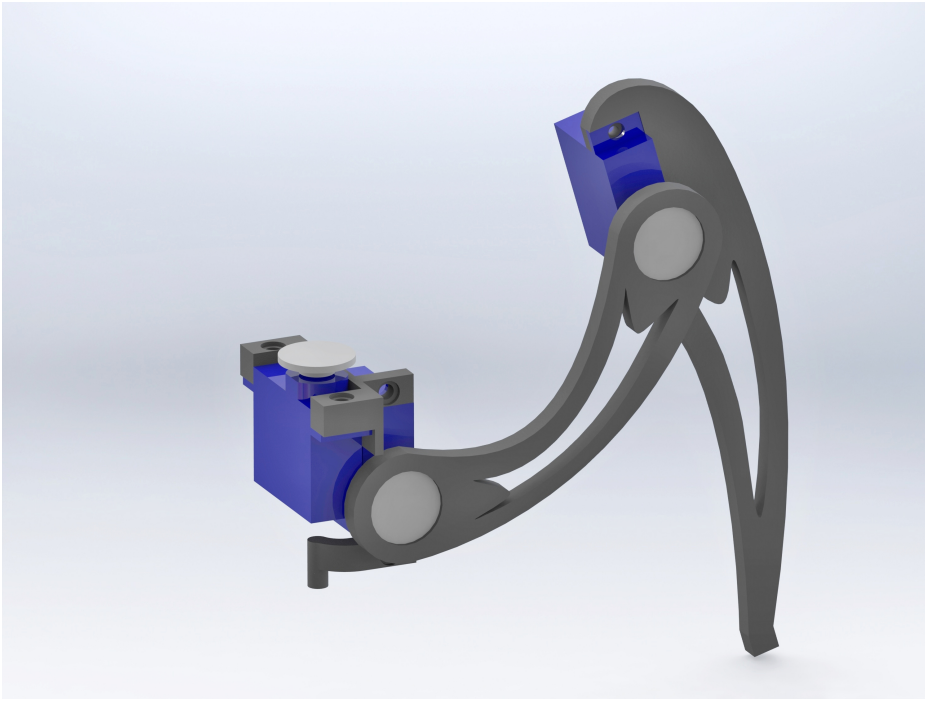
Každá končetina je složena ze tří akčních členů, jedné holenní kosti, jedné stehenní kosti a dvou spojujících prvků, které drží akční členy pohromadě a fixují končetinu do těla robota. Na obrázku 2.1 je 3D model holenní kosti, kde v horní části je místo pro uložení akčního členu, který je uchycen za stacionární část pouzdra a plní funkci kolenního kloubu.

Na obrázku 2.1 je model stehenní kosti, který navazuje na holenní kost a je s touto kostí spojen přes akční člen. Uchycení akčního členu do stehenní kosti je pomocí výstupního kotoučku akčního členu, který je do stehenní kosti vlepen. Na druhé straně je stehenní kost připevněna stejným mechanismem ke dvojici akčních členů, které tvoří kyčelní kloub. Obrázek 2.2 znázorňuje spojovací část mezi dvěma akčními členy, které tvoří kyčelní kloub. Posledním prvkem končetiny je trn na spodní části kyčelního kloubu (obr. 2.2), který fixuje končetinu do spodní části těla. Celý model končetiny je na obrázku 2.3.



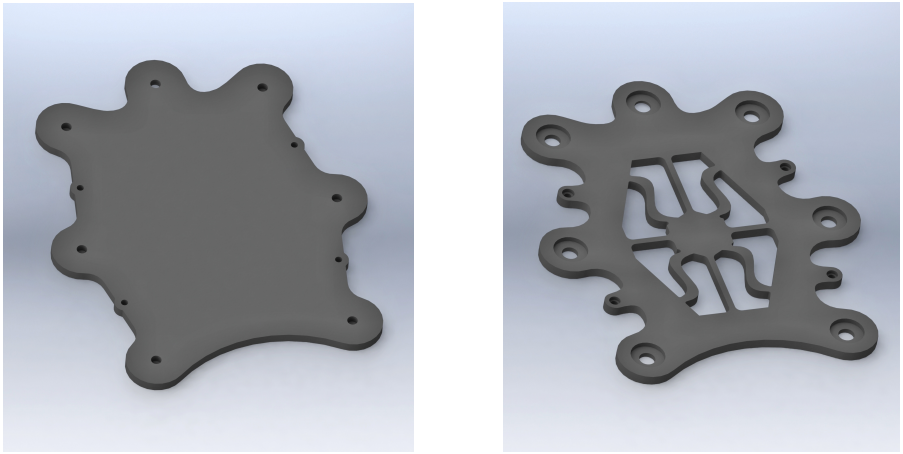
(a) Horní spojovací část (b) Trn

Obr. 2.2: Kyčelní kloub



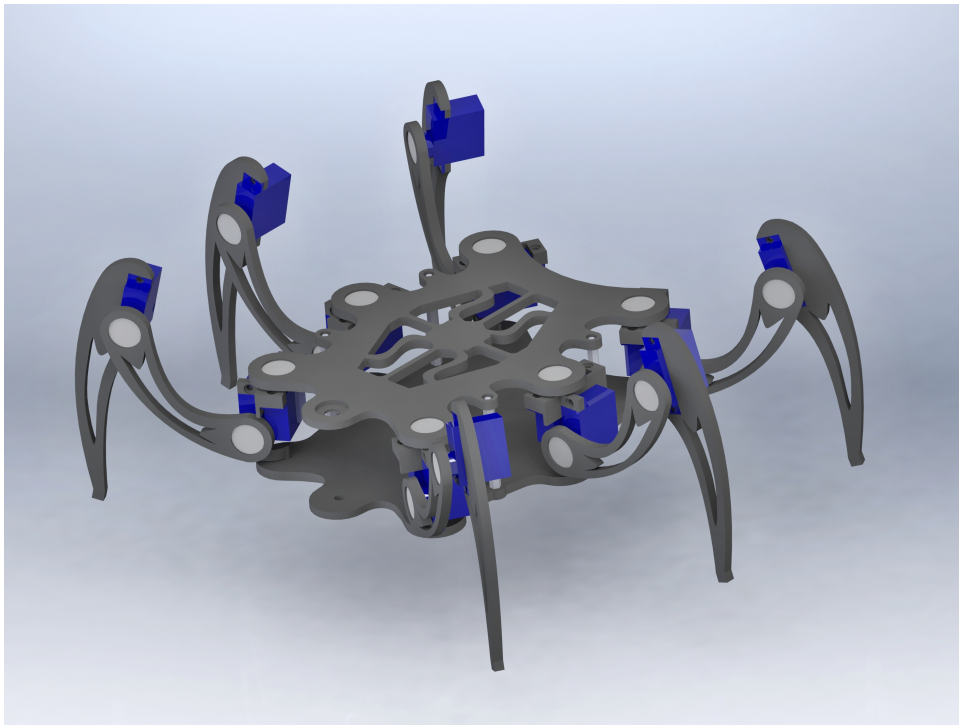
Obr. 2.3: Celá levá končetina

Tělo robota se skládá ze dvou částí, které jsou spojeny čtveřicí distančních sloupků. Mezi těmito dvěma částmi je prostor pro hardwarové vybavení robota a napájecí akumulátor. Horní a dolní část těla jsou znázorněny na obrázku 2.4. Končetiny jsou do horní části těla usazeny pomocí výstupních kotoučků akčních členů a do spodní čísti těla jsou fixovány pomocí trnů na spodní části kyčelních kloubů. Celá mechanická konstrukce je na obrázku 2.5.



(a) Spodní část těla (b) Vrchní část těla

Obr. 2.4: Části těla



Obr. 2.5: Celá mechanická konstrukce robota

2.2 Výroba mechanických částí

Jednotlivé díly byly navrženy tak, aby je bylo možné vyfrézovat z tvrzených PVC desek o tloušťce 5mm. Frézování probíhalo na fréze LPKF ProtoMat S100 a konkrétně se jedná o díly stehenní kosti, holenní kosti a spodní a horní části těla. Díly, které nebylo možné vyfrézovat z důvodu tvarové náročnosti, tj. díl spojující akční členy v kyčelním kloubu a trnu fixující nohu do těla, byly vytisknuty na 3D tiskárně Ultimaker z materiálu PLA.

Celá namodelovaná konstrukce je obsahem přiloženého CD-R.

3

Akční členy

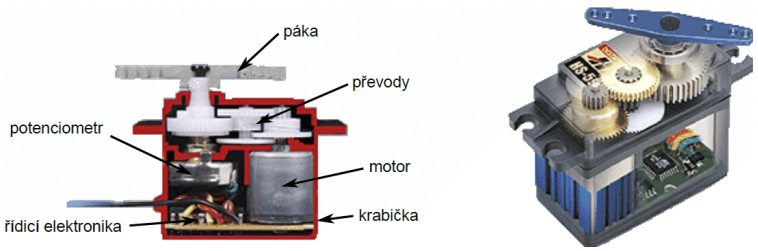
Požadavky na všechny akční členy jsou stejné a to přenášet točivý moment, malé rozměry, otáčení v rozsahu 0°-180°. Těmto požadavkům plně vyhovují modelářská serva. Vybrána byla serva od firmy Hitec konkrétně model HS-82MG. Jedná se o analogová rychlá mikroserva s kovovými převody.

Napájecí napětí [V]	4,8 - 6
Síla [Kg/cm]	2,8 - 3,4
Rychlost [sec/60°]	0,12 - 0,10
Rozměry [mm]	29,8 x 12 x 29,6
Hmotnost [g]	19
Kuličková ložiska	ne

Tab. 3.1: Parametry zvoleného modelářského serva Hitec HS-82MG

3.1 Modelářské servo

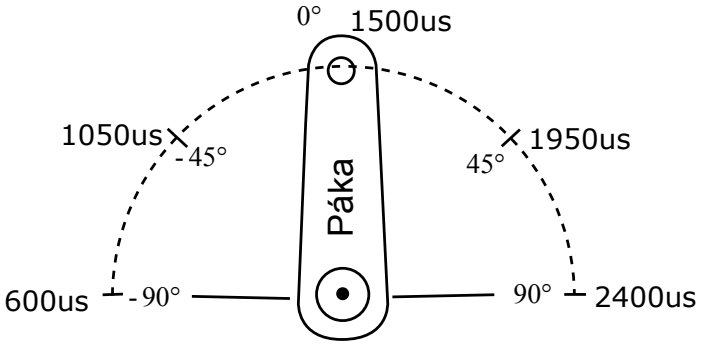
Modelářské servo je elektromechanický měnič, používaný v modelářství na přeměnu elektrické energie na mechanickou práci. Skládá se z malého stejnosměrného motoru, převodovky, řídicí elektroniky a krabičky. Vyrábějí se v různých velikostech od pikoserv o velikosti 1 x 2 x 1 cm až po velká maxiserva o velikosti 6 x 3 x 5 cm. Velikost serva přímo ovlivňuje jeho parametry, které jsou dány vlastnostmi stejnosměrného motoru. [3]



Obr. 3.1: Průřez modelářským servem |Převzato z [3]|

3.2 Řízení modelářského serva

Modelářské servo se řídí krátkými pulzy generovanými s frekvencí 50 Hz. Poloha natočení serva je přímo úměrná šířce pulzů. Standardně mají serva výchylku ±45°, ale většina serv zvládá úhel ±90°. Středová poloha odpovídá délce pulzu 1500 μs a koncové polohy 600 μs a 2400 μs. [3]



Obr. 3.2: Výchylka modelářského serva

4

Kinematika

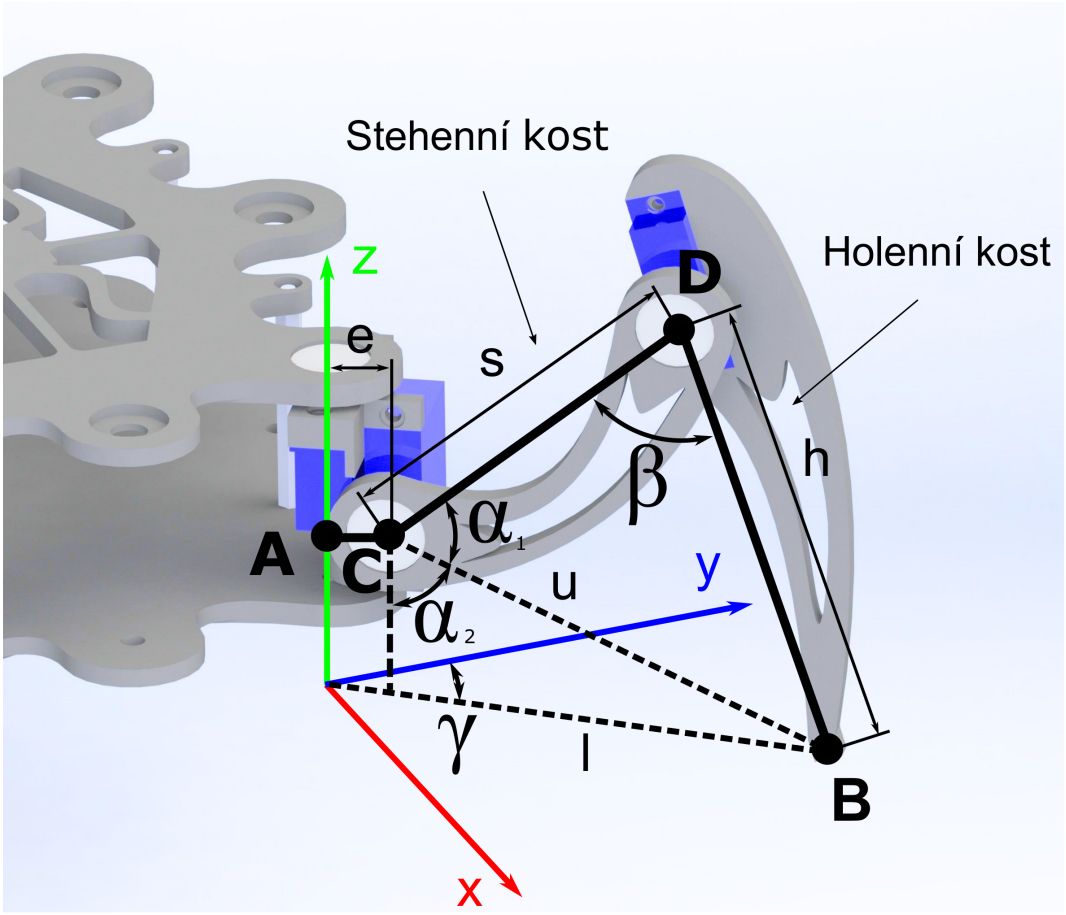
Pod slovem kinematika robotů se skrývá celý vědní obor, který se zabývá studií geometrie, pohybu a trajektorie, po kterých se pohybují jednotlivé body robota. V kinematice existují dvě základní úlohy. Přímá úloha kinematiky a inverzní úloha kinematiky.

Přímá úloha kinematiky slouží k získání souřadnic efektoru ze znalosti jednotlivých poloh kloubů. Tato úloha není nezbytně nutná pro samotný pohyb robota, ale využívá se při inicializaci a kalibraci, kdy jsou známy polohy kloubů a je nutné určit polohu efektoru a následně přemístit efektor do určité pozice. Tato úloha se považuje obecně za jednodušší.

Inverzní úloha kinematiky transformuje světové souřadnice efektoru na kloubové souřadnice. Tato úloha se vypočítává neustále při změně polohy efektoru a pro řízení robota je nezbytná. Uživateli tedy stačí znát polohu, kam se má efektor přemístit a pomocí této úlohy jsou vypočteny polohy jednotlivých kloubů.

4.1 Kinematika končetiny

Celý robot se skládá ze šesti totožných končetin, díky tomu je sestaven matematický model pouze pro jednu končetinu a výpočet se pro ostatní liší jen v umístění končetiny na těle a bodu, který určuje pozici chodidla končetiny. Obrázek 4.1 znázorňuje fyzické provedení končetiny a počítané parametry.



Obr. 4.1: Geometrické řešení inverzní kinematické úlohy

Jak již bylo zmíněno výše, výpočet spočívá v určení kloubových souřadnic při znalosti fyzických parametrů končetiny a dvou bodů v prostoru. Prvním bodem je bod $A[x_A; y_A; z_A]$, který je fixní a jeho souřadnice jsou dány rozměrem těla. Druhý bod $B[x_B; y_B; z_B]$ určuje místo, ve kterém se nachází chodidlo končetiny.

Řešení této úlohy se rozpadá na dvě nezávislé části, které lze počítat odděleně. První částí je určení úhlu γ. Výpočet je nezávislý na souřadnici z a řešení probíhá pouze v rovině definované osami x a y.

$$\gamma = \arctan\left(\frac{x_B - x_A}{y_B - y_A}\right) \tag{4.1}$$

Druhá část úlohy spočívá v určení úhlu α a β. Úhel α je součtem úhlu α_1 a α_2. V prvním kroku je vypočtena délka úsečky l pomocí Pythagorovy věty. Jedná se o vzdálenost mezi bodem C a B při průmětu do roviny, která je daná osami x a y.

$$l = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} - e \tag{4.2}$$

Následně je vypočtena délka úsečky u, což je přímá vzdálenost mezi body C a B v prostoru.

$$u = \sqrt{l^2 + z_A^2} \tag{4.3}$$

Ze znalosti délek úseček l a u lze spočítat úhel α_2, jelikož se jedná o pravoúhlý trojúhelník.

$$\alpha_2 = \arccos\left(\frac{z_A}{u}\right) \tag{4.4}$$

Pro výpočet úhlu α_1 je na trojúhelník BCD aplikována kosinová věta.

$$\alpha_1 = \arccos\left(\frac{s^2 + u^2 - h^2}{2us}\right) \tag{4.5}$$

Součtem úhlu α_1 a α_2 je získán úhel α.

$$\alpha = \alpha_1 + \alpha_2 \tag{4.6}$$

Poslední úhel β je opět určen pomocí kosinové věty z trojúhelníka BCD.

$$\beta = \arccos\left(\frac{s^2 + h^2 - u^2}{2su}\right) \tag{4.7}$$

Tímto postupem jsou vypočteny všechny potřebné kloubové souřadnice pro jednoznačnou konfiguraci končetiny v prostoru. Tento postup je navržen s co největší univerzalitou, aby bylo možné všech šest končetin počítat v cyklu a jen měnit body A a B pro každou končetinu.

4.2 Kinematika těla

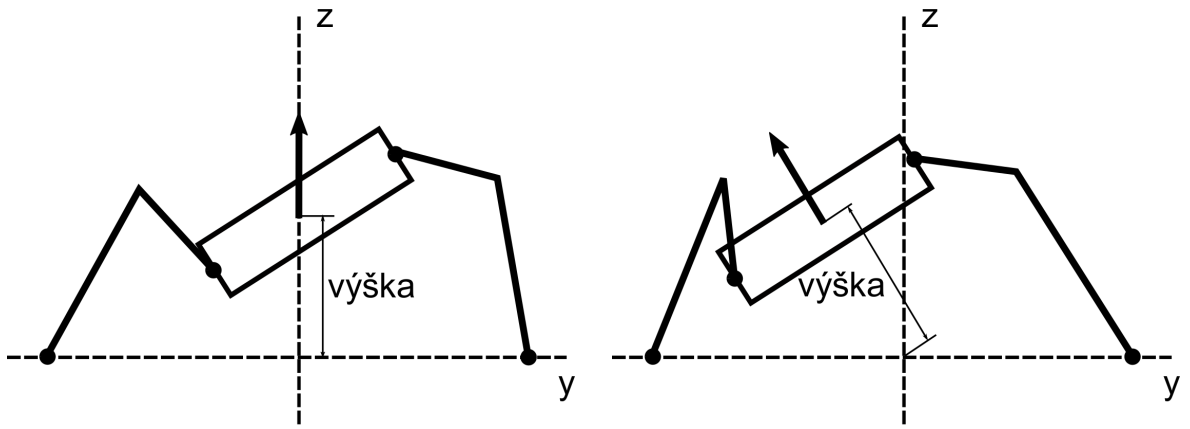
V předchozí kapitole je vysvětlen výpočet parametrů končetiny při znalosti bodu A a B. Tento výpočet tedy poskytuje výsledky pro jednotlivé akční členy, ale jelikož body A jednotlivých končetin jsou uloženy jako konstanty fyzického rozložení těla a body B leží v rovině, kde souřadnice z je nulová, je výpočet prováděn pro tělo rovnoběžně s rovinou definovanou osami x a y.

Kinematika těla spočívá v upravení buď bodů A nebo B tak, aby tělo bylo otočeno okolo osy x, y a z. Klasické řešení by bylo nechat body B zafixované na rovině definované osami x a y a body A otáčet v prostoru okolo všech tří os. Tento způsob přináší komplikace v podobě offsetu jednotlivých úhlů, které by nebyly jen výsledné vypočtené z kinematiky končetiny, ale musel by se korigovat o úhel samotného otočení, protože výpočet kinematiky končetiny je definován pro pevný souřadnicový systém. Druhým a zároveň použitým způsobem je nechat zafixované body A a v prostoru okolo všech tří os otáčet body B, tento způsob nepřináší žádné následné korekce úhlů a na výpočet je tudíž efektivnější.

Vedle rotace těla okolo os je na tělo implementován translační pohyb ve směru všech tří os. I při této úloze je opět použita úprava bodů B.

Pokud mají být translační pohyby těla rovnoběžné s osami a ne kolmé na povrch těla, musí být nejprve aplikován translační pohyb těla a následně otáčení okolo jednotlivých

os. Tuto závislost zobrazuje obrázek 4.2 pro translaci ve směru ose z a otočení okolo osy x.



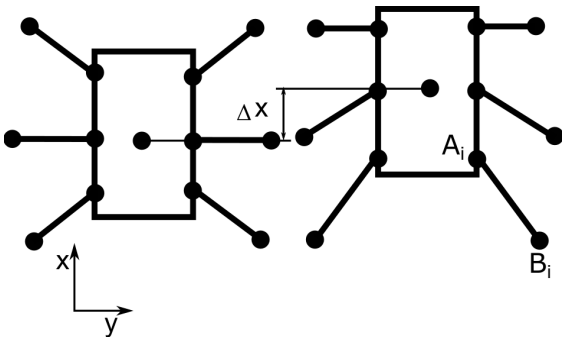
(a) Translace ve směru osy z (b) Translace ve směru kolmice na tělo

Obr. 4.2: Vlastnosti translačního pohybu těla ve směru osy z

Rozdíl v těchto dvou metodách spočívá v pořadí provádění úkonu rotačního pohybu a translačního pohybu těla. Pro tento model byla vybrána translace podle osy, kterou znázorňuje obrázek 4.2. Tato metoda je vhodná pro případ, kdy robot stojí na rovném povrchu a tím pádem změna výšky nad povrchem vyvolá pohyb rovnoběžný s osou z. Druhý způsob je vhodný v případě, že se robot pohybuje po nakloněné rovině a tělo se vyrovnává do vodorovné pozice. V takovém případě změna výšky způsobí pohyb který je kolmý na vodorovnou plochu.

4.2.1 Translační pohyb těla

Translační pohyby těla se provádí ve všech třech osách a je možné jakkoli je kombinovat. Translace ve směru osy x je znázorněna na obrázku 4.3. Jelikož je úloha řešena způsobem, kdy jsou upravovány body B, má vždy pohyb těla opačný směr a to i při rotaci těla. Na obrázcích je znázorněna situace, kdy robot pevně stojí na zemi a tudíž se nepohybují body B, ale body A a tím i celé tělo.

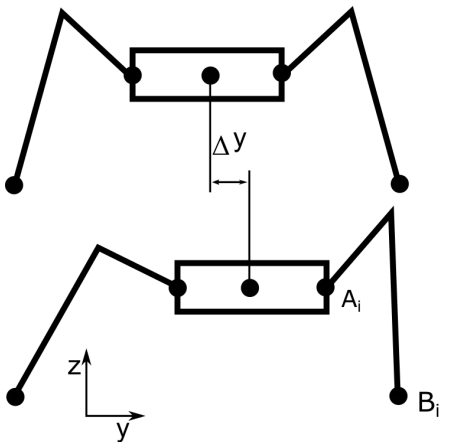


Obr. 4.3: Translační pohyb ve směru osy x

Výpočet probíhá pro všech šest bodů B nezávisle, kde x'_B je nová souřadnice bodu. Ostatní souřadnice zůstávají nezměněny.

$$x'_B = x_B - \Delta x \tag{4.8}$$

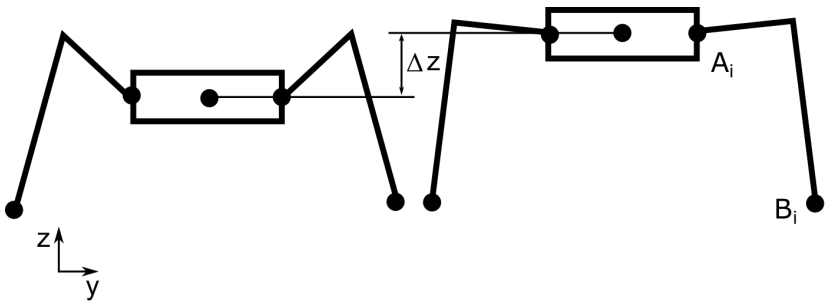
Translace ve směru osy y probíhá stejně jako v případě osy x, akorát jsou modifikovány souřadnice y bodu B:



Obr. 4.4: Translační pohyb ve směru osy y

$$y'_B = y_B - \Delta y \tag{4.9}$$

Translace v ose z:

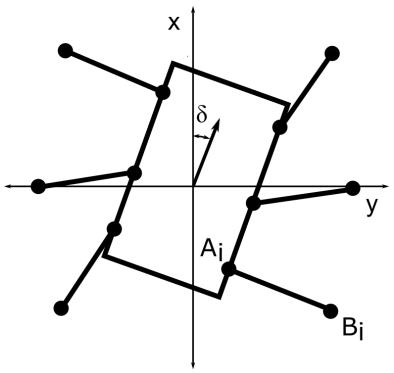


Obr. 4.5: Translační pohyb ve směru osy z

$$z'_B = z_B - \Delta z \tag{4.10}$$

4.2.2 Rotační pohyb těla

Rotace bodů B okolo os je náročnější než translace, jelikož se mění vždy dvě souřadnice. Na obrázku 4.6 je znázornění rotace okolo osy z. Při rotaci okolo osy z se mění souřadnice x_B a y_B.

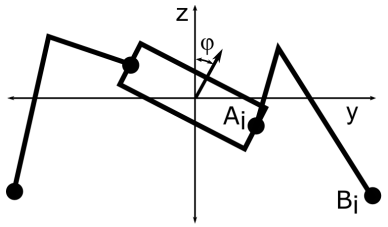


Obr. 4.6: Rotační pohyb okolo osy z

Rotace bodů B okolo osy z se provádí dle matice 4.11. Nové souřadnice jsou x'_B a y'_B. Souřadnice z'_B, jak vyplývá z matice, zůstává nezměněna.

$$\begin{bmatrix} x'_B \\ y'_B \\ z'_B \end{bmatrix} = \begin{bmatrix} \cos(\delta) & \sin(\delta) & 0 \\ -\sin(\delta) & \cos(\delta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \tag{4.11}$$

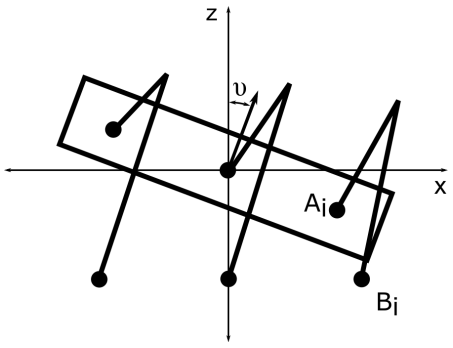
Rotace okolo osy x probíhá dle matice 4.12:



Obr. 4.7: Rotační pohyb okolo osy x

$$\begin{bmatrix} x'_B \\ y'_B \\ z'_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \tag{4.12}$$

Rotace těla okolo osy y:



Obr. 4.8: Rotační pohyb okolo osy y

$$\begin{bmatrix} x'_B \\ y'_B \\ z'_B \end{bmatrix} = \begin{bmatrix} \cos(\upsilon) & 0 & -\sin(\upsilon) \\ 0 & 1 & 0 \\ \sin(\upsilon) & 0 & \cos(\upsilon) \end{bmatrix} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \tag{4.13}$$

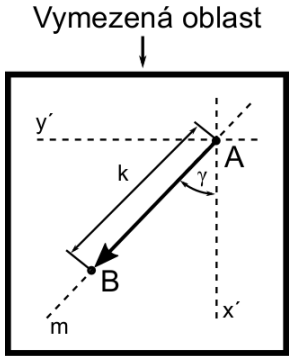
Aby bylo možné tělo natáčet ve všech osách kombinovaně, musí vždy následující rotace brát v potaz již provedenou rotaci a tudíž každý výpočet musí projít všemi třemi maticemi. Výše popsané pohyby těla, jak rotační tak translační, jsou všechny elementární pohyby, které lze s tělem provádět a jakýkoli složitější pohyb je kombinací těchto základních pohybů.

4.2.3 Kinematika chůze

Dnes je rozpracována spousta mechanismů chůze, které se vesměs zaměřují na deterministické střídání jednotlivých skupin končetin podle pevně daného diagramu. Nejjednodušší řešení je střídat vždy dvě krajní končetiny na jedné straně, spolu s jednou prostřední na straně druhé. Toto řešení přináší výhodu v zaručené stabilitě robota. Jiné teorie mají různě tvořené skupiny končetin, ale většinou se jedná o pevně daný diagram, jak končetinami pohybovat. Takovýto pohyb je příliš deterministický a robot při chůzi vypadá příliš mechanizovaně.

Pro tento model byla vyvinuta metoda, která končetiny střídá náhodně a v okamžiku, kdy je to nutné. Metoda spočívá v definování určitého prostoru pro každou končetinu, ve kterém se může pohybovat. Při chůzi jsou všechny končetiny posouvány proti směru požadované chůze. V okamžiku, kdy končetina překročí vymezený prostor, je končetina zvednuta nad povrch a přemístěna takovým způsobem, aby opětovný pohyb končetiny byl co nejdelší přes vymezenou oblast. Zvedání končetiny se řídí tabulkou, která specifikuje, které končetiny mohou být zvednuty společně, aby nebyla ohrožena stabilita robota.

Navržená metoda chůze se skládá ze dvou základních pohybů. Prvním pohybem je chůze přímá v jakémkoli směru. Pohyb spočívá v posouvání končetiny opačným směrem, než je směr chůze. Na obrázku 4.9 je znázorněn pohyb jedné končetiny z bodu A do bodu B. Rychlost chůze je dána délkou vektoru k. Směr chůze určuje úhel γ. Pro všechny končetiny je tento pohyb stejný.

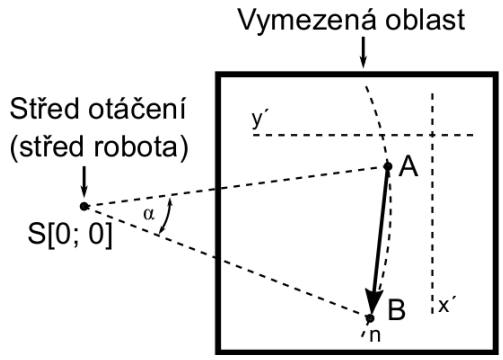


Obr. 4.9: Pohyb končetiny při chůzi v přímém směru

Polohu bodu B lze získat z následující matice:

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} x_A & \cos(\gamma) \\ y_A & \sin(\gamma) \end{bmatrix} \begin{bmatrix} 1 \\ -k \end{bmatrix} \tag{4.14}$$

Druhý pohyb implementuje otáčení robota na místě okolo osy z. Pohyb končetiny v tomto případě je po kružnic n, která má střed S v místě robota, okolo kterého se má otáčet, v našem případě ve středu těla. Rychlost otáčení je dána úhlem α. Obrázek 4.10 ukazuje opět elementární pohyb jedné končetiny.

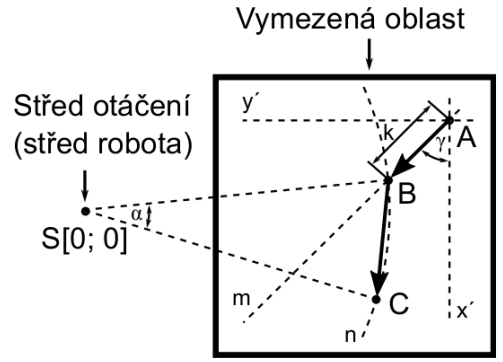


Obr. 4.10: Pohyb končetiny při otáčení

Opět lze odvodit matici, ze které lze vypočítat novou polohu bodu B.

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x_A \\ y_A \end{bmatrix} \tag{4.15}$$

Oba již zmíněné pohyby lze libovolně kombinovat, z čehož vyplývá takřka neomezený pohyb, kterým se může robot pohybovat. Kombinování spočívá v postupném aplikování nejprve přímého pohybu matice 4.15 a následně otáčivého pohybu matice 4.15. Tato situace je na obrázku 4.11. Nejprve je aplikován pohyb přímý z bodu A do bodu B a následně s bodem B je provedeno otočení okolo středu S do konečné polohy, kterou reprezentuje bod C.



Obr. 4.11: Kombinace pohybu přímého a otáčivého

5

Elektronické vybavení robota

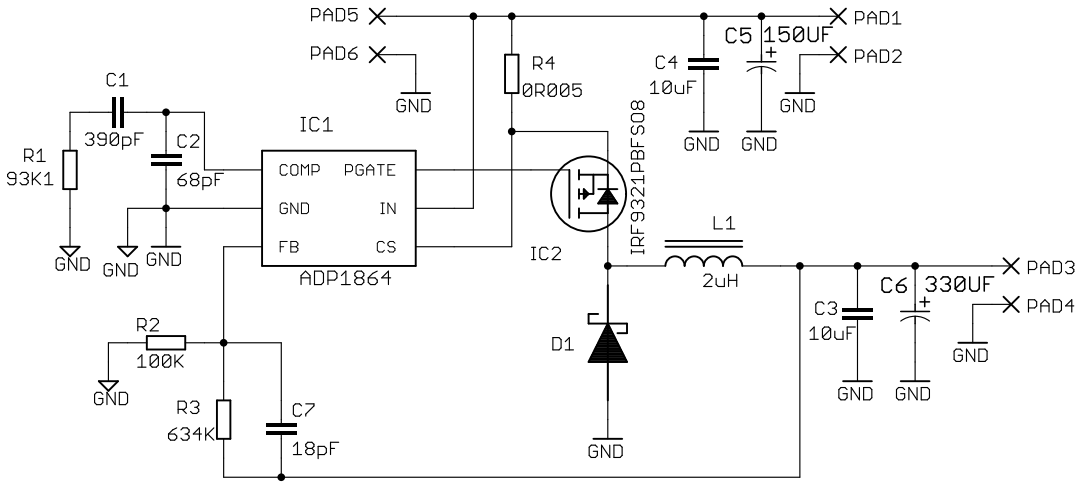
Veškeré elektronické vybavení robota je rozděleno do tří samostatných částí, kde každá z nich je implementována na vlastním plošném spoji. První částí je napájecí jednotka, která se stará především o napájení výkonových prvků, tj. servomotorů. Ostatní moduly si řeší vlastní napájení sami. Druhým modulem je řídicí jednotka osazená mikrokontrolérem, která se stará o veškeré výpočty týkající se kinematiky a generuje signály pro jednotlivá serva. Poslední modul je Bluetooth modul, který zprostředkovává komunikaci mezi ovladačem a robotem.

5.1 Napájecí jednotka

Napájení robota je značně variabilní, požadavky jsou od napájení robota z Li-Pol baterie o jmenovitém napětí 7,4 V, až po napájení ze stejnosměrného zdroje o napětí 12 V. Vybraná serva pracují v rozsahu napájecích napětí od 4,8 V do 6 V. Čím vyšším napětím jsou serva napájena, tím větší sílu mají, z toho důvodu bylo zvoleno maximální napájecí napětí 6 V. Proud, který musí být jednotka schopna dodat, je až 10 A v případě, že jsou všechna serva v pohybu. Tato hodnota je dosahována jen výjimečně a většinu času odebírají serva proud do 5 A, nicméně jednotka musí být dimenzována pro špičkový odběr až 10 A. Z uvedených požadavků vyplývá, že se jedná o měnič typu DC-DC Step-Down.

Konkrétně byl vybrán kontrolér od výrobce Analog Devices ADP1864. Jedná se o asynchronní buck regulátor, který pracuje s konstantní frekvencí v proudovém modu. Kontrolér obsahuje driver pro spínací tranzistor typu MOSFET s indukovaným P-kanálem. Provozní napětí na vstupu jsou v rozsahu od 3,15 V do 14 V. Maximální dosažitelné výstupní napětí je nižší o 0,8 V než vstupní napětí. Kontrolér pracuje s pevně danou spínací frekvencí 580 kHz a dosahuje efektivity až 94%. Na obrázku 5.1 je schéma celého obvodu.

Vstupní svorky obvodu jsou PAD1 a PAD2. Hned za svorkami se nachází vstupní kondenzátory. První kondenzátor je elektrolytický o kapacitě 150 μF a druhý keramický s kapacitou 10 μF. Oba kondenzátory poskytují cestu s nízkou impedancí pro pulzní proud, odebírající externí MOSFET tranzistor. Spínací frekvence je pevně nastavena na 580 kHz, z toho důvodu je použit keramický kondenzátor, který má na takto vysokých frekvencích



Obr. 5.1: Schéma napájecí jednotky

vlastní rezonanční kmitočet. Svorky PAD5 a PAD6 slouží pro napájení řídicí jednotky, která má vlastní lineární stabilizátor, proto jsou tyto svorky jen čistě průchozí ze vstupu na výstup. Výstupní svorky obvodu jsou PAD3 a PAD4.

Kontrolér ADP1864 je asynchronní buck regulátor, z čehož vyplývá použití diody D1. Pro správnou funkci obvodu se musí jednat o rychlou diodu typu Schottky. Důležité je diodu správně dimenzovat výkonově. Nejhorší případ, který pro diodu může nastat, je za předpokladu, kdy je vstupní napětí maximální tedy 12 V. V tento okamžik je Duty Cycle=0,52 a přes diodu se vrací proud nejdelší dobu. V takovémto případě je efektivní hodnota proudu při maximálním zatížení 4,8 A. Vybrána byla dioda od výrobce ON Semiconductor typ MBRB20H100CTT4G v pouzdře D2PAK.

Další kritickou součástkou, která má vliv na správnou funkci obvodu, je pracovní indukčnost L1. Velice důležitý parametr tlumivky, který má následně vliv na účinnost celého měniče, je její sériový odpor, který by měl být co možná nejnižší. Dalším parametrem je indukčnost tlumivky. Z požadovaných parametrů vyplývá použití tlumivky s indukčností minimálně 1,95 μH. Posledním požadavkem je tlumivku správně proudově dimenzovat, aby nedocházelo k přesycování jádra tlumivky. Zvolena byla tlumivka od firmy Coilcraft typ SER2010-202MLD s indukčností 2 μH, sérovým odporem 0,9 mΩ a saturačním proudem 27 A.

Externí spínací tranzistor typu MOSFET s indukovaným P-kanálem. Tento tranzistor musí splňovat následující požadavky. Jelikož se jedná o tranzistor s indukovaným kanálem typu P, je tento tranzistor otevírán stažením řídicí elektrody na potenciál země, jak vyplývá ze schéma, které je v katalogovém list [5]. Povolené napětí U_{GS} tranzistoru musí být tedy zápornější než -12 V. Zároveň stejně jako pracovní indukčnost má vliv na účinnost odpor kanálu při sepnutém stavu a stejně jako indukčnost je nutné tranzistor dostatečně výkonově dimenzovat. Maximální efektivní hodnota proudu tekoucího přes tranzistor je 9,309 A. Poslední hodnota je povolené napětí U_{DS} tranzistoru, které musí být opět zápornější než -12 V. Zvolen byl tranzistor od firmy International Rectifier typ IRF9321PbF.

Kondenzátor C3 a C6 tvoří výstupní filtr. Elektrolytický kondenzátor C6 musí mít co nejmenší sériový odpor a musí být dimenzován na efektivní hodnotu proudu 0,809 A.

Rezistory R2 a R3 tvoří zpětnou vazbu, která je uvnitř kontroléru porovnána s referenčním zdrojem o napětí 0,8 V. U těchto rezistorů je kladen důraz na přesnost, a proto byly vybrány rezistory s přesností 1%.

Kombinace kondenzátorů C1, C2 a rezistor R1 slouží pro kompenzaci chybového zesilovače a zároveň kondenzátor C2 redukuje spínací jitter. Poslední součástí je rezistor R4, pomocí kterého je tvořena nadproudová ochrana, kdy při překročení napětí 125 mV na rezistoru je zastaveno generování spínacího signálu pro spínací tranzistor.

5.2 Řídicí jednotka

Jsou dvě varianty jak implementovat strukturu robota. První možnost je řídicí jednotkou pouze generovat signály pro akční členy a celý kinematický výpočet provádět mimo robota a do řídící jednotky přenášet pouze jednotlivá data pro akční členy. Tato varianta přináší výhodu v jednodušší řídicí jednotce, která nemusí mít tak velký výpočetní výkon. Na druhou stranu nevýhody, které toto řešení přináší znatelně převažují. Absencí externího výpočetního systému (PC, smartphone) je znemožněna funkce celého robota a celé řešení je značně neuniverzální. Pro každou platformu, ze které by měl jít robot ovládat, by bylo nutné znovu implementovat celé chování robota od začátku.

Druhá varianta spočívá v implementaci celého kinematického výpočtu "na palubě robota". Nevýhoda v podobě výkonnější řídicí jednotky je při dnešních cenách výkonných mikrokontroléru a dostupnosti zanedbatelná. Velká výhoda tohoto řešení je v zachování funkčnosti i při ztrátě konektivity s ovládáním a zároveň možnosti implementace autonomního chování. Implementace ovládání pro více platforem bude také znatelně jednodušší, jelikož se posílají jen data o pohybu robota a následný kinematický výpočet je už řešen v řídící jednotce. Nesporné výhody, které toto řešení přináší, rozhodly o implementaci tohoto řešení.

5.2.1 Hardware

Po hardwarové stránce je řídicí jednotka velice jednoduchá. Pro napájení je použit lineární stabilizátor MC33269 s výstupním napětím 3,3 V v katalogovém zapojení. Kromě napájecí části je jednotka osazena mikrokontrolérem, který se stará o veškeré výpočty a generování signálů pro jednotlivé akční členy. Výstupem řídicí jednotky jsou konektory obsahující napájení a signál pro akční členy.

Při výběru mikrokontroléru byl kladen důraz na výpočetní výkon, cenu a dostupnost jak součástky, tak vývojového prostředí a programovacích nástrojů. Volba padla na dnes velice rozšířenou architekturu ARM. Tuto architekturu procesoru dnes vyrábí nespočet firem, nicméně firma STMicroelectronics nabízí velice levné a dostupné vývojové pro-

středky v podobě Discovery kitů. Pomocí tohoto kitu lze programovat mikrokontrolér, který je osazen přímo na kitu, ale také lze po odstranění propojek programovat externí mikrokontrolé přes rozhraní SWD nebo JTAG. Samozřejmostí je i možnost přes obě rozhraní externí mikrokontrolér ladit. Z tohoto důvodu byl vybrán mikrokontrolér od již zmiňované firmy konkrétně typ STM32F103C8. Jedná se o 32-bitový RISC mikrokontrolér s jádrem Cortex^{TM}-M3. Maximální taktovací frekvence jádra je 72 MHz a výkon, který dosahuje, je 1.25 DMIPS/MHz. K jádru je připojena FLASH paměť o kapacitě 64 KB pro program a pro data je připravena SRAM o kapacitě 20 KB. Periferie, kterými mikrokontrolér disponuje a jsou v této práci použity: 16-ti bitový Timer/Counter, USART, GPOI, SysTick.

V příloze na obrázku B.1 je celé schéma řídicí jednotky, většina součástí slouží pro korektní funkci mikrokontroléru a jsou doporučeny výrobcem. Sada blokovacích kondenzátorů 100 nF, které jsou umístěny hned u napájecího pinu na každé straně obvodu. Tlumivka s kondenzátorem tvořící filtr typu dolní propust, pro napájení analogových periferií v tomto případě jen fázového závěsu, který generuje kmitočty pro různé bloky mikrokontroléru. Krystalový rezonátor o frekvenci 8 MHz. Konektory pro akční členy, které jsou přivedeny přímo na brány mikrokontroléru. Svorky PAD1 a PAD2, přes které je přivedeno napájecí napětí pro akční členy. Konektor SW1 slouží pro programování mikrokontroléru přes SWD rozhraní a konektor SW3 pro komunikaci s bluetooth modulem přes rozhraní USART.

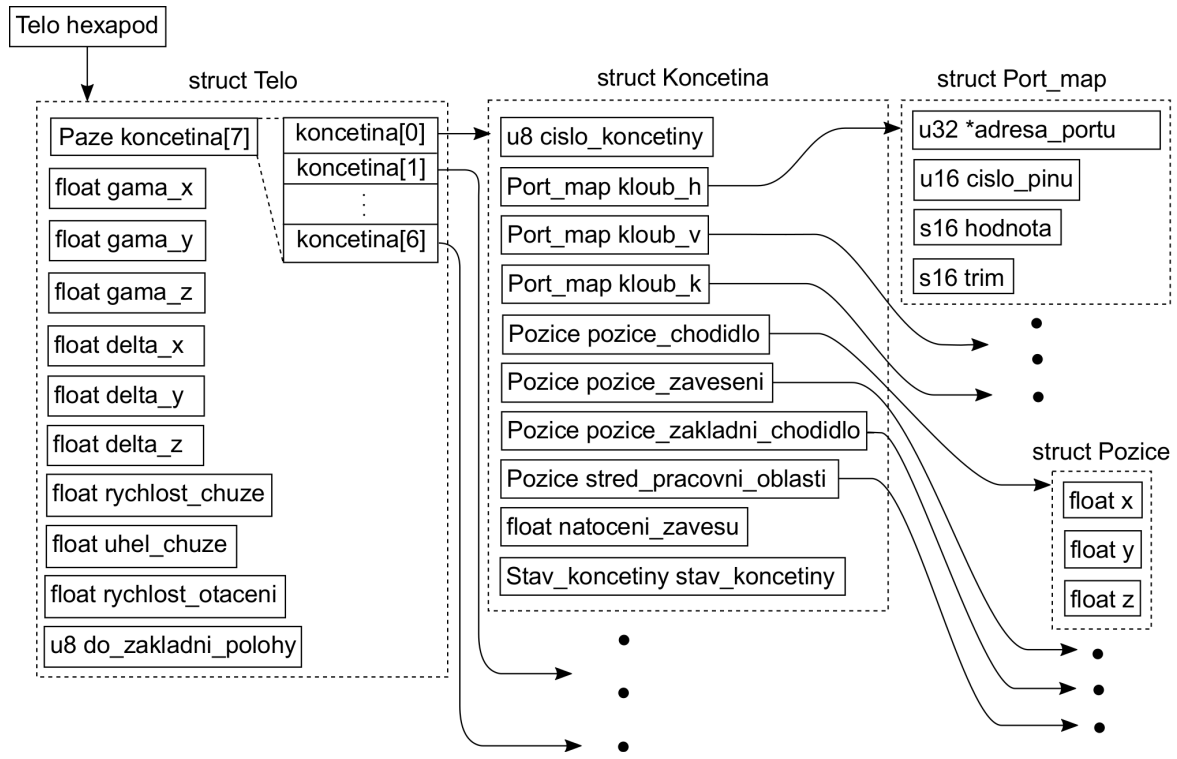
5.2.2 Firmware

Pro vývoj firmwaru bylo použito vývojové prostředí od výrobce KEIL μVision V5.12.0.0 [6] ve verzi Lite Edition, která generuje kód do maximální velikosti 32 KB. Jazyk, ve kterém je firmware napsán, je ANSI C. Kompletní zdrojový kód firmwaru se nachází na přiloženém CD-R

5.2.2.1 Datová struktura popisující robota

Velké množství fyzických i elektrických parametrů robota znemožňuje používání proměnných a konstant jen rozdělených pomocí jmen. Proto byla vytvořena datová struktura, která popisuje celého robota. Struktura byla navržena s ohledem na co největší mechaničnost při celém výpočtu. Na obrázku 5.2 je datová struktura, která popisuje šestinohou robotickou platformu.

První úroveň je tvořena strukturou *Telo*, obsahující data, která jsou společná pro celého robota. Proměnné *gama_x*, *gama_y*, *gama_z* reprezentují otočení těla robota okolo jednotlivých os a vyjadřují tedy úhly. Další proměnné *delta_x*, *delta_y* a *delta_z* mají podobný význam jako předchozí proměnné, jen se jedná o translační pohyb těla ve směru všech os. Pro nastavení vlastností chůze slouží proměnné *rychlost_chuze*, *uhel_chuze* a *rychlost_otaceni*. Pomocí proměnné *do_zakladni_polohy* se nastavuje požadavek na uve-



Obr. 5.2: Datová struktura popisující robota

dení jednotlivých končetin robota do základní polohy. Posledním prvkem této struktury je pole o sedmi prvcích, obsahující struktury *Koncetina*. Díky tomu, že jsou jednotlivé končetiny v poli, je možné kinematický výpočet přes všechny končetiny provádět efektivně v cyklu.

Druhá úroveň obsahuje parametry, které se vážou vždy jen k jedné končetině a tudíž těchto struktur je v paměti sedm. Sedmá struktura slouží pro ovládání hlavy. Na obrázku je tato skutečnost vyjádřena třemi tečkami. Pro identifikaci, s jakou končetinou se pracuje, slouží proměnná *cislo_koncetiny*. Tato proměnná se zdá být nadbytečná, jelikož pracujeme s polem a index do pole známe, nicméně v kinematickém výpočtu si funkce předávají ukazatel, který ukazuje jen na jeden prvek celého pole *koncetina*, pomocí kterého se lze dostat na všechny parametry dané končetiny. Tímto mechanismem je docíleno i toho, že upravované parametry ve struktuře se projeví i z venku a není do funkce struktura kopírována a měněna jen lokálně, se žádným výsledkem na strukturu *hexapod*.

Následující tři proměnné *kloub_h*, *kloub_v*, *kloub_k* jsou struktury typu *Port_map*. Každá z těchto struktur popisuje jeden akční člen na končetině. Proměnná $*adresa\_portu$ je ukazatel do registru $BSRR(BitSet/ResetRegistr)$ jednoho ze dvou použitých portů. Pro port $GPIOA$ je v proměnné adresa 0x40010810 a pro port $GPIOB$ 0x40010C10. Tímto je tedy definováno, jaké akční členy jsou připojeny k jakému portu. Pro definování jednotlivého pinu, na který je akční člen připojen, slouží proměnná *cislo_pinu*. Pro jednoduchost a čitelnost kódu jsou definovány konstanty $P0$ až $P15$, které mají nastavený jediný bit, který odpovídá pinu, jež definují. Pomocí těchto proměnných jsou nastavovány

a mazány bity opět efektivně v univerzálním cyklu, i když se střídají brány i piny. Poslední dvě proměnné souvisí přímo s generováním signálu pro akční člen, kde proměnná *hodnota* je přímo vypočtená hodnota pro natočení akčního členu a *trim* reprezentuje jemné doladění polohy a kalibraci.

Proměnné *pozice_chodidlo* a *pozice_zaveseni* jsou struktury typu *Pozice*, která obsahuje souřadnice x, y a z. Tyto dvě struktury reprezentují body A a B z obrázku 4.1. Jsou to dva body, pomocí nichž je možné vypočítat všechny polohy akčních členu pro danou končetinu. Další proměnnou, která opět určuje bod v prostoru, je *pozice_zakladni_chodidlo*. V tomto bodě jsou končetiny robota v základní poloze, která nastává při startu robota, nebo v případě přepínání z módu chůze do módu pohybu těla. Oblast pro pohyb končetiny vymezuje proměnná *stred_pracovni_oblasti*, která definuje bod ležící v jejím středu. Pro zohlednění prostorového rozložení končetin slouží proměnná *natoceni_zavesu*, což je úhel, o který jsou rohové končetiny mechanicky otočeny oproti prostředním dvěma.

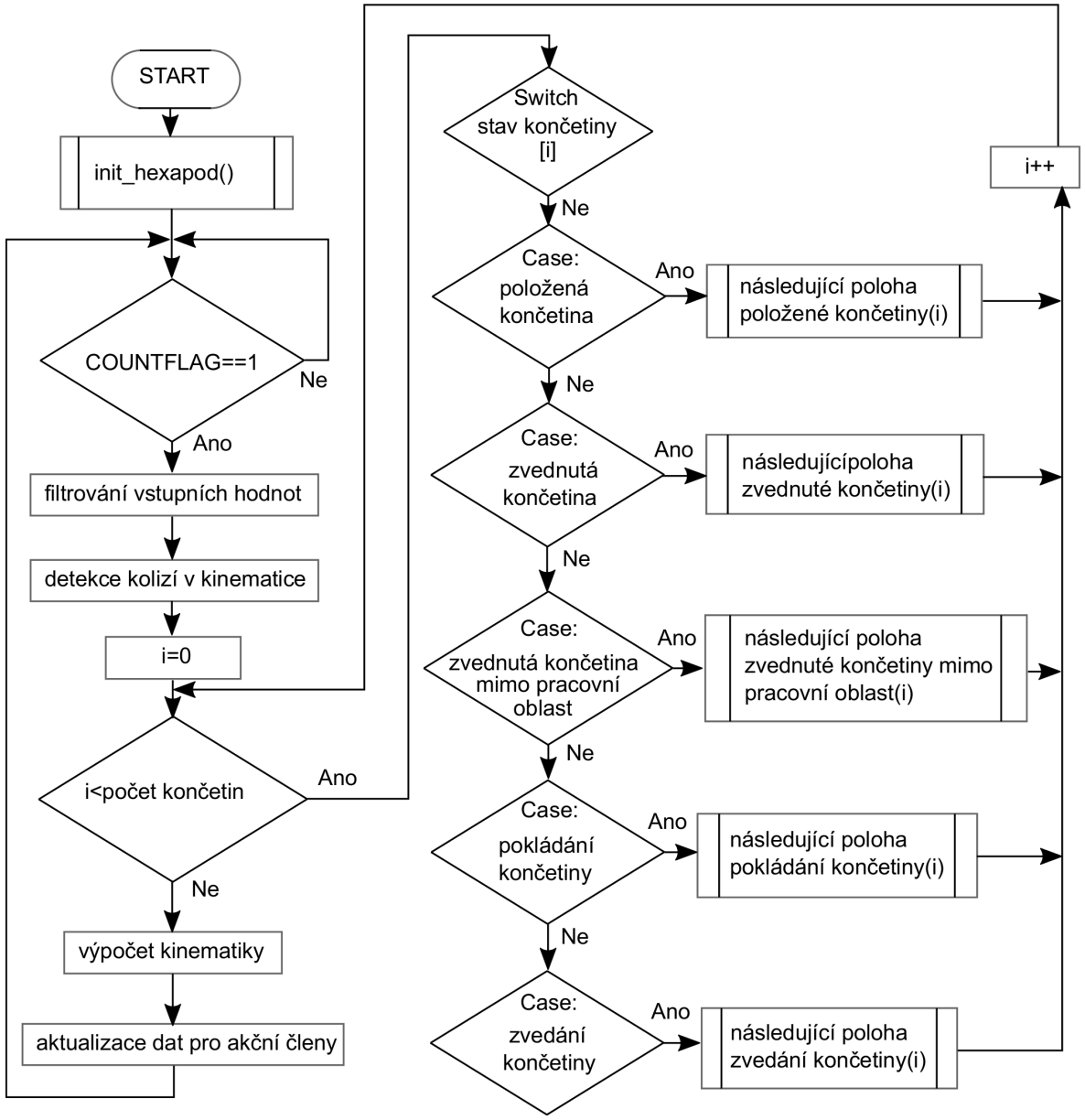
Poslední proměnnou v této struktuře je *stav_koncetiny*. Tento typ je definován jako *enum* a může nabývat hodnot *polozena_koncetina*, *zvednuta_koncetina*, *pokladani_koncetiny*, *zvedani_koncetiny*, *zvednuta_koncetina_mimo_pracovni_oblast*. Tato proměnná odráží stav, ve kterém se končetina nachází a pomocí této proměnné je tvořen stavový automat, kterým je implementována chůze robota. Celá struktura zabere v datové paměti prostor o velikosti 625 bytu.

5.2.2.2 Vývojový diagram

Hlavní část programu je velice jednoduchá. Po připojení napájeni je zavolána funkce *init_hexapod*(), ve které je inicializace celého mikrokontroléru. Tato funkce postupně inicializuje tyto periferie a oblasti:

- Naplnění datové struktury, která popisuje robota
- Zakázání JTAG a piny obsazené touto periferií uvolnit pro GPOI
- GPIO pro připojení akčních členů
- USART pro komunikace s Bluetooth jednotkou
- SysTick pro časové opakování hlavní smyčky programu
- Timer2 pro generování signálů pro akční členy
- Nastavení priority přerušení od jednotlivých periferií a jejich povolení
- Odstartování SysTick a Timer2

Po nastavení mikrokontroléru se běh programu přesune do hlavní smyčky. V hlavní smyčce se vyčkává na podtečení čítače SysTick, který v tomto okamžiku nastaví bit $COUNTFLAG$ do logické „1“. Tato perioda je nastavena na dobu opakování 20 ms,



Obr. 5.3: Vývojový diagram hlavního programu

což vyplývá z použití analogových serv. Tato serva se řídí pulzy, které jsou generovány s periodou 20 ms. Z toho důvodu je vhodné počítat kinematický výpočet o této periodě. Na obrázku 5.3 je vývojový diagram hlavní smyčky programu.

Po průchodu do nekonečné smyčky o periodě 20 ms je provedeno filtrování vstupních hodnot. Data, která produkuje akcelerační snímač, jsou zatížena šumem, a proto je nezbytné data filtrovat. Filtrování dat je také z důvodu hezčí pohybové odezvy robota na skokovou změnu polohy, kdy se tělo robota do požadované polohy přesune s exponenciální odezvou. Jedná se o filtry typu dolní propust implementované jako aperiodický člen s časovou konstantou 0,5 s. Filtry jsou implementovány v pevné řádové čárce a každá veličina týkající se pohybu těla má vlastní filtr.

Po vyfiltrování hodnot následuje detekce kolizí v kinematice. Detekce zabraňuje stavům při pohybu těla, které není možné dosáhnout, nebo by mohly poškodit mechanickou konstrukci robota. Pokud nastane takováto kolizní situace, robot zůstane na poslední platné pozici před kolizním stavem a vstupní filtry jsou zablokovány do té doby, než se robot dostane do bezkolizního stavu a je jedno, jakým vstupním parametrem se tak stane.

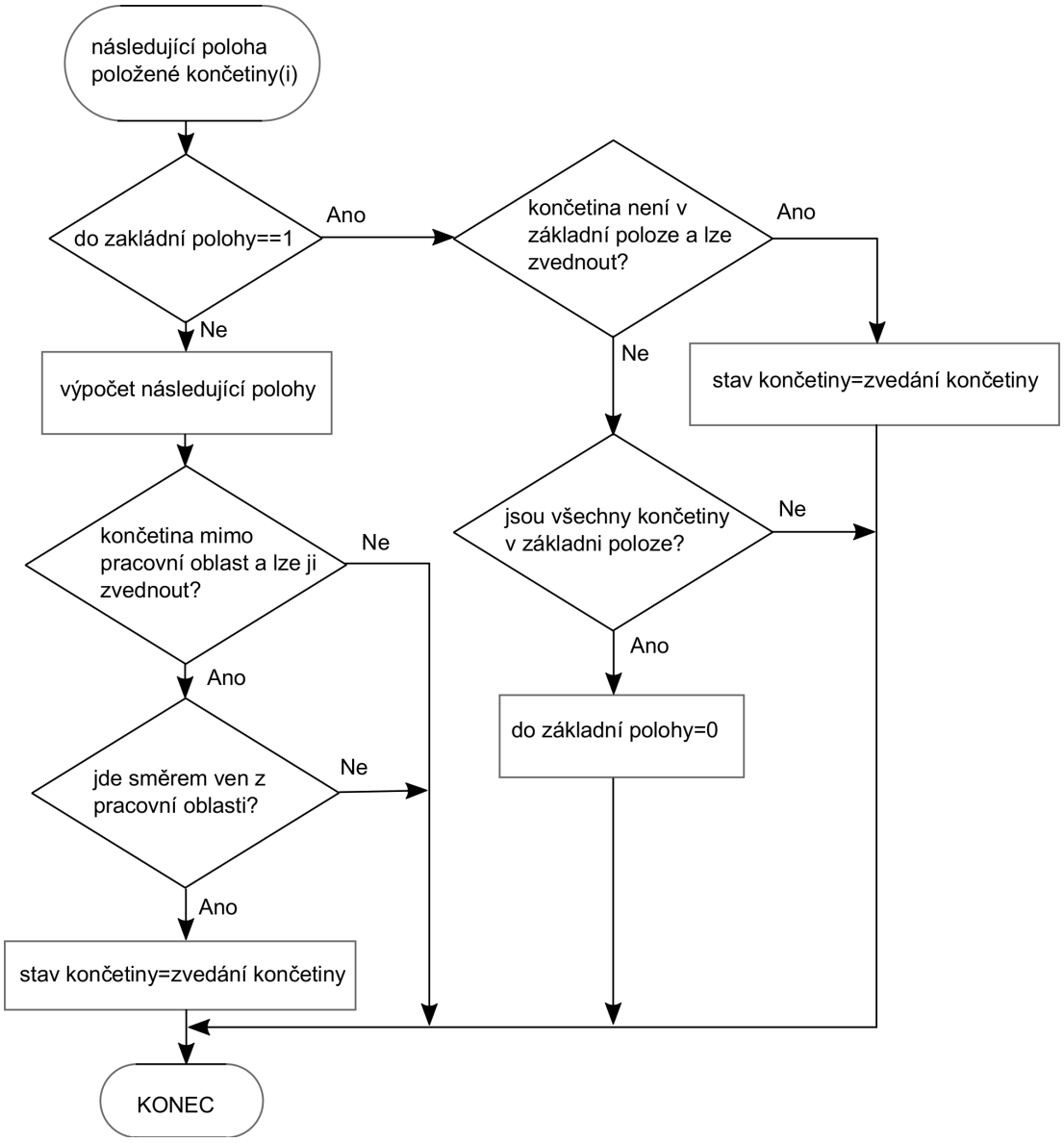
Následující část programu je cyklus, který je prováděn přes všechny končetiny robota. Tímto cyklem je tvořena chůze robota a je implementována jako stavový automat pro každou končetinu. Každá končetina má ve struktuře proměnnou *stav_koncetiny*, která odráží stav, ve kterém se končetina nachází a pomocí ní jsou přes konstrukci switch-case volány příslušné metody, které tvoří jednotlivé stavy automatu. Mezi stavy se přechází právě pomocí proměnné *stav_koncetiny*, která je modifikována uvnitř metod. Jednotlivé stavy budou rozebrány níže, i s uvedením jejich vývojových diagramů.

Po průchodu všemi končetinami přechází program k samotnému kinematickému výpočtu celého robota, kterým se zabývá kapitola 4. Po výpočtu všech úhlů pro akční členy jsou tyto hodnoty upraveny a uloženy do proměnné *hodnota* ve struktuře každého akčního členu. Odtud si hodnoty převezme řízení akčních členů, které je implementováno v přerušení a bude podrobně popsáno níže.

- **položená končetina**

Vývojový diagram tohoto stavu je na obrázku 5.4. V tomto stavu se nacházejí všechny končetiny po inicializaci robota. Při vstupu do tohoto stavu se program větví dle proměnní *do_zakladni_polohy*. Pokud je tato proměnná rovná jedné, je požadavek všechny končetiny přemístit do základní polohy. V případě, kdy končetina není v základní poloze a končetinu lze zvednout, je nastaven stav na zvedání končetiny. Zvednout končetinu lze pouze za předpokladu, že není narušena stabilita robota. Stabilní stavy jsou realizovány tabulkou, ve které jsou povolené kombinace končetin, které mohou být zvednuty současně. Pokud tedy není tato podmínka splněna, končetina vyčkává do doby, než se nepřesunou do základní polohy ostatní končetiny, které tuto končetinu blokují. Po přesunutí všech končetin do základní polohy, je proměnná *do_zakladni_polohy* vynulována, čímž je indikováno dokončení této operace.

Za předpokladu, že není nastaven požadavek na přesunutí končetin do základní polohy, je vypočítána následující poloha položené končetiny. Dále je vypočtená poloha končetiny testována, zda se pozice nachází v pracovní oblasti. Pokud se pozice nachází v pracovní oblasti, je pozice končetiny aktualizovaná a metoda ukončena. V případě, kdy končetina opustila pracovní oblast, je stav končetiny změněn na zvedání končetiny, ale pouze za předpokladu, že následující poloha končetiny je směrem ven z pracovní oblasti. Pokud je směrem do pracovní oblasti, je končetina ponechána v tomto stavu. Tato situace může nastat v případě, kdy končetina je blokována ostatními končetinami a dostane se ven z pracovní oblasti a zároveň je změněn směr chůze. V takovémto případě by bylo zbytečné končetinu přesouvat na kraj pracovní oblasti, proto se ponechá mimo pracovní oblast

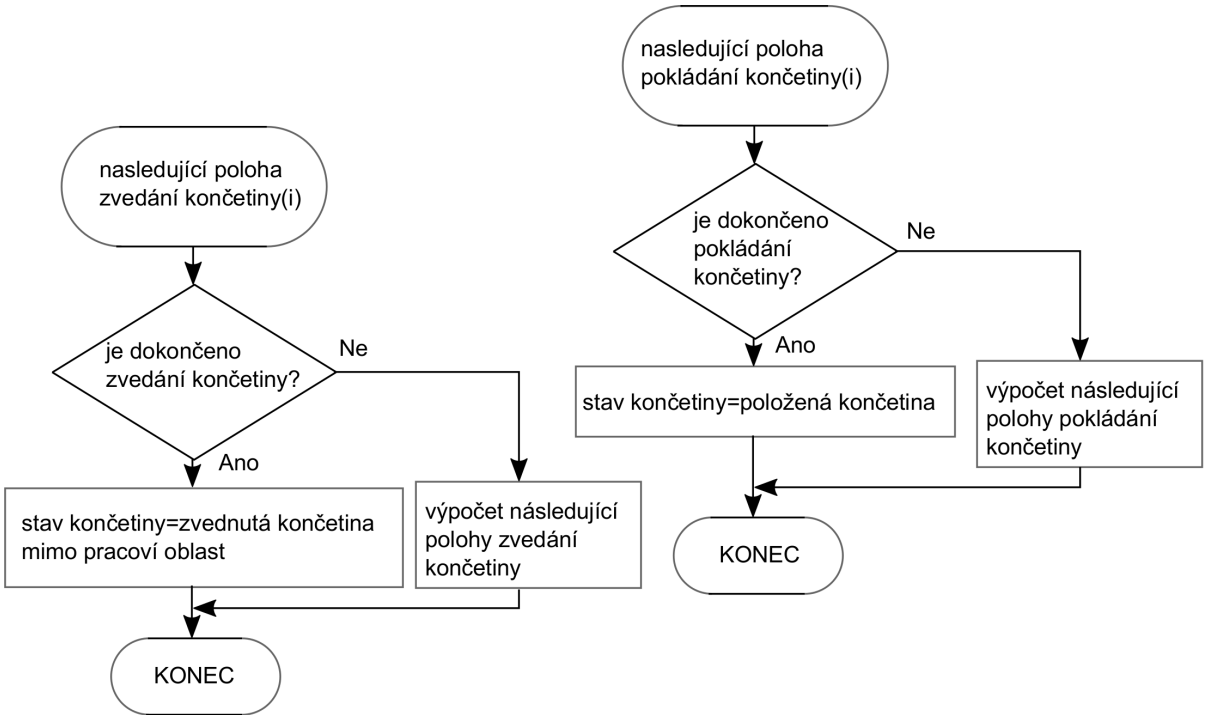


Obr. 5.4: Vývojový diagram pro stav položená končetina

položená a končetina se sama dostane do pracovní oblasti v položeném stavu. Kdyby se během této situace opět změnil směr chůze, může být končetina okamžitě přesunuta do optimální polohy pro následný pohyb. Po návratu ze stavového automatu, je vypočtena inverzní úloha kinematiky a jsou aktualizovaná data pro generování signálu pro akční členy.

- **zvedání končetiny a pokládání končetiny**

Ze stavu položená končetina se lze přesunout pouze do stavu zvedání končetiny. Tento stav reprezentuje vývojový digram na obrázku 5.5, kde je společně s vývojovým diagramem pro stav pokládání končetiny. Oba stavy jsou velice podobné a liší se jen tím, jakým směrem se končetina pohybuje a do jakého následného stavu končetina přechází.

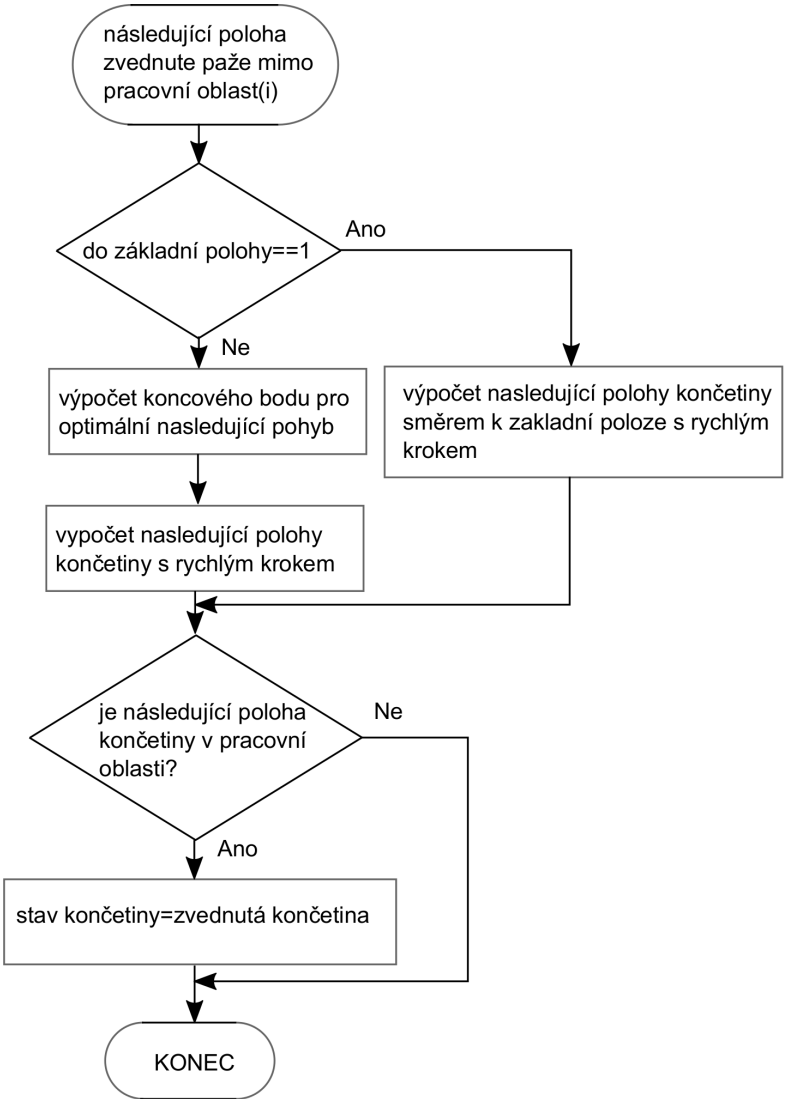


Obr. 5.5: Vývojový diagram pro stav pokládaní končetiny a zvedáni končetiny

- **zvednutá končetina mimo pracovní oblast**

Po dokončení zvedání končetiny se přechází do tohoto stavu (obr. 5.6). V tomto stavu se končetina pohybuje ve zvednutém stavu a konstantní rychlostí, která je větší než pohyb položené končetiny. Pokud je požadavek na přemístění končetin do základní polohy aktivní, je naplánován pohyb končetiny směrem k této poloze. Tato poloha ale v tomto stavu není dosažena. V okamžiku, kdy se končetina dostane do pracovního prostoru, přechází na stav zvednutá končetina.

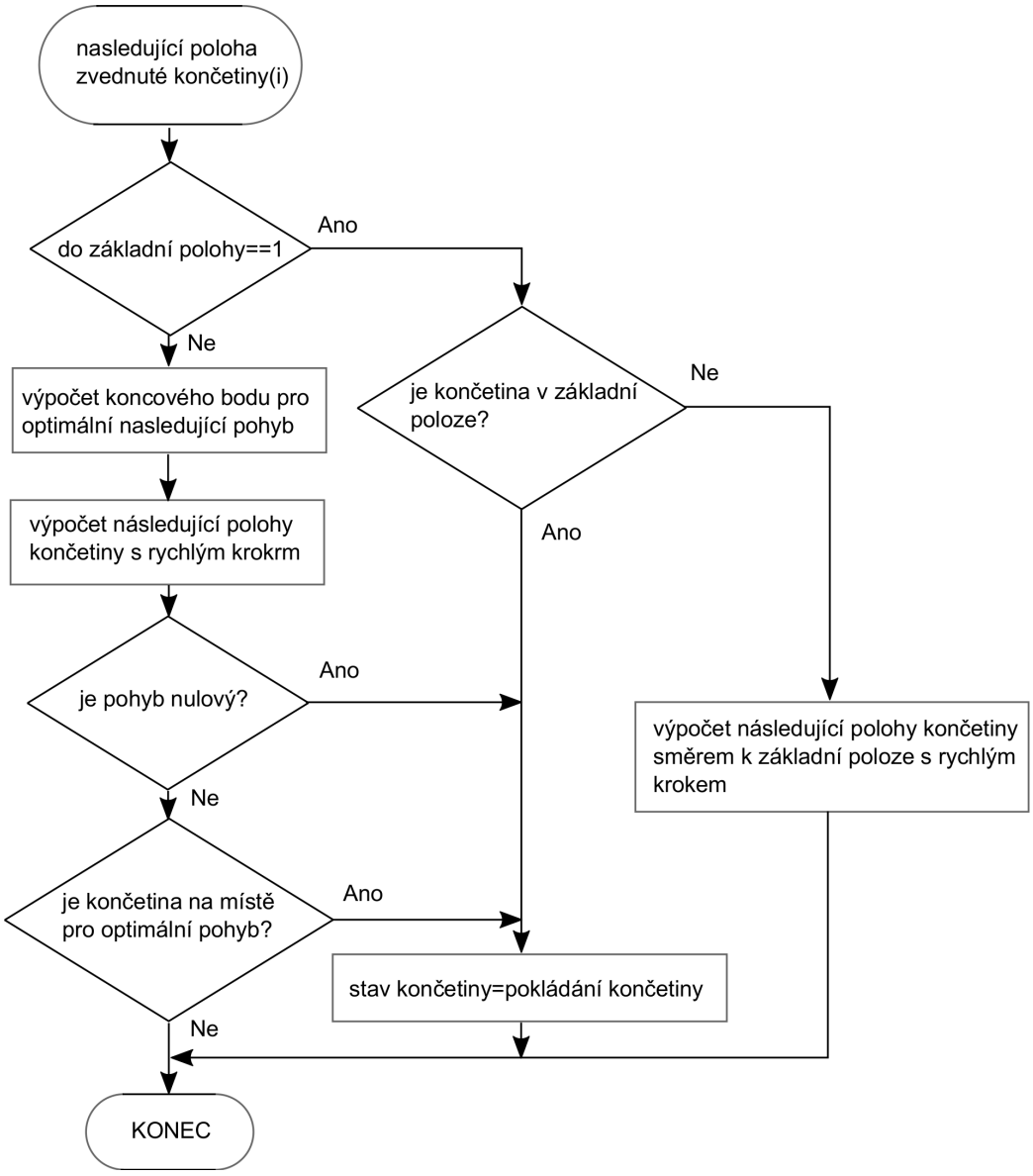
V případě, že není aktivní požadavek na přemístění končetin do základní polohy, je vypočítáno optimální místo pro přemístění končetiny. Optimální místo pro návrat je počítáno tak, aby následující pohyb položené končetiny ve směru, který odpovídá poslednímu pohybu robota, byl přes pracovní oblast co nejdelší. Opět v tomto stavu ale nebude této polohy dosaženo. Jakmile se končetina dostane do pracovní oblasti, mění se stav na zvednutá končetina. Tento stav se tedy zdá být nadbytečný, nicméně zajišťuje to, že v tomto stavu končetina nemůže přejít do stavu pokládání, pokud je vyžadován nulový pohyb. Končetina se tedy nejdřív přesune do pracovní oblasti a až tam může být položena při nulovém pohybu.



Obr. 5.6: Vývojový diagram pro stav zvednutá končetina mimo pracovní oblast

- **zvednutá končetina**

Tento stav (obr. 5.7) reprezentuje vracející se končetinu, která je zvednutá nad povrchem a v pracovní oblasti. Výpočet optimálního místa pro návrat končetiny je stejný jako ve stavu zvednutá končetina mimo pracovní oblast, rozdíl nastává jen v přechodu do dalšího stavu v okamžiku, kdy noha dosáhla buď optimální polohy pro návrat, nebo polohy pro základní umístění končetiny. Po tomto stavu následuje pokládání končetiny a celý cyklus stavového automatu se opakuje.



Obr. 5.7: Vývojový diagram pro stav zvednutá končetina

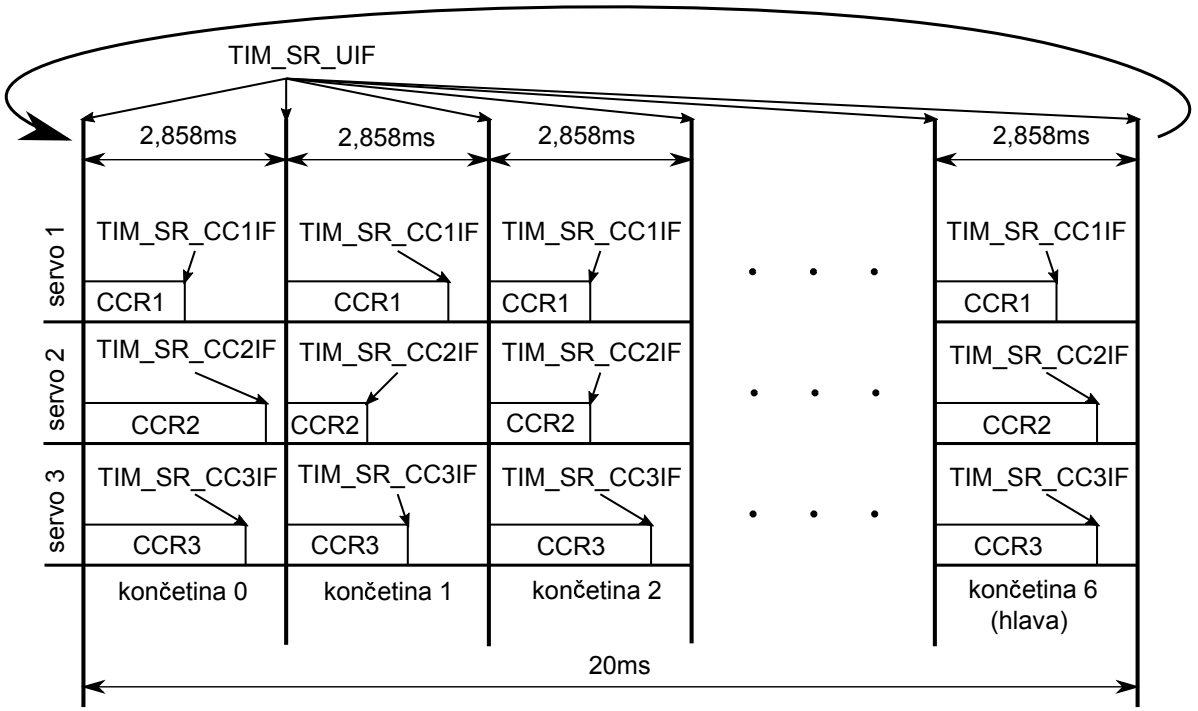
5.2.2.3 Generování signálů pro akční členy

Jak již bylo zmíněno v kapitole 3.2, modelářské servo se ovládá pulzy generovanými s frekvencí 50 Hz. Trvání pulzů je od 600 μs do 2400 μs. Pro ovládání je použit 16-ti bitový Čítač/časovač2, pomocí něhož jsou generovány signály pro osmnáct serv. Toto množství ovšem není maximum. Maximum je při použití Čítače/časovače2 generovat signály až pro třicet dva serv. Použitý Čítač/časovač2 obsahuje celkem čtyři capture/compare jednotky, ale použity jsou pouze tři.

Při generování pulzů se využívá toho, že maximální doba pulzu je výrazně kratší než perioda. Díky této vlastnosti je celá perioda trvající 20 ms rozdělena do sedmi oken, kde každé trvá 2,858 ms. Tyto okna jsou generována pomocí čítače a jeho autoreload registru $TIM2\_ARR$. Čítač čítá do hodnoty, kterou obsahuje registr $TIM2\_ARR$. V okamžiku rovnosti obou registrů je vyvoláno přerušení bitem TIM_SR_UIF a čítací registr je vy-

nulován. Na začátku každého okna jsou nastaveny výstupní piny do logické „1“. Jedná se vždy o tři piny, které odpovídají všem třem servům na jedné končetině. Díky tomu je možné pracovat se strukturou celé končetiny a v jednotlivých oknech jen měnit index pole. Po nastavení pinů je do třech capture/comare jednotek, konkrétně do jejich registrů $CCR1$, $CCR2$ a $CCR3$ vložena hodnota, která odpovídá době trvání pulzů. Následné přerušení bude od jedné ze tří capture/compare jednotek, nastavením bitu $TIM\_SRCCxIF$. V přerušení je pin, který odpovídá tomuto přerušení, nastaven do logické „0“ a přerušení je ukončeno. Do doby, než bude vyvoláno další časové okno, přijdou ještě dvě zbývající přerušení od captur/compare jednotek, kde jsou příslušné piny opět nastaveny do logické „0“. Tento cyklus se opakuje pro sedm oken, z čehož vyplývá perioda kanálu 20 ms. Časový diagram je na obrázku 5.8

Čítače/Časovače2 je řízen hodinovým signálem, který je odvozen od AHB sběrnice, která je kvůli jádru mikrokontroléru taktována na maximální kmitočet 72 MHz. Signál vstupuje do děličky kmitočtu pro sběrnici APB1, která kmitočet vydělí dvěma. Dále hodinový signál prochází násobičkou, která kmitočet naopak vynásobí dvěma a na vstupu Čítače/časovače2 je kmitočet 72 MHz. Čítač/časovač2 má vlastní děličku, která je nastavena tak, aby jedna inkrementace čítacího registru trvala 2 μs. V děličce je tedy hodnota 0x8F. Pro generování časového okna trvajícího 2,858 ms je v autoreload registru $TIM2\_ARR$ hodnota 0x595. Minimální rozlišení výstupního signálu je tedy 2 μs, což převyšuje citlivost modelářských serv, která mají pásmo necitlivosti okolo 8 μs.



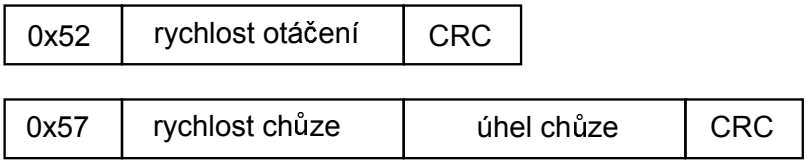
Obr. 5.8: Časový diagram ovládání serv

5.3 Komunikace

Pro komunikaci mezi ovladačem a robotem bylo zvoleno Bluetooth rozhraní, které je dnes implementováno v každém smartphone nebo notebooku. Díky tomu není potřeba připojovat žádná externí zařízení a celé řešení je značně flexibilní.

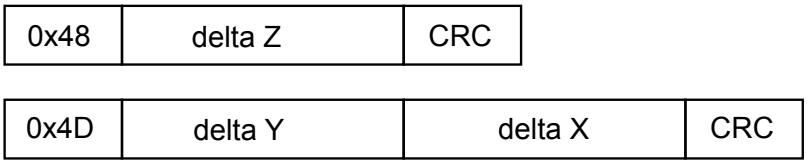
5.3.1 Komunikační protokol

Komunikace mezi robotem a ovladačem je jednosměrná a to od ovladače směrem k robotovi. Komunikace probíhá po paketech. Každý paket začíná specifickým symbolem, který udává typ dat. Všechny pakety mají pevnou délku a jsou zakončeny kontrolním součtem CRC, který je počítán přes celý paket. Generující polynom je CRC-8-CCITT ve tvaru $x^8 + x^2 + x^1 + x^0$. Na následujících obrázcích jsou všechny druhy paketů s vysvětlivkami, o jaká data se jedná.



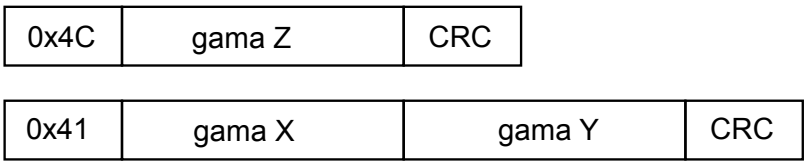
Obr. 5.9: Pakety pro chůzi robota

Na obrázku 5.9 jsou znázorněny dva pakety, které souvisí s chůzí robota. První paket uvozuje byte s hodnotou 0x52. Tento paket nese jediný byte dat s označením rychlost otáčení. Data jsou reprezentována jako znaménkový byte a hodnoty -128 nastavuje maximální rychlost otáčení proti směru hodinových ručiček a hodnota 127 otáčení po směru hodinových ručiček. Při hodnotě 0 se robot neotáčí. Druhým paketem, kterým lze ovlivnit chůzi, je paket obsahující dva byty dat a začínající hodnotou 0x57. Tyto dva byty tvoří vektor, jakým směrem a jakou rychlostí robot jde. Byte rychlost chůze je bezznaménkový a čím vyšší číslo, tím rychleji robot jde. Úhel chůze určuje, jakým směrem robot jde. Tento byte je reprezentován jako znaménkový a je normován na referenci π. Oba pakety jsou zakončeny bytem s kontrolním součtem CRC.



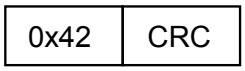
Obr. 5.10: Pakety pro translační pohyb těla

Pakety, které provádí translační pohyb těla (obr. 5.10), začínají hodnotou 0x48 nebo 0x4D. Všechny datové byty jsou znaménkového typu. Data jsou rozděleny do dvou paketů kvůli ovládání, kde pro pohyb po osách x a y je dvourozměrný ovládací prvek, který modifikuje data současně. Osa z má vlastní ovládací jednorozměrný prvek.



Obr. 5.11: Pakety pro rotační pohyb těla

Poslední dva pakety (obr. 5.11), které přímo ovlivňují pohyb robota, začínají hodnotou 0x4C a 0x41. Data jsou opět byty znaménkového typu a nastavují rotaci těla okolo všech tří os. Rozdělení do dvou paketů opět souvisí s ovládáním, kde osa x a y je nastavována pomocí akceleračního snímače. Osa z přebírá data od jednorozměrného ovládacího prvku.

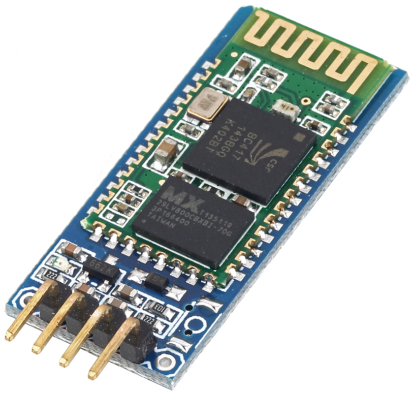


Obr. 5.12: Paket pro návrat do základní polohy

Posledním paketem, který se může v komunikaci objevit, je paket, který nemá žádná data. Skládá se pouze z bytu o hodnotě 0x42, který je následován kontrolním součtem. Tento paket slouží pro uvedení končetin do základního rozestavení. Tento paket je posílán při přechodu z módu chůze do módu pohybu těla a opačně.

5.3.2 Bluetooth modul

Jak již bylo zmíněno výše, robot s ovladačem komunikuje pomocí Bluetooth rozhraní. Mikrokontrolér však nedisponuje touto periferií, proto je k mikrokontroléru připojen modul, který toto rozhraní implementuje. Vybrán byl modul HC-06 (obr. 5.13), který je připojen k mikrokontroléru pomocí USART rozhraní. Parametry modulu jsou v tabulce 5.1



Obr. 5.13: Bluetooth modul HC-06 |Převzato z [7]|

Mód	Slave
Bluetooth specifikace	v2.0+EDR
Frekvence	2.4GHz ISM pasmo
Modulace	GFSK(Gaussian Frequency Shift Keying)
Vyzářený výkon	≤4dBm, Class 2
Citlivost	≤84dBm
Přenosová rychlost	2,1Mbps (Max)
Zabezpečení	Autentizace a šifrování
Profil	Bluetooth serial port
Napájecí napětí [V]	3 - 5
Rozměry [mm]	26,9 x 13 x 2,2

Tab. 5.1: Parametry Bluetooth modulu HC-06

Bluetooth rozhraní je obecně typu master slave. Zařízení typu master vždy navazuje komunikaci a v celé síti smí být pouze jedno. Modul HC-06 umí pracovat pouze v módu slave. Zařízení, které se chová jako master, musí být ovladač, tedy smartphone nebo PC.

Inicializace modulu se provádí ve stavu, kdy modul není spárován. Tento stav je indikován blikající LED na modulu. V tomto stavu modul přijímá příkazy přes sériové rozhraní, které jinak slouží pro přenos dat. Defaultní nastavení sériové linky je 9600 Bd, 8 datových bitů, 1 stop bit a žádná parita. Modul se nastavuje AT příkazy. Každý AT příkaz začíná znaky AT+, po kterých následuje konkrétní příkaz. Nastavit lze přenosovou rychlost, jméno, které vidí ostatní Bluetooth zařízení, PIN pro párování a paritu. Pro účely ovládání, kde nejsou vyžadovány velké datové toky, rychlost 9600 Bd bez problému vyhovuje. Modifikovány byly pouze parametry jméno příkazem AT+NAMEFaky, kde Faky je jméno, které vidí ostatní zařízení. Druhý změněný parametr je PIN pro párování pomocí příkazu AT+PIN7171. Pro párování je tedy nutné zadat 7171. Po každé změně modul odpoví, v případě změny jména výpisem OKFaky a v případě změny PINU OKsetpin.

6

Ovládání

Jak již bylo zmíněno výše, elektronické vybavení robota obsahuje Bluetooth modul pro komunikaci s ovládáním. Jediný požadavek na ovládání je schopnost být master a komunikovat definovanými pakety, které jsou popsány v kapitole 5.3.1 V úvahu připadají dvě řešení ovládání. Prvním je ovládat robota přes PC. V dnešní době každý notebook disponuje Bluetooth rozhraním, které splňuje výše zmíněné požadavky. Toto řešení nabízí značně jednodušší implementaci na platformě Windows, na druhou stranu řešení nenabízí takové možnosti komfortu a celé řešení je značně nemobilní. Druhým řešením a zároveň zvoleným pro tuto práci, je ovládání robota pomocí smartphone s operačním systémem Android. Toto řešení přináší výhody v celkové mobilitě, možnosti použití akceleračního snímače a dotykového ovládání.

6.1 Hardware

Pro vývoj softwaru byl požit smartphone HTC HD2, na který je portován operační systém Android 4.3 Jelly Bean, API Level 18. Tento smartphone obsahuje procesor Qualcomm Snapdragon S1-QSD8250 taktovaný na 1 GHz. Procesor je doplněn o 448 MB operační paměti a 512 MB systémové paměti. Dále je telefon osazen displejem o rozlišení 480x800 bodů. Z parametrů je zřejmé, že dnes se jedná o zařízení, které se řadí do úplného low-end segmentu a tudíž vyvinutý program by měl bez oblému pracovat s jakýmkoli novým zařízením. Problém může nastat v případě jiného rozlišení displeje, jelikož celé grafické rozhraní bylo navrženo pro toto rozlišení.

6.2 Aplikace

Aplikace byla napsána ve vývojovém prostředí ADT Bundle verze 20140702 [8] v jazyce Java7. Tento balík obsahuje Android SDK, který poskytuje API knihovny, vývojové nástroje pro ladění a sestavení programu. Dále tento balík obsahuje vývojové prostředí Eclipse IDE s plaginem ADT (Android Developer tools).

6.2.1 Menu aplikace

Menu aplikace zajišťuje tři hlavní funkce. První funkcí je zapínání a vypínání Bluetooth adapteru. Druhou funkcí je vyhledávání dostupných Bluetooth zařízení a párování s nimi. Tyto první dvě funkce nejsou nezbytně nutné, jelikož zapínání a vypínání adapteru a párování zařízení je možné v manažeru, který je součástí systému Android. Implementace těchto funkcí zvyšuje komfort uživatele, který může veškeré úkony spojené s robotem provádět v jedné aplikaci. Poslední funkcí je výběr zařízení a přechod do aktivity s joystickem.

Vstupním bodem do programu je funkce *onCreate*(), která se nachází ve spouštěné aktivitě. V této funkci probíhá přiřazení jednotlivých instancí, které jsou vytvořeny pomocí XML návrhu, aby s nimi mohlo být dále pracováno. Jedná se o instance tříd *Switch* na zapínání Bluetooth adapteru, *ListView* pro zobrazení jednotlivých parametrů nalezených Bluetooth zařízení, *TextView* pro spuštění hledání Bluetooth zařízení, *TextView* pro zobrazení informace o stavu vyhledávání a *ProgressBar*, který indikuje proces hledání Bluetooth zařízení. Rozložení jednotlivých prvků je specifikováno v XML souboru, který je na začátku této metody zobrazen.

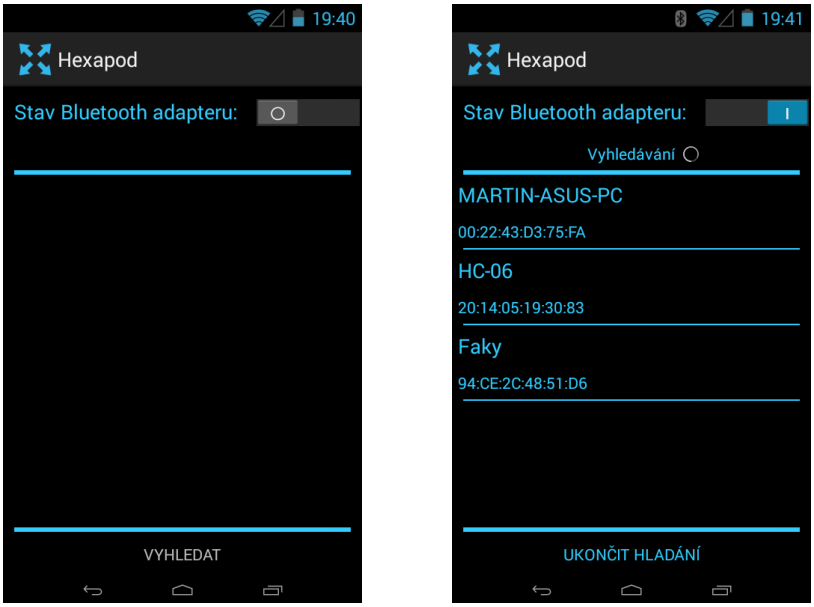
Dále se v této metodě vytváří instance třídy *BluetoothAdapter*, zavoláním statické metody *getDefaultAdapter*(), která se nachází ve třídě *BluetoothAdapter*. Tato metoda vrací objekt odpovídající lokálnímu Bluetooth adapteru, pokud zařízená tímto adapterem disponuje, pokud ne, vrací null.

V této metodě se také *ListView* plní jednotlivými zařízeními. Klasický widget *ListView* systém android implementuje jako obyčejný výčet textových řetězců. Tato rozvržení se pro zobrazení jednotlivých zařízení moc nehodí. Z toho důvodu byla navržena podoba jednotlivých řádků, které má *ListView* obsahovat. Rozvržení řádku je definováno opět pomocí XML souboru, ve kterém se nachází dvakrát widget *TextView*, kde jeden zobrazuje jméno Bluetooth zařízení a druhý MAC adresu téhož zařízení. Poslední widgetem je modrá vodorovná čára, která graficky odděluje jednotlivé prvky *ListView* od sebe. Aby bylo možné rozvržení implementované XML souborem zobrazit jako jednotlivé řádky *ListView*, musí být vytvořen adapter, který dodává data do *ListView*. Tento adapter musí implementovat metodu *getView*(), která vrací objekt vytvořeného a naplněného *View*. Data pro adaptér jsou získána pomocí metody *getBondedDevices*(), která je volána na objektu třídy *BluetoothAdapter*. Tato metoda vrací kolekci objektů párovaných zařízení. Poté, co je spuštěno vyhledávání dostupných zařízení asynchronní metodou *startDiscovery*() na objektu typu *BluetoothAdapter*, jsou pomocí metody *add*(), kterou dědí adapter od třídy *ArrayAdapter*, nová zařízení do *ListView* přidávána.

V této metodě se ještě nachází nastavení filtru k jednotlivým intentům. Jsou povoleny intenty, které jsou vyvolány při startu vyhledávání zařízení, nalezení zařízení, dokončení vyhledávání a při změně stavu zařízení. Poslední zmíněný nastává například při vypnutí a zapnutí Bluetooth adapteru a dalších stavech, které ovlivňují adapter. Tento filtr je zaregistrován k receiveru, což je objekt typu *BroadcastReceiver*, který na intenty, které projdou filtrem, reaguje. Příkladem reakce na intent od nalezeného zařízení je vložení

tohoto zařízení do *ListView*, pokud se tam již nenachází. Posledním blokem v této metodě je implementace rozhraní a následná registrace posluchače k jednotlivým view, které mají reagovat na dotek.

Párování nových zařízení je realizováno vybráním zařízení z *ListView*, který ještě nemá status párovaného zařízení. V takovém případě je volána asynchronní metoda *createBond*(), která musí být volána přes odkaz na třídu a metodu, jelikož do třídy *BluetoothDevice* je implementována až od API 19. Po zavolání metody je uživatel vyzván k zadání PINU pro párování. V této aktivitě se ještě nachází metoda *onPause*(), kterou prochází životní cyklus aktivity v případě, že jiná aktivita přechází do popředí. V této metodě je zastaveno vyhledávání Bluetooth zařízení a odregistrován *BroadcastReceiver*. Na obrázku 6.1 je výše popsaná aktivita. Levý print screen zobrazuje deaktivovaný Bluetooth adapter. Na pravém je aktivovaný adapter a spuštěné vyhledávání s nalezenými zařízeními.



(a) Vypnutý Bluetooth adapter

(b) Zapnutý Bluetooth adapter a spuštěno vyhledávání

Obr. 6.1: Menu aplikace

Po vybrání zařízení z *ListView* je vyvolán intent, do kterého jsou vložena extra data v podobě objektu třídy *BluetoothDevice*, implementující rozhraní *Parcelable*, které je nutné pro předání objektu aktivitě. Spolu s tímto objektem je do intentu vložen řetězec, pomocí něhož je ve vyvolané aktivitě přístup k předávaným datům. Předávaný objekt reprezentuje vybrané zařízení, s nímž bude probíhat komunikace.

6.2.2 Joystick

Aktivita, která implementuje joystick, se skládá z jednotlivých widgetů, které graficky implementují jednotlivé ovládací prvky. Dále tato aktivita zajišťuje správu komunikace s

vybraným zařízením, které bylo do této aktivity předáno.

Aktivita opět začíná v metodě *onCreate*(), ve které se provádí přiřazení instancí vytvořených v XML souboru. Konkrétně se jedná o vytvořené widgety, které jsou popsány v kapitole 6.2.2.1. Dále jsou z intentu, který vyvolal tuto aktivitu, získána data, reprezentující objekt třídy *BluetoothDevice*, se kterým má být zahájena komunikace.

V této metodě se ještě nachází registrace broadcastreceiveru, který reaguje na intent vyvolaný přerušením komunikace, byť z důvodu slabého signálu, nebo ukončení komunikace od druhého zařízení. Aktivita na tento intent reaguje návratem na předchozí aktivitu, která tvoří menu aplikace.

Dalším blokem této metody je inicializace jednotlivých widgetů, které tvoří ovládací prvky. Jedná se o rozmístění widgetu a registraci posluchačů na dotyk. V callback metodě, kterou posluchač implementuje, jsou zpracována data o poloze dotyku a na tuto polohu je přemístěn widget, který zobrazuje rukojeť ovládacího prvku. Při zpracování dat o poloze jsou data upravena tak, aby prvky zobrazující rukojeti jednotlivých prvků nemohly opustit pracovní oblast. V jednorozměrném prvku se rukojeti smí pohybovat jen v jedné ose a po vymezeném prostoru, podobně u dvourozměrného prvku se rukojeť smí pohybovat jen uvnitř kruhu. Následně získaná data jsou upravena do definovaných paketů a předána k odeslání. Dále zde probíhá inicializace akcelerometru a implementace callback metody pro zpracování a odeslání dat.

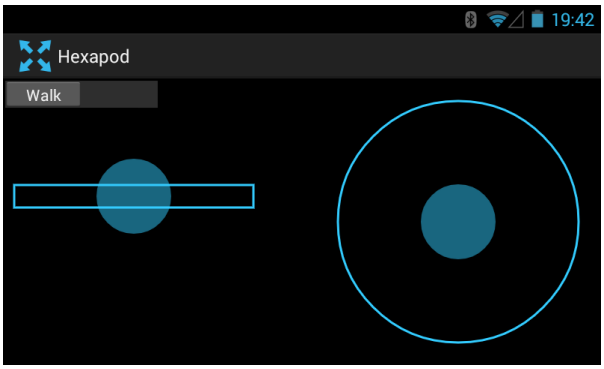
Posledním bodem této metody je vytvoření vlákna *ConnectThread*, které se stará o navázání komunikace. Postup je popsán v kapitole 6.2.2.2

Na začátku této aktivity se ještě nachází definice objektu třídy *Handler*. Jedná se o objekt, pomocí něhož si jednotlivá vlákna předávají zprávy. Pomocí tohoto objektu je možné v hlavním vláknu zpracovávat data, která byla přijata pře Bluetooth, který je obsluhován v pracovním vláknu. Další zprávy, které jsou předávány, jsou o neúspěšném navázání spojení. Díky tomuto objektu může být vypsáno varování, jelikož grafické prvky smí být měněny pouze z hlavního vlákna. Poslední funkcí tohoto objektu je, při úspěšném navázání spojení ve vláknu *ConnectThread*, spustit vlákno *ConnectedThread*, které se stará o vstupní a výstupní proudy.

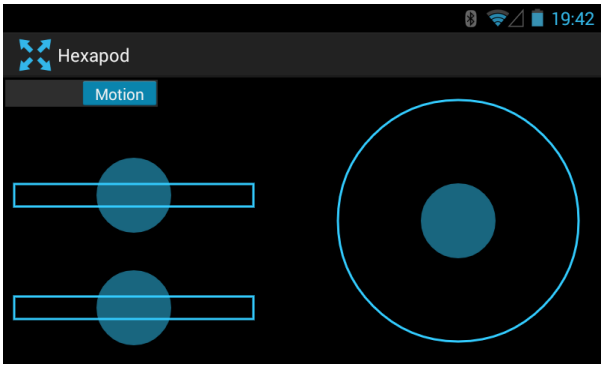
Jelikož je příliš mnoho veličin, které lze ovládat a na jednu obrazovku by se všechny ovládací prvky nevešly, jsou pohyby rozděleny do dvou skupin a mezi nimi se lze přepínat pomocí přepínače v levém horním rohu. První skupina pohybu se týká chůze, která je ovládaná pomocí jednoho jednorozměrného ovládacího prvku pro otáčení a jednoho dvourozměrného pro směr a rychlost chůze. Druhou skupinou jsou pohyby těla, kde jsou dva jednorozměrné ovládací prvky pro nastavení translace těla ve směru osy z a rotace těla okolo osy z. Dále je v této skupině dvourozměrný ovládací prvek pro translační pohyb těla ve směru os x a y. Poslední možností ovládání těla v této skupině je použití akcelerometru pro rotaci okolo os x a y. Při přechodu mezi stavy je poslána zpráva o uvedení robota do základní polohy. Na obrázku 6.2 jsou oba módy ovládání.

V manifestu aplikace je nadefinováno rozvržení obrazovky pouze v landscape módu

bez možnosti rotace. Toto nastavení zamezí otáčení obrazu na displeji při módu pohybu těla, kde pohybem smartphone je ovládán pohyb těla robota.



(a) Joystick pro ovládání chůze



(b) Joystick pro ovládání pohybu těla

Obr. 6.2: Aktivita joystick

6.2.2.1 Widgety ovládacích prvků

Pro tuto aplikaci byly vytvořeny dva druhy ovládacích prvků. První je použit pro ovládání jen v jedné ose a tudíž se jedná o táhlo. Tento ovládací prvek je použit pro translační pohyb těla ve směru osy z, rotaci okolo osy z a otáčení okolo osy z. Druhým typem je dvourozměrný ovládací prvek pro translační pohyb těla ve směru os x a y a nastavení směru a rychlosti chůze.

View, které tvoří ovládací prvek, je implementováno vlastní třídou, která dědí od předka $View$. Tato třída musí implementovat metodu $onMeasure()$. V této metodě jsou definovány rozměry view. Do této metody vstupují parametry, které navrhují jeho rozměry. Návrh rozměru pochází od kontejneru, ve kterém je view umístěno. Na konci této metody musí být zavolána metoda $setMeasuredDimension()$, která rozměry nastaví.

V této třídě se ještě nachází metoda $onDraw()$, ve které probíhá samotné kreslení prvků. Jednorozměrný ovládací prvek je nakreslen obdélníkem a dvourozměrný kruhem. Posledním vytvořeným widgetem je kruh vyplněný barvou, který simuluje rukojeť ovláda-

cích prvků tím, že kopíruje pohyb prstu po displeji. Implementace je stejná jako u widgetu popsaných výše.

Se samotným widgetem se pracuje úplně stejně jako s každým jiným view. View je možné rozmístit pomocí XML souboru a dále měnit pozici v programu, jak je tomu u view, které zobrazuje rukojeť ovládacího prvku a kopíruje pohyb prstu po displeji. Rozmístění jednotlivých prvků je na obrázku 6.2.

6.2.2.2 Komunikace

Komunikace probíhá ve vlastním vláknu. Toto vlákno je spuštěno z hlavního vlákna z metody *onCreate*(). Prvním úkolem tohoto vlákna je získat objekt typu *BluetoothSocket* z objektu typu *BluetoothDevice*, který reprezentuje vybrané zařízení pro komunikaci. Vybrané zařízení, reprezentované objektem typu *BluetoothDevice*, je předáno do konstruktoru tohoto vlákna. V konstruktoru je na tomto objektu zavolána metoda *createRfcommSocketToServiceRecord*(), která má jako parametr 128-bitový univerzální a jedinečný identifikátor *UUID*. Tato metoda vrací objekt typu *BluetoothSocket*, který používá protokol Serial Port Profile (SPP), ve kterém modul HC-06 pracuje.

Objekt typu *BluetoothSocket* vznikl z původního objektu, který reprezentoval vybrané zařízení a tudíž má dostatek informací pro navázání spojení s tímto zařízením pomocí volání metody *connect*() na objektu typu *BluetoothSocket*. Obě metody výše popsané musí být ošetřeny na výjimky, které jsou vyhozeny v případě neúspěchu. Pokud proběhne vše v pořádku, vlákno pošle úspěšně vytvořený objekt typu *BluetoothSocket* pomocí objektu typu *Handler* zpět do hlavního vlákna a toto vlákno skončí.

Následně je v hlavním vláknu vytvořeno vlákno *ConnectedThread*. Do konstruktoru toho vlákna vstupuje objekt, který je výsledkem práce vlákna *ConnectThread* a je typu *BluetoothSocket*. V konstruktoru jsou na tomto objektu volány metody *getInputStream*() a *getOutputStream*(), metody vrací objekty typu *InputStream* a *OutputStream*. Oba objekty reprezentují datové proudy.

Po startu vlákna voláním metody *run*() se vlákno zacyklí v nekonečné smyčce, která na objektu typu *InputStream* volá metodu *read*() pro čtení přijatých dat. Pokud jsou nijaká data přijata, jsou pomocí objektu typu *Handler* poslána do hlavního vlákna, kde jsou zpracována. Pro odesílání dat slouží v tomto vlákně metoda *write*(), do které vstupuje jako parametr pole znaků pro odeslání. Jelikož se běh vlákna zacyklí v nekonečné smyčce, je v tomto vlákně implementována metoda *cancel*(), která uzavře daný socket voláním metody *close*() právě na tomto objektu. Všechny výše zmíněné metody musí být ošetřeny na výjimky.

Kompletní zdrojový kód aplikace se nachází na přiloženém CD-R

7

Závěr

Cílem této diplomové práce byl celistvý návrh malého modelu robotické platformy s více končetinami, která by sloužila pro prezentační účely naší fakulty.

První část této práce se zabývá mechanickým návrhem celé robotické platformy. Vybrána byla robotická platforma, která obsahuje celkem šest končetin rozmístěných ve dvou řadách podél delších stran těla. Mechanický návrh byl proveden v 3D CAD softwaru SolidWorks 2013, kde byl celý robot sestaven a kde byla ověřena pohyblivost jednotlivých končetin.

Namodelované díly plošného charakteru byly vyrobeny frézováním z tvrzených PVC desek o tloušťce 5mm. Díly prostorově náročnějšího charakteru byly vyrobeny technologií 3D tisku z materiálu PLA. Nedílnou součástí mechanické konstrukce jsou akční členy, jelikož v konstrukci tvoří samonosné prvky. Vybrána byla analogová modelářská mikroserva Hitec HS-82MG s kovovými převody. Po sestavení mechanické konstrukce se ukázala hmotnost celého robota značně vysoká, z důvodu použitého materiálu tvrzeného PVC. Akční členy v určitých případech nemají dostatek síly a při chůzi, kdy robot stojí jen na třech končetinách, se tělo robota propadá. Propad těla je v řádu několika milimetrů a robot i přes tento hendikepem je schopen plnohodnotné chůze.

Po sestavení mechanické konstrukce byl sestaven matematický model, který popisuje reálného robota. Kinematika byla navržena tak, aby bylo možné provádět s tělem robota translační pohyby ve všech třech osách a zároveň i okolo těchto os tělo rotovat. Tělo robota je tedy schopno jakéhokoli pohybu v prostoru. Chůze robota byla implementována stavovým automatem. Implementovány jsou pohyby přímočaré chůze jakýmkoli směrem a otáčení robota okolo svého středu. Samozřejmostí je tyto pohyby kombinovat, kde výsledný pohyb může být například chůze do zatáčky bokem těla.

Další částí této práce je návrh elektronického vybavení. Řídicí jednotka je osazena mikrokontrolérem STM32F103. Jádro mikrokontroléru je taktováno na 72 MHz, kde při tomto taktu poskytuje dostatek výkonu pro výpočet celé kinematiky. Dále je mezi elektronickým vybavením navržený DC-DC step-down měnič pro snížení napětí pro akční členy.

Poslední část práce je zaměřena na realizaci ovládání celého robota. Zvoleno bylo

ovládání pomocí smartphone s operačním systémem Android přes bezdrátové rozhraní Bluetooth. Pro Android 4.3 Jelly Bean, API Level 18 byla napsána aplikace v jazyce Java s grafickým rozhraním simulující joystick. Dále aplikace využívá akcelerační snímač pro pohyb těla robota, kde tělo kopíruje pohyb smartphone.

Výsledkem práce je tedy plně funkční model s bezdrátovým ovládáním, který splnil všechny body zadání. Dále se model zúčastnil soutěže Studentská vědecká odborná činnost, kde získal první místo.

V budoucnu je plánované místo, které tvoří hlavu robota osadit kamerou a elektronické vybavení doplnit o obvod FPGA s implementovanými metodami rozpoznávání obrazu. Robot by s tímto vylepšením mohl být schopen aportovat míček, nebo vyhledávat cestu, popřípadě přenášet obraz rovnou do smartphone.

Literatura

[1] *John Deere Walking Tractor*. 2009. Dostupné z: http://1.bp.blogspot.com/˙qgeHobkBqrA/SXqMe3qNobI/AAAAAAAAABw/YkIm26DMADs/s1600-h/walking-tractor.jpg

[2] *LS3 Mule*. 2014. Dostupné z: http://www.technologyreview.com/˙longform/boston-dynamics/assets/imgs/wildcat.jpg

[3] FAKTOR, Martin. *Delta-robot řízený z počítače*. Plzeň, 2013. Bakalářská. Západočeská univerzita v Plzni.

[4] *3D CAD Design Software SOLIDWORKS* [online]. 2014 [cit. 2015-04-23]. Dostupné z: https://www.solidworks.com

[5] ANALOG DEVICES, Inc. *Constant Frequency Current-Mode Step-Down DC-to-DC Controller in TSOT*. Rev. C. Norwood, MA 02062-9106, U.S.A., 2005, 16 s. Dostupné z: http://www.analog.com/en/products/power-management/switching-power-converters/switching-controllers/adp1864.html

[6] *Keil MDK-ARM* [online]. 2015 [cit. 2015-04-23]. Dostupné z: http://www.keil.com/arm/mdk.asp

[7] *HC-06*. 2013. Dostupné z: https://tienda.patagoniatecnology.com/wp-content/uploads/2013/12/bluetooth˙hc06˙master.jpg

[8] *Android Development Tools (ADT)*. 2014. Dostupné z: http://developer.android.com/tools/sdk/eclipse-adt.html

[9] *HC Serial Bluetooth Products: User Instructional Manual*. 2012. Dostupné z: www.wavesen.com

[10] HEROUT, Pavel. *Učebnice jazyka C*. 6. vyd. České Budějovice: Kopp, 2009, 271, viii s. ISBN 978-80-7232-383-8.

[11] HEROUT, Pavel. *Učebnice jazyka Java*. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

[12] SCHILDT, Herbert. *Java 7: výukový kurz*. 1. vyd. Brno: Computer Press, 2012, 664 s. ISBN 978-80-251-3748-2.

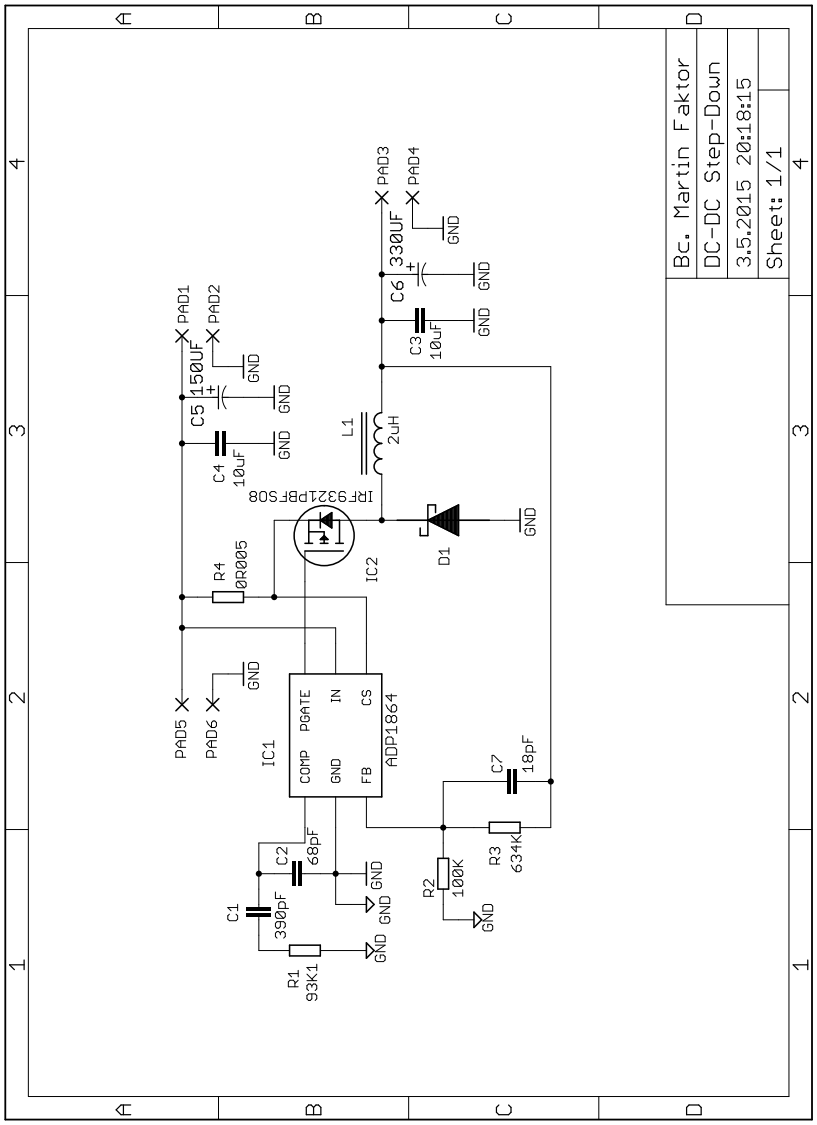
[13] MURPHY, Mark L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

[14] UJBÁNYAI, Miroslav. *Programujeme pro Android*. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.

Příloha A

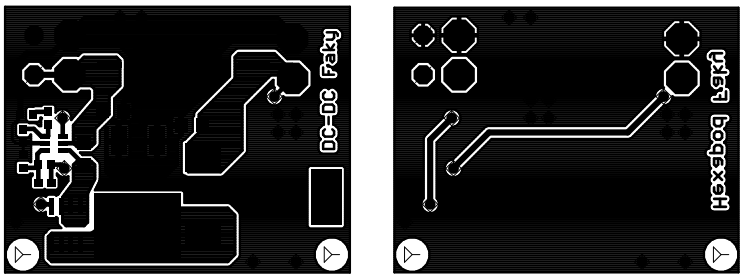
Napájecí jednotka

A.1 Schéma zapojení



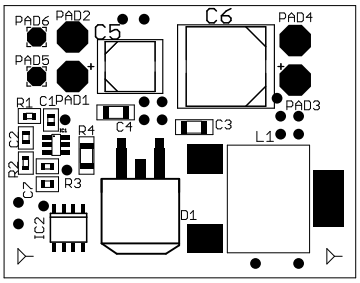
Obr. A.1: Schéma napájecí jednotky

A.2 Deska plošných spojů



(a) TOP vrstva (b) BOT vrstva

Obr. A.2: Plošný spoj



Obr. A.3: Osazovací výkres TOP vrstvy

A.3 Partlist

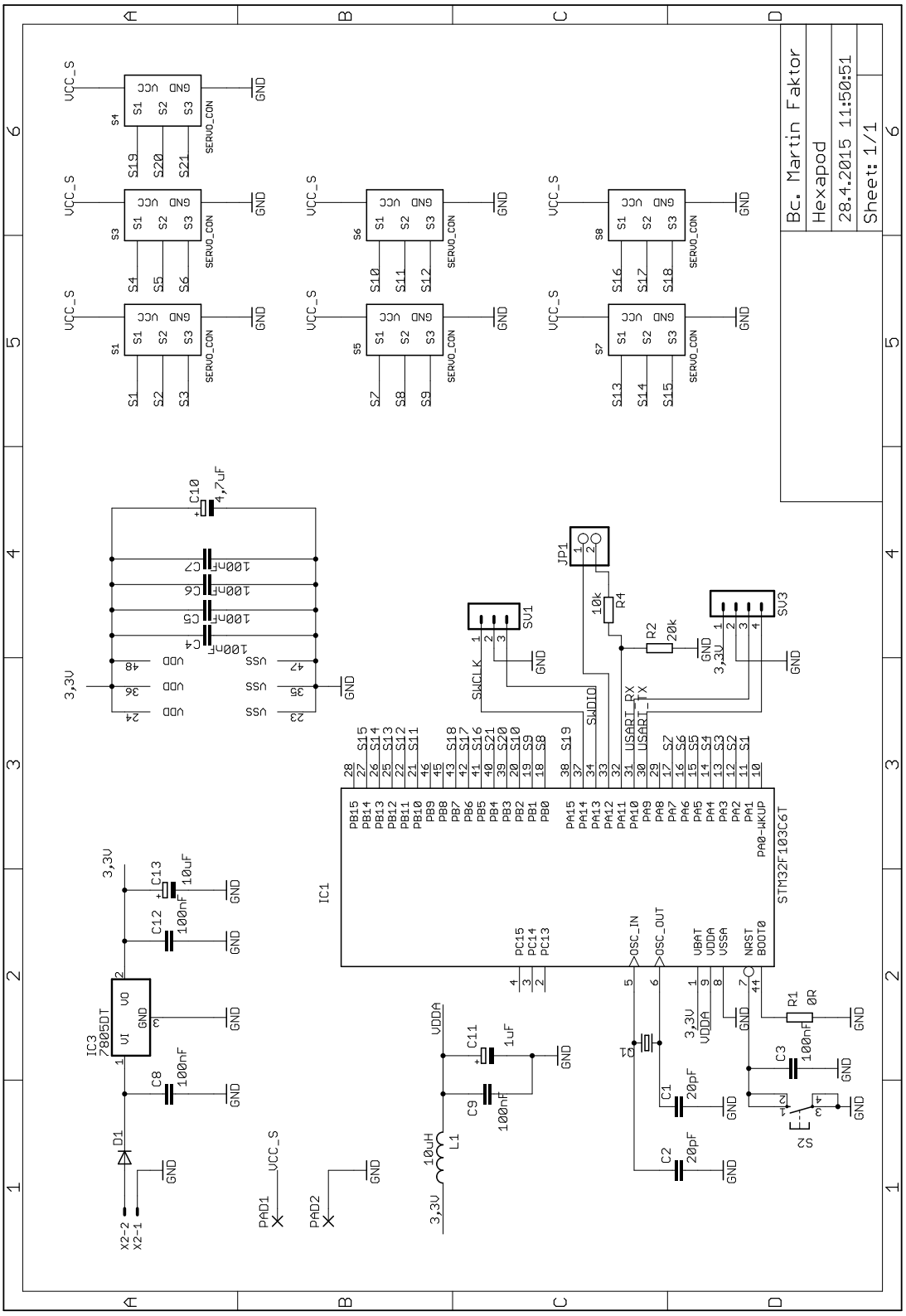
Part	Value	Device	Package
C1	390 pF	C-EUC0603	C0603
C2	68 pF	C-EUC0603	C0603
C3, C4	10 μF	C-EUC1206	C1206
C5	150 μF	EEE-FK1C151XP-CASE-D8	CAPAE660X800N
C6	330 μF	EEE-FT1H331AP-CASE-G	CAPAE1030X1050N
C7	18 pF	C-EUC0603	C0603
D1		MBRB20H100CTT4GD2PAK	D2PACK
IC1		ADP1864	TSOT
IC2		IRF9321PBFSO8	SO8
L1	2 μH	SER2010-202MLD	SER20
R1	93k1	R-EU_R0603	R0603
R2	100k	R-EU_R0603	R0603
R3	634k	R-EU_R0603	R0603
R4	0R005	R-EU_R1206	R1206

Tab. A.1: Seznam součástek

Příloha B

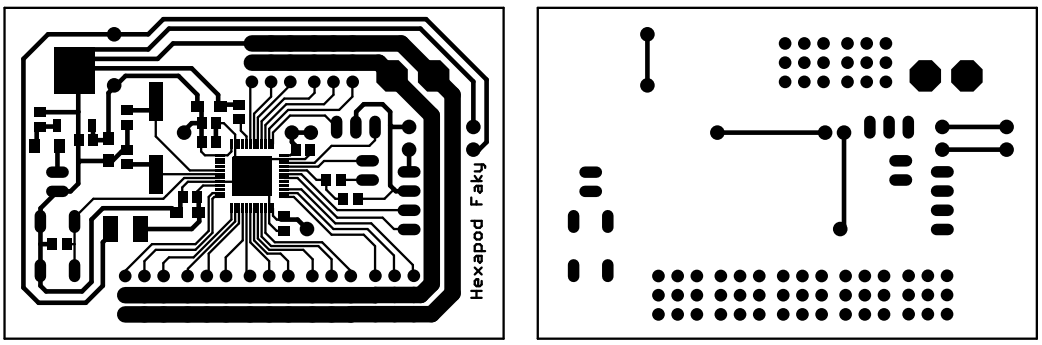
Řídicí jednotka

B.1 Schéma zapojení



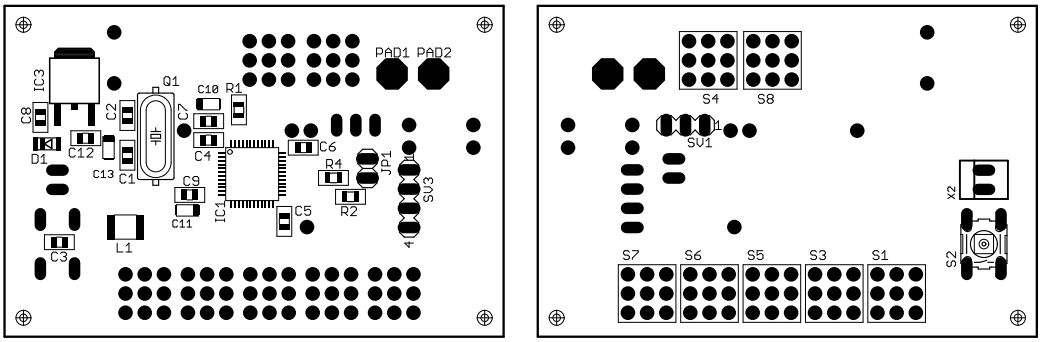
Obr. B.1: Schéma řídicí jednotky

B.2 Deska plošných spojů



(a) TOP vrstva (b) BOT vrstva

Obr. B.2: Plošný spoj



(a) TOP vrstva (b) BOT vrstva

Obr. B.3: Osazovací výkresy

B.3 Partlist

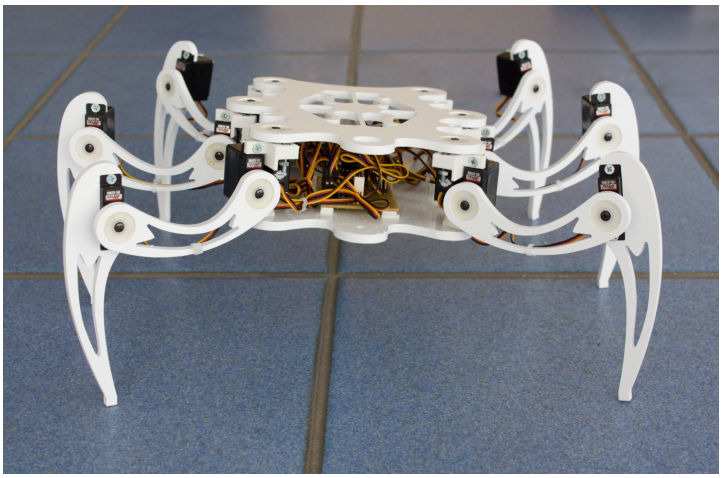
Part	Value	Device	Package
C1, C2	20 pF	C-EUC0805	C0805
C3, C4, C5, C6, C7, C8, C9, C12	100 nF	C-EUC0805	C0805
C10	4,7 μF	CPOL-EUSMCA	SMC_A
C11	1 μF	CPOL-EUSMCA	SMC_A
C13	10 μF	CPOL-EUSMCA	SMC_A
D1		DIODE-SOD80C	SOD80C
IC1		STM32F103C6T	LQFP48
IC3		7805DT	TO252
JP1		PINHD-1X2	1X02
L1	10 μH	L-USL1812	L1812
Q1		CRYSTALSM49	SM49
R1	0R	R-EU_R0805	R0805
R2	20k	R-EU_R0805	R0805
R4	10k	R-EU_R0805	R0805
S1, S3, S4, S5, S6, S7, S8		SERVO_CON	SERVO_CON
S2		10-XX	B3F-10XX
SV1		MA03-1	MA03-1
SV3		MA04-1	MA04-1
X2	22-23-2021	22-23-2021	22-23-2021

Tab. B.1: Seznam součástek

Příloha C

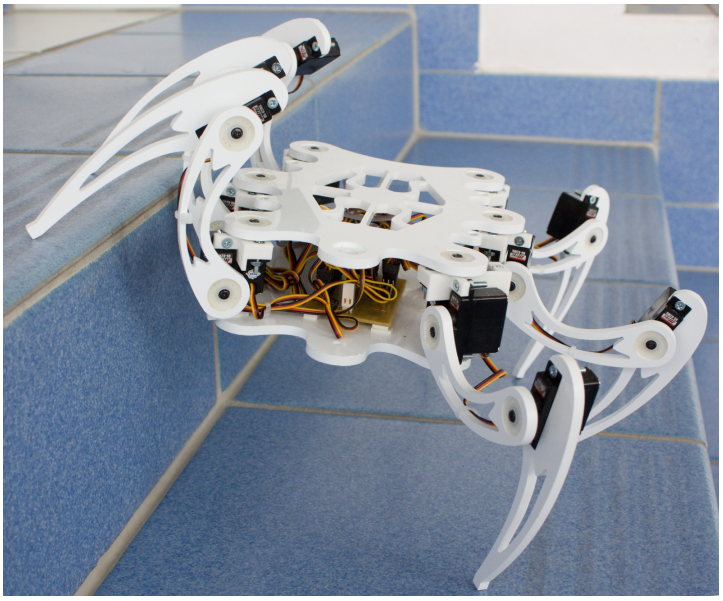
Fotografie

Obr. C.1: Základní poloha

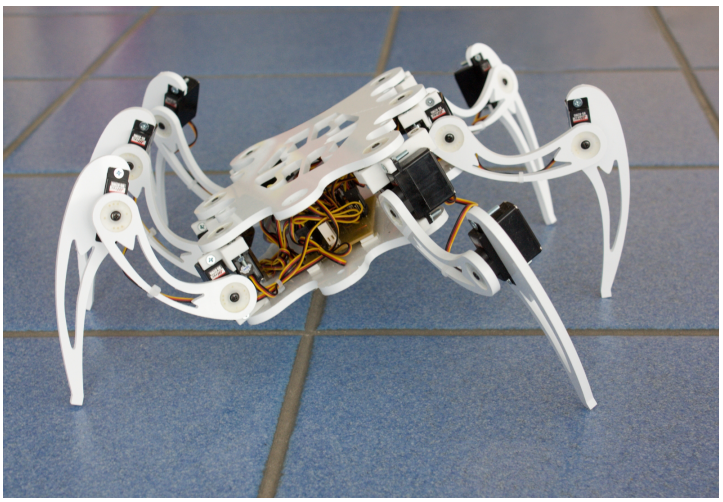


Obr. C.2: Translace ve směru osy z

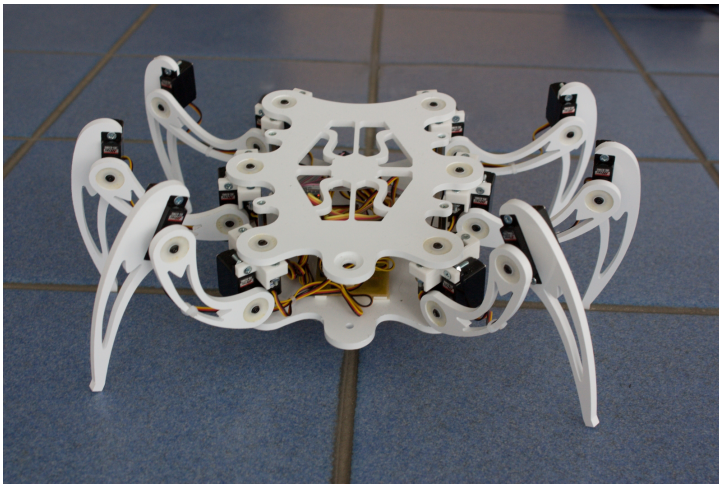
Obr. C.3: Translace ve směru osy y



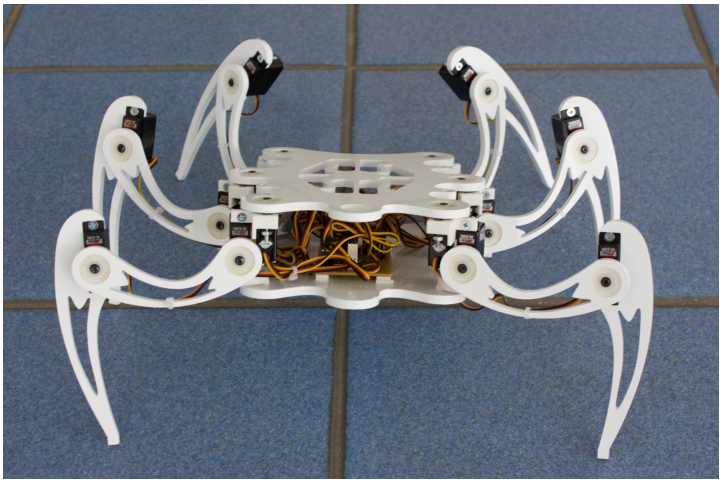
Obr. C.4: Rotace okolo osy x



Obr. C.5: Rotace okolo osy x



Obr. C.6: Rotace okolo osy y



Obr. C.7: Rotace okolo osy y

Obr. C.8: Kombinace pohybů