ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

BAKALÁŘSKÁ PRÁCE

Řídící jednotka manipulátoru

vedoucí práce: **Petr Beneš**
autor: **Pavel Adler**

2013

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Pavel ADLER**
Osobní číslo:	**E09B0245P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Název tématu:	**Řídicí jednotka manipulátoru**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Seznamte se s úlohami kinematiky robotických systémů.
2. Navrhněte vhodné řídicí algoritmy pro konkrétní manipulátor.
3. Navrhněte jednotku pro řízení manipulátoru.
4. Implementujte navržené algoritmy do řídicí jednotky.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **20 - 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

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

Vedoucí bakalářské práce: **Ing. Petr Beneš**
Katedra elektromechaniky a výkonové elektroniky

Konzultant bakalářské práce: **Ing. Petr Beneš**
Katedra elektromechaniky a výkonové elektroniky

Datum zadání bakalářské práce: **15. října 2012**

Termín odevzdání bakalářské práce: **7. června 2013**

L.S.

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

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. června 2012

Abstrakt

Předkládaná bakalářská práce je zaměřena na základní pohled na robotiku a její rozdělení v současnosti. Dále je tato práce zaměřena na přímou a inverzní kinematiku, seznámení se s modelem AL5A od společnosti Lynxmotion a navrhnutí jeho řídícího algoritmu. Při počítání natočení úhlu serv bylo využito goniometrických funkcí a Heronova vzorce. Dále je tato práce zaměřena na návrh a konstrukci řídící jednotky pro manipulátor a následné implementování algoritmu do vyrobené desky.

Klíčová slova

Robotika, manipulátor, servo, kinematika, chapadlo, efektor, ATMega32A, Eagle

Abstract

Presented bachelor thesis focuses on the basic view for the area of robotics and its distribution in the present. Furthermore, this work focuses on direct and inverse kinematics. Introduction to model AL5A from Lynxmotion and propose the control algorithm. When calculating the rotation angle of servers were used trigonometric functions. Furthermore, this work focuses on the design and construction of the control unit for the manipulator and then implemented the algorithm to the board.

Key words

Robotics, manipulator, servo, kinematics, tentacle, effector, ATMega32A, EAGLE

Prohlášení

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

Prohlašuji, že jsem tuto práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 7.6.2013 Pavel Adler

……..

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Petru Benešovi za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

OBSAH **7**

1 SEZNAM SYMBOLŮ A ZKRATEK **9**

ÚVOD **10**

2 ÚVOD DO PROJEKTU **11**

- FÁZE PROSTUDOVÁNÍ 11
- FÁZE ANALYZOVÁNÍ 11
- NÁVRH 11
- REALIZACE ŘEŠENÍ 12
- ZKUŠEBNÍ PROVOZ 12
- UVEDENÍ DO PROVOZU 12

3 MANIPULÁTORY **13**

3.1 DĚLENÍ MANIPULÁTORŮ DLE TYPU PROGRAMU: 13
3.1.1 Manipulátory s pevným programem *13*
3.1.2 Manipulátory s proměnným programem *13*
3.2 DĚLENÍ DLE STUPŇŮ VOLNOSTI: 14
3.3 DĚLENÍ DLE DRUHU POHONŮ: 14
3.4 DĚLENÍ MANIPULÁTORU DLE TYPU POHYBU: 14
3.4.1 Pohyb z bodu do bodu *14*
3.4.2 Lineární interpolace *14*
3.4.3 Kruhová interpolace *14*
3.5 DĚLENÍ DLE TYPU GEOMETRIE PRACOVNÍHO BODU: 15
3.5.1 Kartézská kinematická struktura *15*
3.5.2 Cylindrická kinematická struktura *15*
3.5.3 Sférická kinematická struktura *15*
3.5.4 Angulární kinematická struktura *15*
3.6 DĚLENÍ MANIPULÁTORU DLE ZPŮSOBU PROGRAMOVÁNÍ: 16
3.6.1 Přímé programování - učení *16*
3.6.2 Nepřímé programování – offline *17*
3.6.3 Přímé plánování – online *17*
3.7 PRACOVNÍ REŽIMY ROBOTA: 17
3.7.1 Ruční řízení *17*
3.7.2 Zadávání programu *17*
3.7.3 Automatické řízení *17*

4 ZÁKLADNÍ INFORMACE O HARDWAROVÉM VYBAVENÍ **18**

4.1 MODEL MANIPULÁTORU 18
4.2 ŘÍDÍCÍ DESKA 20
4.3 NÁVRH DESKY 22

5 TEORIE ŘÍZENÍ SERV **27**

5.1 ŘÍZENÍ SERVOMECHANIZMŮ 27

5.2 PŘÍMÁ A INVERZNÍ KINEMATIKA 27
5.3 GONIOMETRICKÉ FUNKCE 27

6 SOFTWARE 28

6.1 TESTOVÁNÍ TLAČÍTEK 28
6.2 OBSLUHA LCD 28
6.3 VÝPOČTY ÚHLŮ SERV 28
6.4 VÝPOČET SOUŘADNIC A ÚHLU EFEKTORU 30
6.5 PŘEPOČET VELIKOSTI ÚHLU DO ŠÍŘKY PULZU 31
6.6 ODESÍLANÍ PULZŮ NA SERVA 31
6.7 KONTROLA ROZSAHŮ 32

7 ZÁVĚR 33

SEZNAM LITERATURY A INFORMAČNÍCH ZDROJŮ 34

PŘÍLOHY 1

1 Seznam symbolů a zkratek

např. - například

tzv. - takzvaný

tzn. - to znamená

resp. - respektive

Úvod

Důvod, proč jsem si vybral tuto práci je, že robotika je velice rozšířené téma a je využívána ve velkém množství odvětví.

První část je zaměřena na seznámení se s robotikou. Jsou zde popsány způsoby rozdělení dle počtu stupňů volnosti, kinematické struktury, vykonávání činností či druhu pracovního prostoru.

V druhé části je popsán model AL5A od firmy Lynxmotion a popis jednotlivých servomechanismů, pomocí kterých ovládáme celé rameno. Dále je zde popsán návrh řídící desky, její schéma zapojení a deska plošného spoje v programu Eagle.

Ve třetí části je popsána přímá a inverzní kinematika, výpočet souřadnic efektoru a výpočet úhlů jednotlivých serv, za použití goniometrických funkcí a Heronova vzorce.

2 Úvod do projektu

Každý složitější projekt, aby ho bylo možné realizovat, je nutné rozložit na jednotlivé bloky. Ty jsou navrženy tak, aby reprezentovaly systémový přístup k problému. Rozdělení bloků:

- problém je rozdělen na jednotlivé dílčí části
- dílčí části jsou rozloženy na základní problémy
- vyřešení základních problémů
- postupné spojování vyřešených základních problému
- spojení jednotlivých části do jednoho celku
- otestování správné funkčnosti vyřešených problémů

Když se navrhuje řídicí systém, je zapotřebí na jednotlivé problémy nahlížet spíše jako na návrh jednotlivého projektu a posléze jeho realizaci. Jakýkoliv projekt se skládá z jednotlivých, po sobě jdoucích částí, které dohromady tvoří posloupnost řešení problému. Definice každé části se podle různých názoru liší, ale hlavní myšlenku lze vyjádřit jako: [1]

- **Fáze prostudování**

Tato fáze se zabývá co možná nejdůkladnějším a nejlepším definováním dané problematiky. Uživatel zadá úkol a pověřená osoba musí najít jednotlivé základní problémy a postupně na ně najít řešení:

- upřesnění základních problémů
- nalezení řešení problémů
- prezentování zákazníkovi

- **Fáze analyzování**

Absolutní pochopení řešení daných problémů a plánování splnění jednotlivých bodů projektu:

- Analýza funkčního projektu
- Ohodnocení funkčnosti

- **Návrh**

Získání informací a materiálů pro realizaci řešení problému:

Volba software a hardware

Návrh struktury programu

- **Realizace řešení**

  Tvorba, testovaní a postupné dolaďovaní programu

  Tvorba jednotlivých podprogramů

  Testování správné funkčnosti programu

  Zdokumentování projektu

- **Zkušební provoz**

  Fáze, ve které se program testuje. Je testován tak, jako by šlo o normální provoz.

- **Uvedení do provozu**

  Závěr celého projektu, kde je program uveden do pracovního režimu. To, jak se bude program dlouho vyvíjet, vychází z jeho složitosti.

Jednotlivé bloky nemusí vždy být sestavovány v tomto pořadí, je možné některé z nich zaměnit anebo je dokonce dát dohromady. Pokud jsme časově omezeni, je zde možnost sestavování některých bloků současně. Po úspěšném zvládnutí každého bloku by se měla funkčnost otestovat a podle výsledku tohoto testování by se měl odvíjet postup realizace funkčnosti bloku dalšího. Pokud se řešení nějakého bloku nebude zdát efektivní, je nutné jej předělat tak, abychom efektivitu co nejvíce zvýšili. [1]

3 Manipulátory

Manipulátor neboli průmyslový robot je v podstatě automatický stroj, který má dvě a více pracovních os. Je možné jej naprogramovat podle potřeby v podstatě na libovolnou funkci. Výběr funkcí je ovšem omezen potřebností různých nástavců pro určité funkce. Manipulátory nahrazují člověka tam, kde již není potřeba, využívají se zejména kvůli:

- Zvýšení kvality práce
- Větší flexibilitě
- Zrychlení výroby
- Ušetření pracovních míst
- Nasazení tam, kde to je pro člověka životně či zdravotně nebezpečné

Z toho vyplývá, že je efektivnější využití manipulátorů zejména na místech, kde se provádí sériová výroba, pracovní podmínky jsou pro lidi nebezpečné anebo tam, kde již nelze využít člověka kvůli jeho možnostem. [2]

3.1 Dělení manipulátorů dle typu programu:

3.1.1 Manipulátory s pevným programem

Program mají napevno nahraný a změna se neprovád Výhodou je nízká cena, jednoduchost a spolehlivost. [2]

3.1.2 Manipulátory s proměnným programem

Program lze měnit podle potřeby. Využívá se tam, kde se předpokládá, že bude potřeba jednou za čas upravit výrobní linku, například v automobilkách, kde je nutné předělat výrobní linku kvůli výrobě nového typu auta. [2]

Každý manipulátor je složen ze dvou typů systému. První typ systému je systém, který nastavuje polohu robotu dle potřeby, proto se mu říká systém pro nastavení pozice. Druhý typ systému slouží ke změně směru robotu, v podstatě nám robota otočí o nastavený úhel tak, abychom se dostali do požadované polohy. Každý manipulátor mívá dva až tři stupně volnosti v ramenu. Požadavky na jeho konstrukci většinou bývají takové, aby byl co nejmenší, nejlehčí, s maximálním rozsahem otáčení a maximálně možnou rychlostí. Další částí je pak samotná hlavice, která slouží k manuální práci, jako například k vrtání, sváření, broušení či uchopování a přenášení. Uchopování může být pomocí elektromagnetů v hlavici,

podtlakové hlavice či klasické hlavice s úchopnými prsty. Samozřejmě každý pracovní úkon je závislý na správném použití určité násady, například není možné vrtačkou nabarvit auto.

Při pohybu rozlišujeme různé druhy toho, jak se hlavice dostává z počáteční pozice do zadané pozice. [2]

3.2 Dělení dle stupňů volnosti:

Stupeň volnosti je základní směr posunu nebo směru otáčení manipulátoru.

Univerzální manipulátor

Manipulátor, který má šest stupňů volnosti, jednoznačně definuje polohu bodu v kartézském souřadnicovém systému.

Redundantní manipulátor

Tento manipulátor má více než šest stupňů volnosti, což využívá k větší volnosti při obcházení překážek nebo při pohybu ve stísněném prostředí.

Deficitní manipulátor

Tento manipulátor má méně než šest stupňů volnosti, může provádět montáž v rovině. [2]

3.3 Dělení dle druhu pohonů:

V současné době je nejvíce průmyslových robotů a manipulátorů (PraM) s elektrickými pohony. Pokud je vyžadována vysoká nosnost, tak se používají manipulátory s hydraulickým pohonem, pro vysoké rychlosti pak pneumatické pohony. [2]

3.4 Dělení manipulátoru dle typu pohybu:

3.4.1 Pohyb z bodu do bodu

Pohyb z bodu do bodu (PTP- point to point) je pohyb, kdy se rameno manipulátoru přemístí do zadané polohy po nejkratší vzdálenosti. Tento pohyb je velice rychlý, takřka nepotřebuje řízení, ale může zavinit nehodu nebo škodu. [2]

3.4.2 Lineární interpolace

Rameno robota se pohybuje po přímkách. Výsledná trasa se tedy skládá ze všech přímek potřebných k dosažení požadované polohy. [2]

3.4.3 Kruhová interpolace

Při tomto typu pohybu se rameno pohybuje souvisle, to znamená, že se pohybuje ve dvou, ale i více osách zároveň. Je důležité, abychom znali polohu počátečního a koncového

bodu, a pak ještě buď polohu bodu, který leží na kružnici anebo bod, který je ve středu kružnice. [2]

3.5 Dělení dle typu geometrie pracovního bodu:

3.5.1 Kartézská kinematická struktura

Skládá se z tří lineárních interpolací, nedochází ke změně orientace objektu. Pracovní prostor připomíná hranol. Manipulátory s touto strukturou jsou rozměrné, a proto je třeba mít velký prostor, kam manipulátor umístíme. Používají se jako podavače nebo jako obsluha výrobních strojů. [2]

3.5.2 Cylindrická kinematická struktura

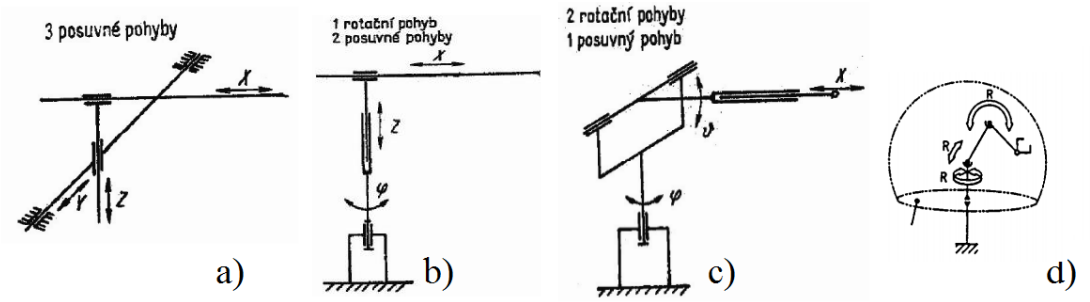
Skládá se z jedné kruhové interpolace a dvou lineárních interpolací, dochází ke změně orientace objektu. Pracovní prostor připomíná válcový prstenec. Vyžaduje velký operační prostor. Používá se jako obsluha vstřikovacích strojů nebo strojů na tlakové lití. [2]

3.5.3 Sférická kinematická struktura

Skládá se ze dvou kruhových interpolací a jedné lineární interpolace, dochází ke změně orientace objektu. Pracovní prostor připomíná kulový vrchlík. Poloha se nastavuje ve sférických souřadnicích. Uplatnění ve svařovacích linkách. [2]

3.5.4 Angulární kinematická struktura

Skládá se ze tří kruhových interpolací. Tato struktura se vyznačuje dobrými dynamickými vlastnostmi (např. vysoká rychlost manipulačních pohybů), nejmenším zastaveným prostorem a je snadné rozšíření jejího pracovního prostoru. Nevýhodou této struktury je, že manipulační prostor je menší. Angulární systémy jsou v praxi nejčastější. [2]

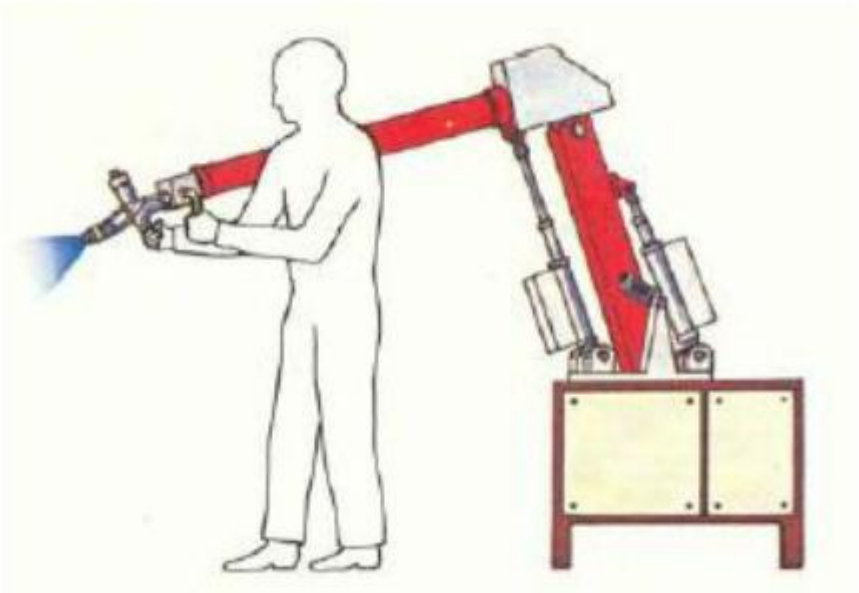


Obr. 01 Základní koncepce ramen [2]
a) Kartézská b) Cylindrická c) Sférická d) Angulární

3.6 Dělení manipulátoru dle způsobu programování:

3.6.1 Přímé programování - učení

Trasa ramena, která je tvořena body a typy pohybů, je zadávána ručně. [2]



Obr. 02 Přímé programování robota[3]

Ruční programování

V této metodě se zadávají souřadnice a pohyby ručně. Nejčastěji z klávesnice (např. chapadlo sevřít, úhel α = 30°). [2]

Metoda Teach-in

Tento způsob pracuje na principu snímání pohybu ramene při najíždění do jednotlivých bodů a souřadnice ukládá do paměti. Výhodou je přesnost a to, že uložených souřadnic je málo. Nevýhodou je, že se musí zadávat způsob, jakým se rameno dostane z bodu A do bodu B. Využívá se například u bodového sváření. [2]

Metoda Play-back

Při této metodě řídící systém snímá průběh dráhy a rychlost pohybu a automaticky tyto hodnoty ukládá do paměti. Využívá se pro složité pohyby s velkou přesností, jako například lakování a svařování. Člověk, který rameno ovládá, musí všechny pohyby dělat s velkou přesností, neboť jakýkoliv pohyb, který s ramenem provedeme, bude zaznamenán. [2]

Textové programování

Při textovém programování se celý pohyb ramene popisuje pomocí slovních příkazů. Nejprve se vytvoří kostra programu, což je sekvence programů, a pak se zadávají souřadnice bodů. [2]

3.6.2 Nepřímé programování – offline

Nepřímé programování zadává do programu trajektorii pohybu ramene, tato trajektorie je zadána křivkami v prostoru. Dále se pak musí nastavit typ úkonu, který má rameno vykonat. Tuto metodu lze použít i na inverzní kinematiku. [2]

3.6.3 Přímé plánování – online

Tato metoda je velice podobná předchozí metodě, s tím rozdílem, že inverzní kinematika se řeší v reálném čase. Na to jsou zapotřebí snímače, které nám dají odpověď na požadovanou otázku. [2]

3.7 Pracovní režimy robota:

3.7.1 Ruční řízení

Pomocí ručního ovládání lze robota ovládat a vyvolat jednotlivé příkazy jako např. různé příkazy k pohybu. Procesor, který řídí systém, tyto příkazy zpracuje a zadá regulátorům pohonů požadované hodnoty. [2]

3.7.2 Zadávání programu

Do paměti řídícího systému se ukládají uživatelské programy a hodnoty poloh, které mohou být vyvolány edičním programem. Veškeré příkazy a funkce jsou ve stanoveném programovacím jazyce. [2]

3.7.3 Automatické řízení

Automaticky jsou vyvolány uživatelské programy. Automatický průběh nám bude provádět jeden řádek programu po druhém, bez přerušení.[2]

4 Základní informace o hardwarovém vybavení

Pro řešení projektu je k dispozici zadané hardwarové vybavení. Vybavení se dá rozdělit na dvě skupiny. První skupinou je model samotného manipulátoru, a druhou je vývojový kit EvB 4.3 s procesorem ATMega32A.[4]

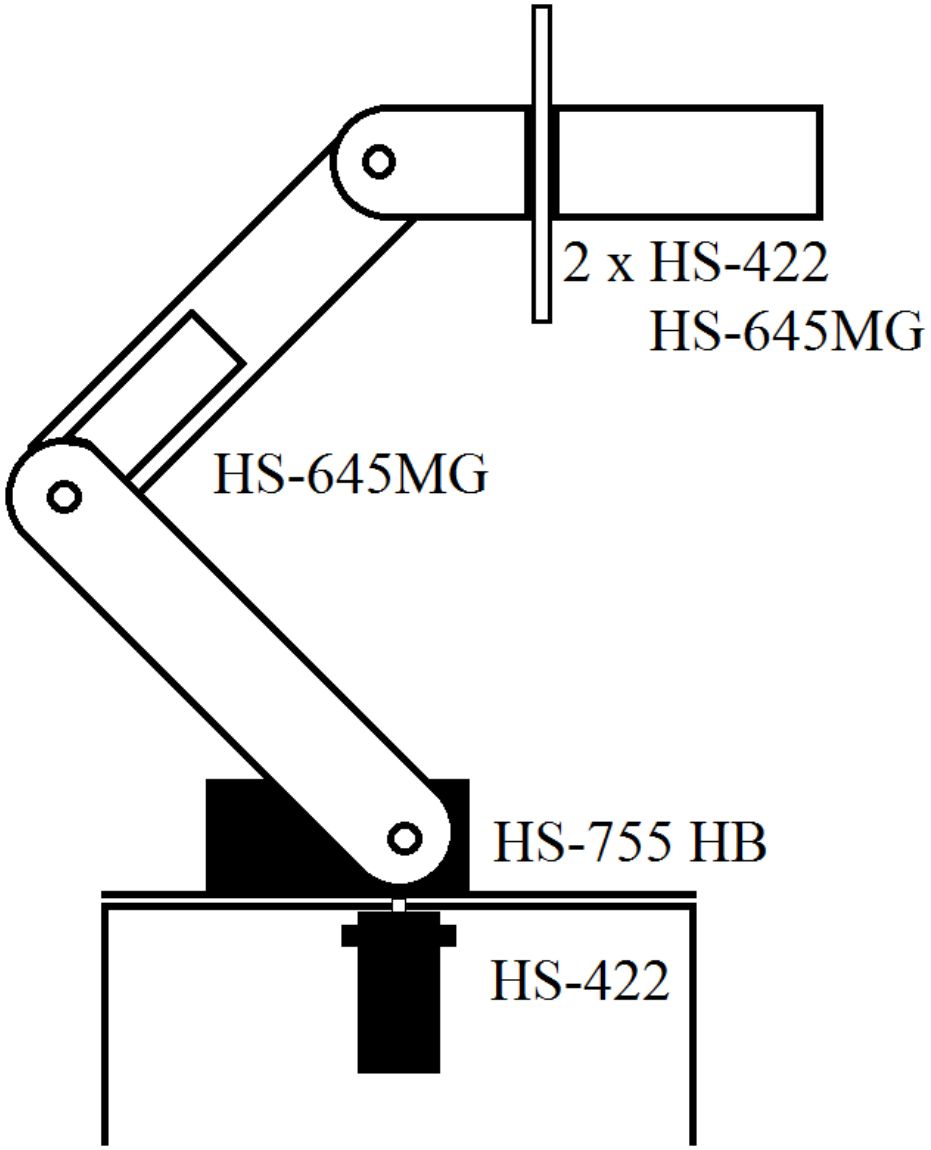
4.1 Model manipulátoru

Jedná se o servo model AL5A společnosti Lynxmotion se šesti stupni volnosti: otočná základna, rameno, loket, náklon zápěstí, rotaci zápěstí a chapadlo. Využívá se pro velké množství aplikací. Modelářská serva jsou od společnosti Hitec. Model manipulátoru je vybaven šesti osami, v kterých se může pohybovat. Pro nastavení pozice a polohy čelistí jsou zde tři osy (otáčení, X, Y). Další tři osy slouží k nastavení čelistí do správné polohy a úhlu. Manipulátor je vyfocen na *obr. 02.*

Obr. 03 Model servo manipulátoru[4]

Každá osa je ovládána servomotorem, který je ovládán pulzy z řídící jednotky. Na řídící desku je přivedeno šest jednotlivých serv. Na manipulátoru byla použita serva HS-422, HS-755HB (toto servo je mohutnější než servo HS-422, při stejném napětí má třikrát větší sílu a také proto jsou jeho součástky z kovu, a ne z plastu, jako HS-422). Třetí typ serva je HS-

645MG, tento typ má součástky také z kovu a je výkonově mezi HS-422 a HS-755HB. Kde je jaké servo použito, je zobrazeno na *obr. 03*.



Obr. 04, Umístění serv

V tab. 01 jsou uvedeny vlastnosti použitých serv.

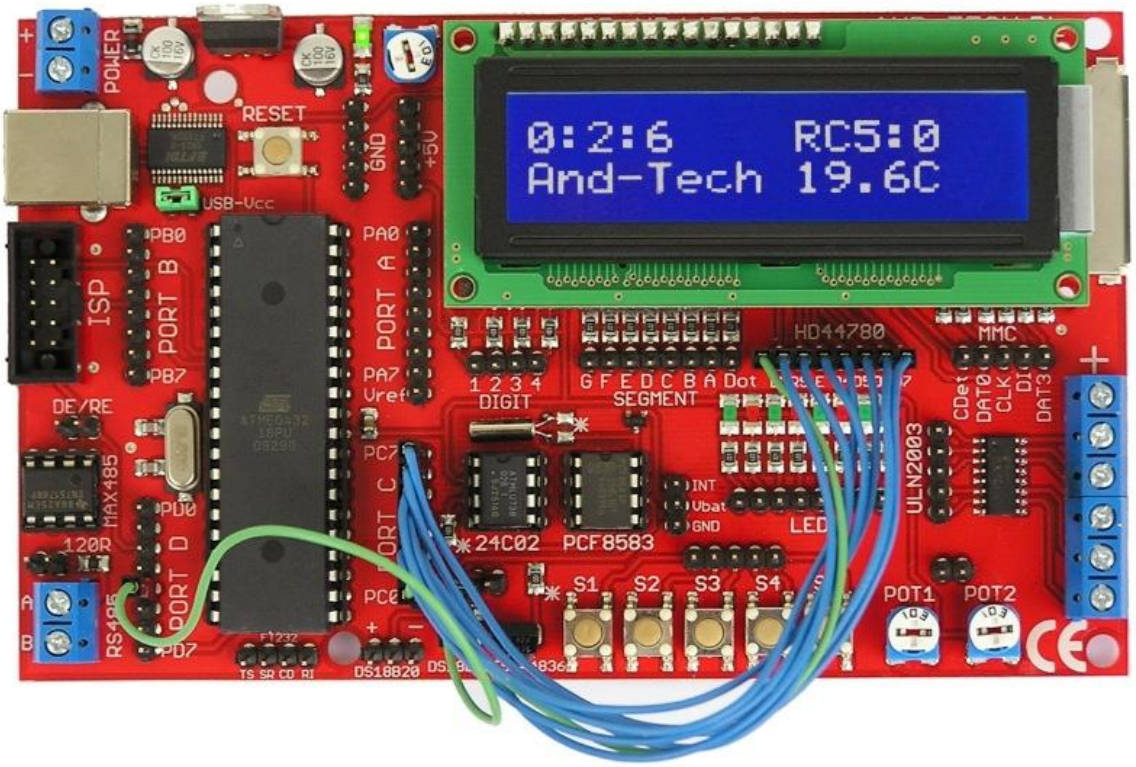
Tab. 01 Vlastnosti použitých serv[5]

Typ serva:	Moment při napájeni 4.8V	Moment při napájeni 6V	Materiál součástek
HS-422	3,3 kg/cm	4,1 kg/cm	Plast
HS-645MG	7,7 kg/cm	9,6 kg/cm	Kov
HS-755MG	11 kg/cm	13,2 kg/cm	Kov

4.2 Řídící deska

Deska EvB 4.3 od společnosti AND-TECH obsahuje:

- procesor ATMega32A
- 2x16 LCD display nebo 4x7segmentovy display
- USB port na propojení s PC
- ISP konektor
- 5 tlačítek[6]



Obr. 05 Vývojový kit EvB 4.3[6]

Firma Lynxmotion prodává ke svým robotům už hotové desky, mezi ně patří například deska BotBoarduino nebo deska SSC-32.[4]

Obr. 06 deska firmy Lynxmotion[4]
a) BotBoarduino *b)SSC-32*

Deska Botboarduino

- Kompatibilní se všemi roboty firmy Lynxmotion.
- Na I/O piny jsou připojena tlačítka s LED diodami, aby uživatelské rozhraní bylo co nejpřehlednější.
- Obsahuje jeden napájecí vstup pro mikrokontroler a jeden pro serva. Mikrokontroler může být napájen z jeho vlastního zdroje, USB kabelem nebo zdrojem pro serva.
- Obsahuje I/O piny, napájecí piny a zemnící piny.
- Deska má malé šroubovací svorky pro připojení napájení, takže není potřeba pájet dráty.
- Firma nabízí spoustu periferií, které se připojí přímo na I / O sběrnici, a mnoho dalších je ve vývoji.[4]

SSC-32

- Dokáže ovládat až 32 serv
- Pohyby serv okamžité, časované, s řízenou rychlostí nebo synchronní
- ATMega8

Desky od firmy Lynxmotion neobsahují žádné ovládací prvky (tlačítka), kterými by šlo ovládat manipulátor. Tyto desky se používají v aplikacích, kde manipulátor přijímá řídící signály od počítače. Pro svou úlohu potřebuji desku, která bude ovládací prvky obsahovat. Na desce EvB 4.3 tlačítka jsou, ale pro mé zadání je zapotřebí tlačítek více.[4]

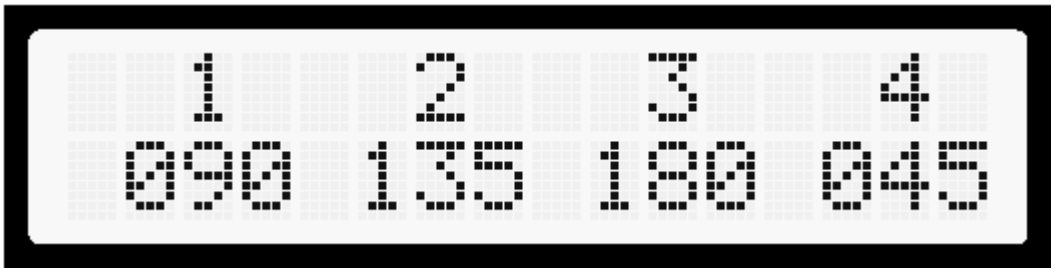
4.3 Návrh desky

Navrhnutá deska obsahuje:

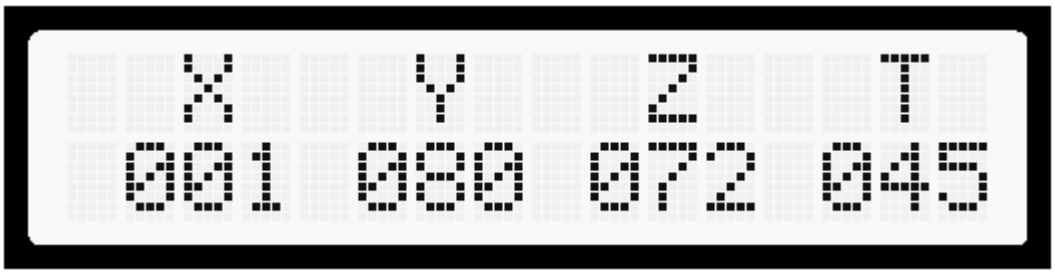
Tab. 02 Seznam použitých součástek

Komponent	Parametry součástky	Typ pouzdra
Procesor ATMega32A	-	DIL 40
17 tlačítek	6x6 mm	-
ISP konektor	10 pinů	-
LED display	16x2 znaků	-
7xLED diody	3 mm	-
Konektor na LCD	1x16 konektoru	-
6 výstupů pro serva	1x3 piny	-
7 rezistorů	1 kΩ	0805
1 rezistor	2,2 kΩ	0805
19 rezistorů	10 kΩ	0805
2 kondenzátory	1 μF	0603
1 kondenzátor	100 nF	0805
Napájecí svorkovnice	5 mm	-
Stabilizátor	-	SOT223

Desku jsem navrhoval s ohledem na počet ovládaných servr a způsob ovládání ramene. Desku jsem navrhoval v programu Eagle. Rameno se dá ovládat dvěma způsoby. První způsob je změnou úhlu každého serva zvlášť pomocí spodních dvanácti tlačítek, které jsou uspořádány do dvojic tak, že první dvojice tlačítek zleva ovládá první servo a první dvojice zprava ovládá šesté servo. Druhý způsob je změna souřadnic X, Y, Z a úhlu, pod kterým se efektor dostává na požadované souřadnice. Na *obr. 07* a *obr. 08* jsou zobrazeny nastavovaná hodnoty.

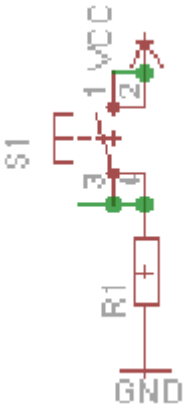


Obr. 07 Nastavování úhlů serv na display[7]



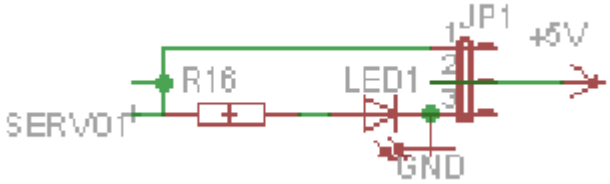
Obr. 08 Nastavování souřadnic efektoru na displey[7]

Nejprve je zapotřebí sestavit seznam, který obsahuje propojení všech součástek. Na navrhovanou desku jsem umístil sedmnáct tlačítek, které jsou připojeny k napájecímu napětí a po sepnutí dávají signál o hodnotě logické 1 (Ucc). Na *obr. 09* je znázorněno zapojení spínače S1, který nám ovládá servomotor 1.



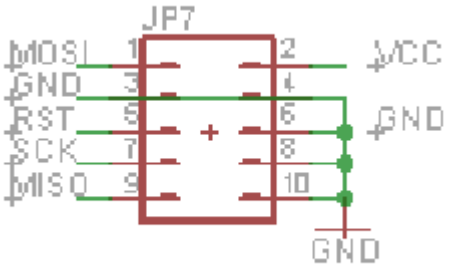
Obr. 09 Schéma zapojení tlačítek

Dále je zde za potřebí konektor, na který se připojí ovládané servo. Tento konektor je zhotoven z kolíkové lišty, která je nastříhána po třech kontaktech. Na první kontakt je připojena zem, na druhý je připojeno napájecí napětí a třetí konektor složí jako zdroj řídícího signálu pro servo.



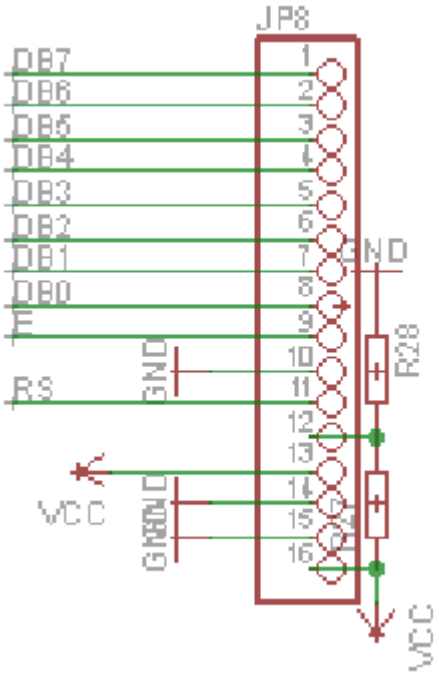
Obr. 10 Schéma zapojení napájecího konektoru pro servo

ISP konektor nám slouží jako propojení PC s procesorem. Na propojení je třeba externího programátoru, který má na jedné straně USB a na druhé ISP konektor. Tento konektor může být nezapojen, pokud už program máme nahraný v procesoru.



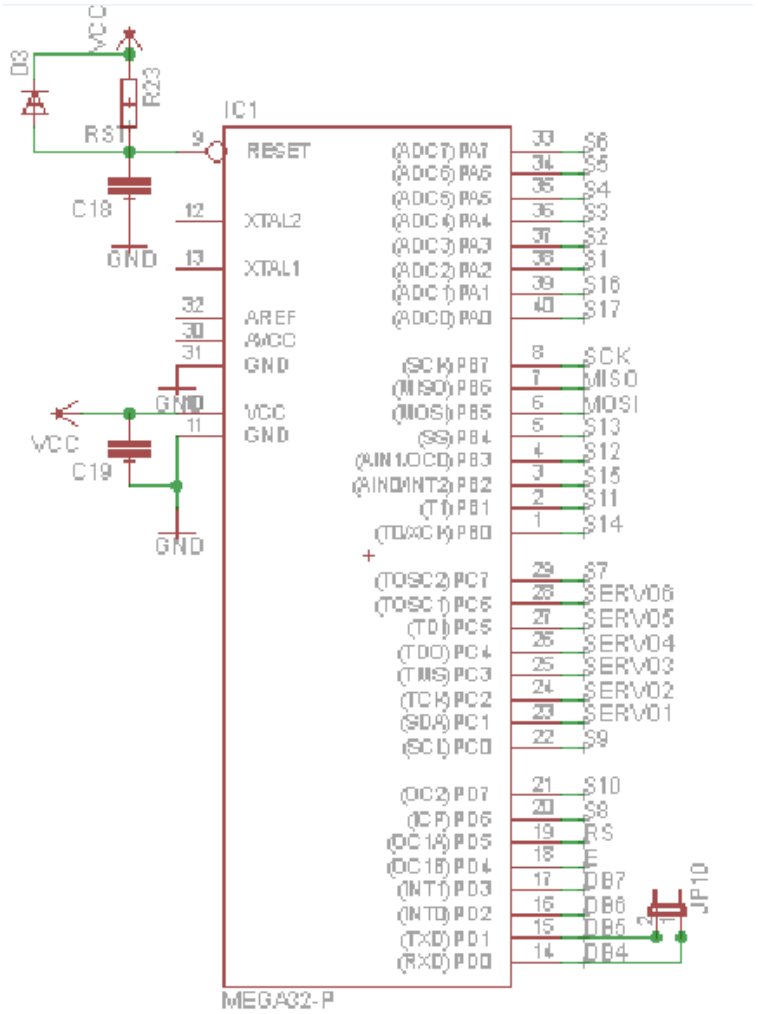
Obr. 11 Schéma zapojení ISP konektoru

Pomocí LED displaye dokážeme nastavovat polohu ramene. Na obrázku 12, je zobrazeno propojení displaye s procesorem.



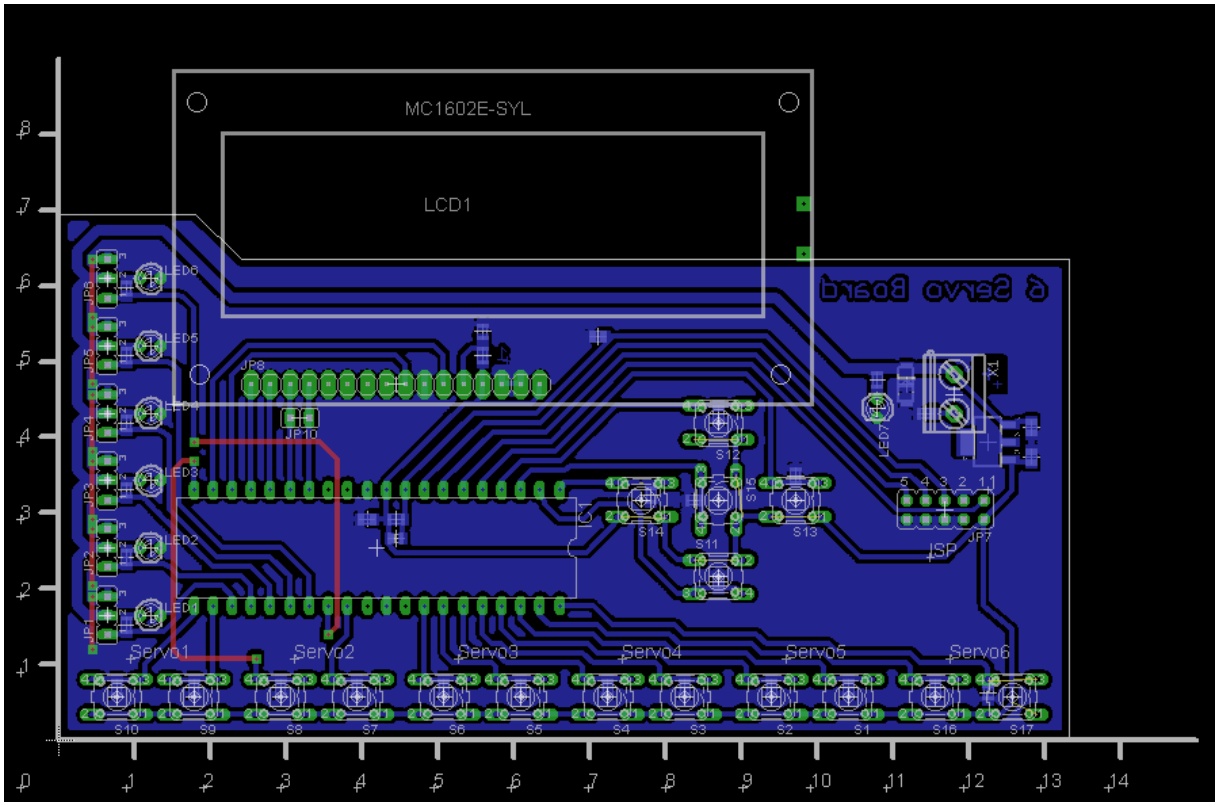
Obr. 12 Schéma zapojení LED displaye[8]

Nejdůležitější částí na desce je procesor ATMega32A, v kterém je uložen program. Procesor také počítá všechny hodnoty, které jsou v programu počítány, posílá nám řídící signály na serva a přijímá signály od tlačítek.[9]

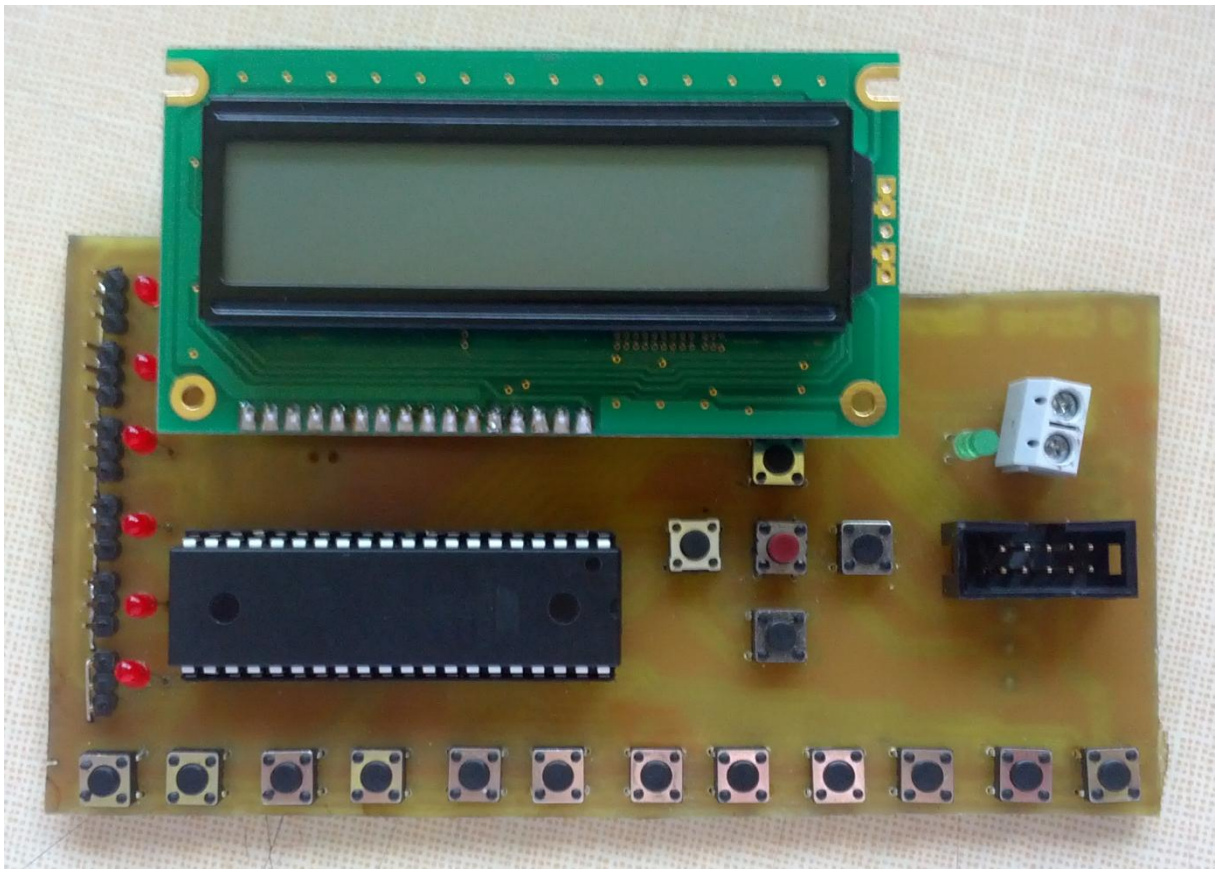


Obr. 13 Schéma zapojení procesoru ATMega32A[9]

Když je vytvořen seznam propojených součástek, musí se vytvořit schéma plošného spoje, které obsahuje vodivé cesty a umístění součástek. Do takto připravené desky se vyvrtají potřebné otvory na součástky. Deska se osadí součástkami a ty se pak připájí. Na *obr. 14* je schéma plošného spoje desky a na *obr. 15* je vyfocena již deska hotová.



Obr. 14 Schéma plošného spoje v Eaglu

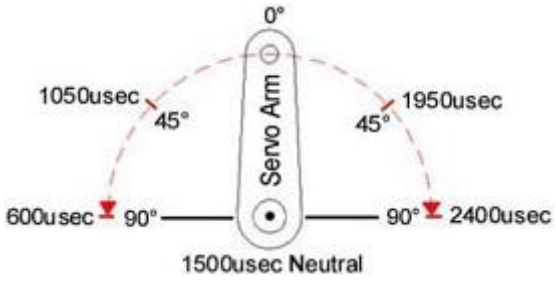


Obr. 15 Fotografie hotové desky

5 Teorie řízení serv

5.1 Řízení servomechanizmů

Servo se skládá z elektromotoru, převodovky a řídící elektroniky. Všechna serva mají pracovní rozsah od -90° do 90°. Abychom servo pootočili o -90°, musíme na něj poslat signál o požadované šířce 600µsec(PWM). Když na servo pošleme požadovaný signál, servo se nám natočí do požadovaného úhlu. Velikosti šířek pro různé natočení serva najdete na *obr. 16.* Signál musí být nepřetržitý, pokud ustane, uvolní se zpětná vazba serva a není nijak zajištěna správná poloha výstupní hřídele.[5]



Obr. 16 Velikosti šířek signálu pro různá natočení[5]

5.2 Přímá a inverzní kinematika

Přímá kinematika nám nastavuje velikosti úhlů na jednotlivých servech a z toho stanoví pozici koncového bodu (efektoru). Ale když známe polohu a orientaci koncového bodu a z toho stanovujeme velikost jednotlivých úhlů serv, jedná se o inverzní kinematiku. Přímá kinematika se dá počítat pomocí základních goniometrických funkcí.

5.3 Goniometrické funkce

Na desce nastavíme hodnoty X, Y, Z, T (pozice efektoru, náklonu zápěstí a popřípadě efektor zavřený nebo otevřený). Program pomocí vytvořených početních funkcí a Heronova vzorce vypočte úhly jednotlivých serv. Tyto úhly jsou převedeny na odpovídající šířku signálu v ms a posléze je signál o požadované šířce poslán do serva.

6 Software

Kompletní výpis programu a vývojové diagramy jsou přiloženy v přílohách. Program pro jednoduchost a názornost využívá vhodně pojmenovaná makra, která odkazují na funkce, např. Servo1_Plus, které zajišťuje zvýšení úhlu na servu1.Left je makro, na posouvání se po display do leva. OK_Button nám testuje tlačítko OK, pomocí níže popsané funkce Port. To nám umožňuje jednoduché definování, které piny procesoru náleží jakému tlačítku na desce už v úvodu programu. Zde uvedené příklady, jsou obecně stejné ve všech případech.

6.1 Testování tlačítek

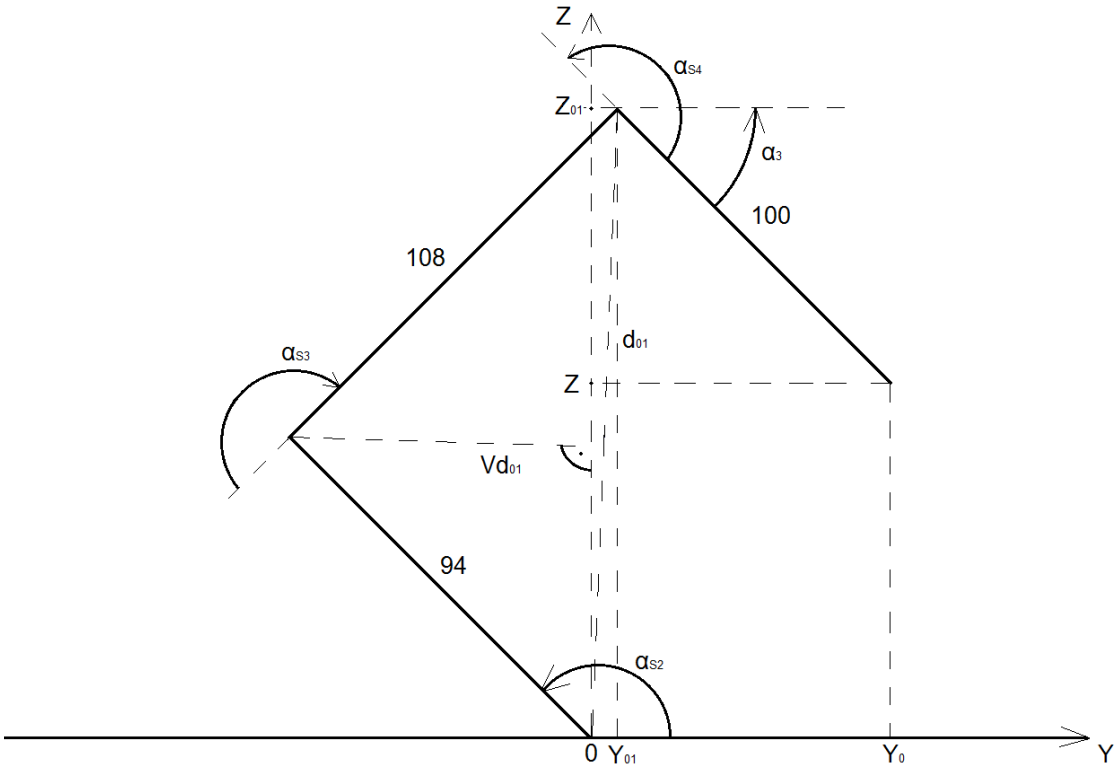
Testování tlačítek zajišťuje funkce char Port(char port, unsigned char Pin), vrací hodnotu typu char, která vypovídá o tom,zda-li je příslušný pin v logické 1 nebo 0. Vstupní proměnné port a pin udávají, na kterém portu (A, B, C a D) je umístěn pin, který chceme testovat. Tato funkce je zároveň prevencí před nedokonalostmi tlačítek – jejich zákmity, které jsou kratší jak 1ms.

6.2 Obsluha LCD

Inicializace LCD displaye s řadičem HD44780 je naprogramována podle zdroje[5]. Obslužné funkce zajišťují výpis řetězců a čísel.

6.3 Výpočty úhlů serv

Přepočet zadaných souřadnic a velikost úhlu, pod kterým se efektor dostane do požadovaných souřadnic, na úhly jednotlivých servr zajišťuje funkce void Calc_Servo_Angles(). Nemá sice návratovou hodnotu, avšak pozměňuje globální proměnné, které uchovávají hodnoty úhlů jednotlivých serv. Na *obr. 17* je znázorněna pozice proměnných.



Obr. 17 Znázornění modelu v osách Y a Z pro výpočet úhlů

V rovnicích z kapitoly 6.3 je naznačen výpočet úhlů jednotlivých serv.

$$\alpha_{S1} = tan\frac{Y}{X} \quad (1.1)$$

$$Y_0 = \sqrt{X^2 + Y^2} \quad (1.1)$$

$$Y_{01} = Y_0 - (100 * cos\ \alpha_3) \quad (1.3)$$

$$Z_{01} = Z - (100 * sin\ \alpha_3) \quad (1.4)$$

$$d_{01} = \sqrt{Y_{01}^2 + Z_{01}^2} \quad (1.5)$$

$$s = \frac{94 + 108 + d_{01}}{2} \quad (1.6)$$

$$S = \sqrt{s * (s - 94) * (s - 108) * (s - d_{01})} \quad (1.7)$$

$$Vd_{01} = \frac{2 * S}{d_{01}} \quad (1.8)$$

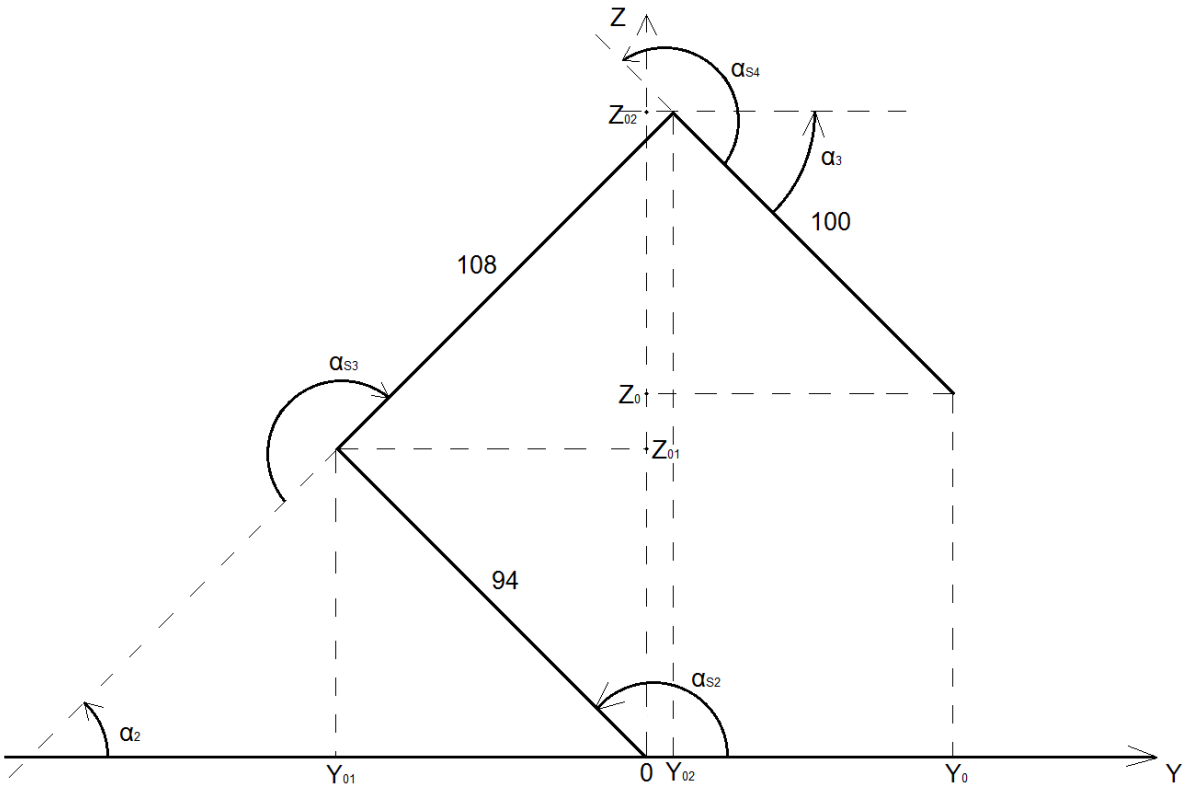
$$\alpha_{S2} = sin^{-1}\frac{Z_{01}}{d_{01}} + sin^{-1}\frac{Vd01}{94} \quad (1.9)$$

$$\alpha_{S3} = 270 - cos^{-1}\frac{Vd_{01}}{94} + cos^{-1}\frac{Vd01}{108} \quad (1.10)$$

$$\alpha_{S4} = \alpha_3 - \alpha_{S2} + 90° - \alpha_{S3} - 90° \quad (1.11)$$

6.4 Výpočet souřadnic a úhlu efektoru

Ze zadaných úhlů serv, nám funkce void Calc_Coordinates() vypočítá souřadnice X, Y, Z a úhel T. Na *obr. 18* je znázorněna pozice proměnných.



Obr. 18 Znázornění modelu v osách Y a Z pro výpočet souřadnic

V rovnicích z kapitoly 6.4 je naznačen výpočet jednotlivých souřadnic a úhlu efektoru.

$$Y_{01} = 94 * cos\ \alpha_{S2} \quad (6.1)$$

$$Z_{01} = 94 * sin\ \alpha_{S2} \quad (6.2)$$

$$Y_{02} = Y_{01} + 108 * cos(\alpha_{S2} + 90^{\circ} - \alpha_{S3}) \quad (6.3)$$

$$Z_{02} = Z_{01} + 108 * sin(\alpha_{S2} + 90^{\circ} - \alpha_{S3}) \quad (6.4)$$

$$Y_0 = Y_{02} + 100 * cos(\alpha_{S2} + 90° - \alpha_{S3} - 90° + \alpha_{S4}) \quad (6.5)$$

$$Z_0 = Z_{02} + 100 * sin(\alpha_{S2} + 90° - \alpha_{S3} - 90° + \alpha_{S4}) \quad (6.6)$$

$$X = \sqrt{\frac{Y_0^2}{(tan\ \alpha_{S1})^2} + 1} \quad (6.7)$$

$$Y = \sqrt{Y_0^2 - X^2} \quad (6.8)$$

$$Z = Z_0 \quad (6.9)$$

$$\alpha_3 = \alpha_{S2} - \alpha_{S3} + \alpha_{S4} \quad (6.10)$$

6.5 Přepočet velikosti úhlu do šířky pulzu

Funkce void Angles_to_Pulses(), nám přepočítá velikost úhlu na šířku pulzu tak, že 1° odpovídá 10µsec, počínaje 600µsec pro 0°. Zároveň ošetřuje, aby se na serva nedostali pulzy o délce menší než 600µsec a vetší než 2400µsec.

Příklad úryvku programu pro servo 1:

```
if(Servo1_Angle>180)
	Servo1_Angle=180;
if(Servo1_Angle<0)
	Servo1_Angle=0;
Servo_Pulses[1] = round(600+(10*Servo1_Angle));
```

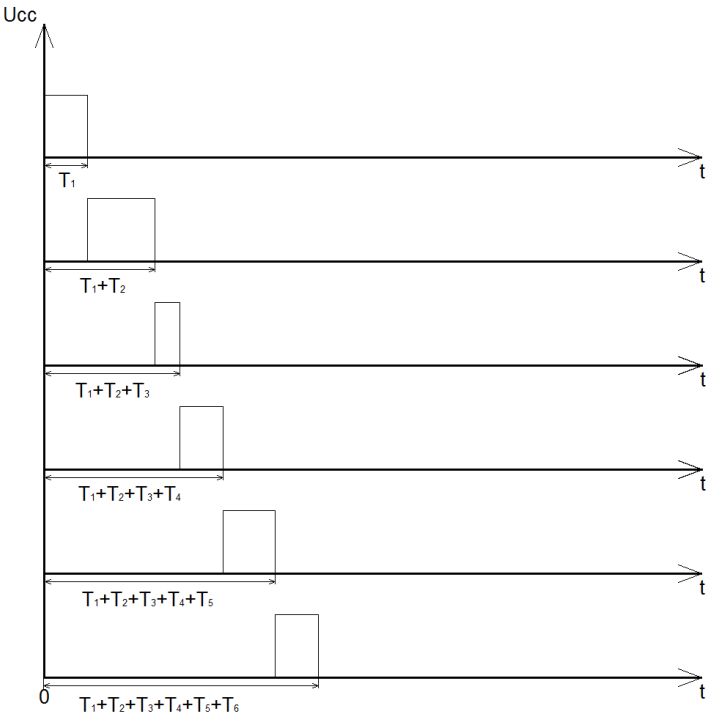
6.6 Odesílaní pulzů na serva

Paralelně s během programu, inkrementuje 16bitový čítač 1 registr TCNT1 každou 1µsec (procesor je taktován vnitřním oscilátorem na 8 MHz, dělička timeru je nastavena na 8). Při každém přerušení od shody registru OCR1A s timerem 1, se vyvolá obsluha, ve které program zajišťuje nastavování a nulování řídících pinů všech servr. Tyto piny jsou hardwarově umístěny tak, aby bylo možné jejich ovládání pouze pomocí bitového posunu a maskování portu. Pokud program odešle pulz na všechna serva, čeká do konce periody (proměnná period), poté začne odesílat pulzy na všechna serva od začátku.

Výpis obsluhy přerušení:

```
ISR(TIMER1_COMPA_vect)
{

	if (TCNT1>=Servo_Pulses[0])
		TCNT1 = 0;

	PORTC &= (0<<Servo_index);
	Servo_index++;
	PORTC |= (1<<Servo_index);

	if (Servo_index>=7)
	{
		Servo_index=0;
```

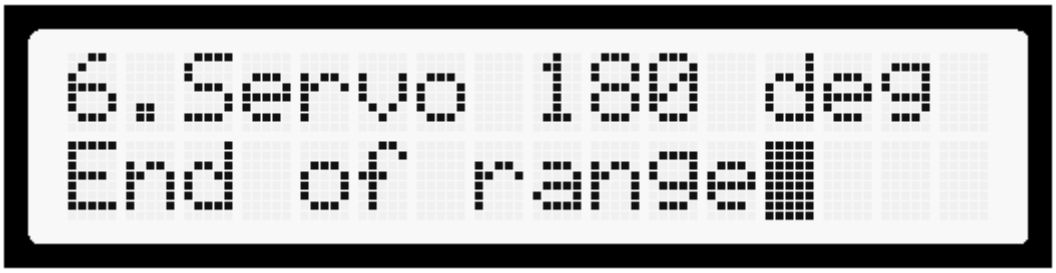
```
            OCR1A = Servo_Pulses[Servo_index];
      }
      else
            OCR1A = TCNT1 + Servo_Pulses[Servo_index];
}
```



Obr. 19 Princip spínání jednotlivých servr

6.7 Kontrola rozsahů

Program má funkce jak na kontrolování vypočtených, nebo zadaných jak souřadnic, tak úhlů. To zajišťují funkce char Check_Angles_Range() a funkce char Check_Coordinates_Range(). Pokud program zaznamená hodnotu větší než je dovolená velikost, vypíše na LCD varovný text, pomocí makra se vstupní proměnou Out_Of_Range(_OoR_Dimension). Do vstupní proměnné je předáván text z prvního řádku. Druhý řádek je vždy stejný. Tento text je zobrazen po dobu tří vteřin. I v této funkci stejně jako ve funkci pro přepočet úhlů na šířku pulzu je provedena kontrola a korekce rozsahu.



Obr. 20 Zobrazení varovného textu[7]

7 Závěr

V této práci jsem se zabýval problematikou řízení manipulátoru. Manipulátory jsou v dnešní době velice rozšířené a to zejména v automobilovém průmyslu, lékařství a dalších odvětvích. Zavedení automatické výroby nám sníží náklady a zefektivní výrobní proces.

Cílem mé bakalářské práce bylo vytvoření řídícího algoritmu, zhotovení řídící jednotky a implementace navrženého algoritmu do vyrobené řídící desky. Vyrobená deska není tak univerzální jako jiné desky na trhu, je vyrobena přímo pro AL5A. Proto při ovládání jiného robota naší deskou by nemuselo vše fungovat správně. Kompatibilita s jinými zařízeními nebyla testovaná.

Navržená deska obsahuje ovládací prvky (tlačítka), což většina desek neobsahuje, protože se manipulátor řídí přes PC. Řídící algoritmus umožňuje ovládání pomocí tlačítek tak, že lze nastavovat spodními dvanácti tlačítky nastavení úhlu jednotlivých serv a horní pěticí tlačítek se lze pohybovat v souřadnicích X, Y, Z a T, kde T je úhel, pod kterým se efektor dostává do nastavených souřadnic.

Desku by šlo nadále obohatit o vhodněji zvolené pozice tlačítek nebo zmenšení rozměrů desky tím, že by se lépe umístil display. Kontrast displaye je dán na pevno daný děličem, zde by byl vhodnější potenciometr, který by reguloval svit displaye. Dále by se pak deska mohla obohatit o označená tlačítka z důvodu lepší přehlednosti. Další verze desky by se dala vylepšit o konektor pro Joystick, který by ovládal manipulátor. Deska neumí komunikovat s PC, i když má vyvedeny dva signálové piny (RXD, TXD) na sériovou komunikaci.

Rychlejší algoritmus programu výpočtu a posílání pulzů na serva by mohl zlepšit plynulost robota. Frekvence taktu by zvýšila četnost testování tlačítek a bylo by častěji aktualizováno jednorozměrné pole, z kterého si registr OCR1A bere hodnoty. Tuto frekvenci jsem z důvodu nutnosti použití externího oscilátoru nezvolil.

Seznam literatury a informačních zdrojů

[1] ŠPALEK, Jaroslav. *Informační a řídicí systém modelu manipulátoru*. Plzeň, 2009. Diplomová práce. Západočeská univerzita. Fakulta aplikovaných věd. Vedoucí práce Miloš Fetter.

[2] KOHOUT, Luděk. *Roboty a manipulátory* [online]. [cit. 2013-02-05]. Dostupné z: http://www.edumat.cz/texty/Roboty_manipulatory.pdf

[3] ŠOLC, František, ŽALUD Luděk. *Robotika* [online]. Brno: Vysoké učení technické v Brně, 2002, [cit. 2013-01-06]. Dostupné z: http://matescb.skvorsmalt.cz/robotika_kybernetika/VUT_Brno_Robotika.pdf

[4] LYNXMOTION, Inc. *Lynxmotion Robot Kits* [online]. [cit. 2013-01-08]. Dostupné z: http://www.lynxmotion.com/default.aspx

[5] SERVOCITZ. Inc. *ServoCity* [online]. [cit. 2013-01-08]. Dostupné z: http://www.servocity.com/

[6] AND-TECH. Inc. *And-Tech* [online]. [cit. 2013-01-22]. Dostupné z: http://and-tech.pl/

[7] AVTANSKI. Inc. *Avtanski* [online]. [cit. 2013-02-04]. Dostupné z: http://avtanski.net/projects/lcd/

[8] WEISSAR, Petr. *Cvičení z předmětu: KAE/MPP* [online]. 2011/2012, s. 59-67 [cit. 2013-02-08].

[9] Atmel. ATMEL CORPORATION. *ATMega32A datasheet* [online]. [cit. 2013-06-06]. Dostupné z: http://www.atmel.com/Images/doc8155.pdf

[10] KOTRČ, Václav. *Řízení zbraně pro airsoft*. Plzeň, 2012. Bakalářská práce. Západočeská univerzita. Fakulta elektrotechnická. Vedoucí práce Petr Weissar.

Přílohy

Příloha A – Zdrojový kód programu

```
# define F_CPU 8000000UL

#include <avr/io.h>
#include <math.h>
#include <avr/interrupt.h>
#include <util/delay.h>

unsigned char LCD_X = 2;


volatile int Servo_Pulses[7];
volatile char Servo_index = 0;

short int X, Y, Z;
volatile double d01, Servo1_Angle, Servo2_Angle, Servo3_Angle, Servo4_Angle,
Servo5_Angle, Servo6_Angle, Arm_Angle;

volatile double Period = 20000;

#define Servo1_Plus {Servo1_Angle++;if(Check_Angles_Range()){ Angles_to_Pulses();
Calc_Coordinates();}LCD_X = 2;LCD_Write_Angle_Signs();}
#define Servo1_Minus {Servo1_Angle--;if(Check_Angles_Range()){Angles_to_Pulses();
Calc_Coordinates();}LCD_X = 2;LCD_Write_Angle_Signs();}
#define Servo2_Plus {Servo2_Angle++;if(Check_Angles_Range()){ Angles_to_Pulses();
Calc_Coordinates();}LCD_X = 6;LCD_Write_Angle_Signs();}
#define Servo2_Minus {Servo2_Angle--;if(Check_Angles_Range()){Angles_to_Pulses();
Calc_Coordinates();}LCD_X = 6;LCD_Write_Angle_Signs();}
#define Servo3_Plus {Servo3_Angle++;if(Check_Angles_Range()){ Angles_to_Pulses();
Calc_Coordinates();}LCD_X =10;LCD_Write_Angle_Signs();}
#define Servo3_Minus {Servo3_Angle--;if(Check_Angles_Range()){Angles_to_Pulses();
Calc_Coordinates();}LCD_X =10;LCD_Write_Angle_Signs();}
#define Servo4_Plus {Servo4_Angle++;if(Check_Angles_Range()){ Angles_to_Pulses();
Calc_Coordinates();}LCD_X =14;LCD_Write_Angle_Signs();}
#define Servo4_Minus {Servo4_Angle--;if(Check_Angles_Range()){Angles_to_Pulses();
Calc_Coordinates();}LCD_X =14;LCD_Write_Angle_Signs();}
#define Servo5_Plus {Servo5_Angle++;if(Check_Angles_Range()){
Angles_to_Pulses();LCD_Write_Angle_Signs();}}
#define Servo5_Minus {Servo5_Angle--
;if(Check_Angles_Range()){Angles_to_Pulses();LCD_Write_Angle_Signs();}}
#define Servo6_Plus {Servo6_Angle++;if(Check_Angles_Range()){
Angles_to_Pulses();LCD_Write_Angle_Signs();}}
#define Servo6_Minus {Servo6_Angle--
;if(Check_Angles_Range()){Angles_to_Pulses();LCD_Write_Angle_Signs();}}
#define Left {LCD_Write_Coordinate_Signs();LCD_X=LCD_X-4; if(LCD_X > 14) LCD_X=14;
LCD_GotoXY(LCD_X,0);}
#define Right {LCD_Write_Coordinate_Signs();LCD_X=LCD_X+4; if(LCD_X > 14) LCD_X=2;
LCD_GotoXY(LCD_X,0);}
#define Up {  if(LCD_X == 2){if(Check_Coordinates_Range()) {X++; Calc_Servo_Angles();
Angles_to_Pulses();}LCD_Write_Coordinate_Signs();}            if(LCD_X == 6)
{if(Check_Coordinates_Range()) {Y++; Calc_Servo_Angles();Angles_to_Pulses();}
LCD_Write_Coordinate_Signs();}           if(LCD_X == 10) {if(Check_Coordinates_Range())
{Z++; Calc_Servo_Angles();Angles_to_Pulses();} LCD_Write_Coordinate_Signs();}
       if(LCD_X == 14) {if(Check_Coordinates_Range()) {Arm_Angle++;
Calc_Servo_Angles();Angles_to_Pulses();} LCD_Write_Coordinate_Signs();}}
#define Down {       if(LCD_X == 2){if(Check_Coordinates_Range()) {X--;
Calc_Servo_Angles(); Angles_to_Pulses();}LCD_Write_Coordinate_Signs();}
       if(LCD_X == 6) {if(Check_Coordinates_Range()) {Y--;
```

```
Calc_Servo_Angles();Angles_to_Pulses();} LCD_Write_Coordinate_Signs();}
       if(LCD_X == 10) {if(Check_Coordinates_Range()) {Z--;
Calc_Servo_Angles();Angles_to_Pulses();} LCD_Write_Coordinate_Signs();}    if(LCD_X ==
14) {if(Check_Coordinates_Range()) {Arm_Angle--;
Calc_Servo_Angles();Angles_to_Pulses();} LCD_Write_Coordinate_Signs();}}
#define OK { Servo1_Angle = 90;Servo2_Angle = 135;Servo3_Angle = 180;Servo4_Angle =
0;Servo5_Angle = 90;Servo6_Angle = 0;Calc_Coordinates();
Angles_to_Pulses();LCD_Write_Coordinate_Signs();}
#define Out_Of_Range(_OoR_Dimension)
{LCD_WriteStringXY(0,0,_OoR_Dimension);LCD_WriteStringXY(0,1,"End of range    ");
_delay_ms(3000);}

#define OK_Button Port(B,2)                          //OK
#define Left_Button Port(B,0)                        //Left
#define Right_Button Port(B,4)                       //Right
#define Up_Button Port(B,3)                          //Up
#define Down_Button Port(B,1)                        //Down
#define Servo1_Minus_Button Port(D,7)          //Servo1_Minus
#define Servo1_Plus_Button Port(C,0)           //Servo1_Plus
#define Servo2_Minus_Button Port(D,6)          //Servo2_Minus
#define Servo2_Plus_Button Port(C,7)           //Servo2_Plus
#define Servo3_Minus_Button Port(A,7)          //Servo3_Minus
#define Servo3_Plus_Button Port(A,6)           //Servo3_Plus
#define Servo4_Minus_Button Port(A,5)          //Servo4_Minus
#define Servo4_Plus_Button Port(A,4)           //Servo4_Plus
#define Servo5_Minus_Button Port(A,3)          //Servo5_Minus
#define Servo5_Plus_Button Port(A,2)           //Servo5_Plus
#define Servo6_Minus_Button Port(A,1)          //Servo5_Minus
#define Servo6_Plus_Button Port(A,0)           //Servo5_Plus


#define E_Puls {PORTD = (PORTD | 0b00010000); _delay_us(1); PORTD = (PORTD &
0b11101111);}
#define Set_Data(_Sending_Data)   {PORTD = (_Sending_Data & 0x0f);}

#define Prikaz 0
#define Data 1


char Port(char port, unsigned char Pin)
       //test na stav pinu z portu s osetrenim zakmitu
{
       unsigned char DetekceZakmitu[4];
       unsigned char Stav = 2;
       unsigned char IndexZakmitu;

       #define A 'A'
       #define B 'B'
       #define C 'C'
       #define D 'D'


       while (Stav == 2)
       {
              for (IndexZakmitu=0;IndexZakmitu<4;IndexZakmitu++)           //4x udela:
              {
                     switch (port)
                     {
                            case 'A':    DetekceZakmitu[IndexZakmitu]=(PINA&(1<<Pin));
       //orotuje jednicku na pozici pozadovaneho pinu odmaskuje port B a do pole
DetekceZakmitu ulozi nulovost/nenulovost pinu
                            break;
```

```
                        case 'B':     DetekceZakmitu[IndexZakmitu]=(PINB&(1<<Pin));

                        break;
                        case 'C':     DetekceZakmitu[IndexZakmitu]=(PINC&(1<<Pin));

                        break;
                        case 'D':     DetekceZakmitu[IndexZakmitu]=(PIND&(1<<Pin));

                        break;
                }
                        _delay_us(250);
                                              //cekej 3ms mezi kazdym testovanim
            }

            if (DetekceZakmitu[0] && DetekceZakmitu[1] && DetekceZakmitu[2] &&
DetekceZakmitu[3])  //pokud se rovnaji 1, vrat 1
                  Stav = 1;
            else if (!DetekceZakmitu[0] && !DetekceZakmitu[1] && !DetekceZakmitu[2]
&& !DetekceZakmitu[3]) //pokud se rovnaji 0, vrat 0
                        Stav = 0;
                  else
                        Stav = 2;                  //pokud se rovnaji 2, opakuj
      }

      return(Stav);
}


void LCD_Cmd(unsigned char Cmd_Data, unsigned char Typ)
{
      unsigned char Mask_Data = Cmd_Data & 0xf0;
      Mask_Data >>= 4;
      Set_Data(Mask_Data);
      if (Typ==Data)
      PORTD = (PORTD | 0b00100000);
      else
      PORTD = (PORTD & 0b11011111);

      E_Puls;
      _delay_us(100);

      Mask_Data = Cmd_Data & 0x0f;
      Set_Data(Mask_Data);
      if (Typ==Data)
      PORTD = (PORTD | 0b00100000);
      else
      PORTD = (PORTD & 0b11011111);

      E_Puls;
      _delay_us(100);
}
void LCD_Init()
{
      _delay_ms(20);

      Set_Data(0x03);
      E_Puls;
      _delay_ms(10);

      Set_Data(0x03);
      E_Puls;
      _delay_ms(10);
```

```
	Set_Data(0x03);
	E_Puls;
	_delay_ms(10);

	Set_Data(0x02);
	E_Puls;
	_delay_ms(40);

	LCD_Cmd(0x2c, Prikaz);
	LCD_Cmd(0x0f, Prikaz);
	LCD_Cmd(0x01, Prikaz);
	_delay_ms(2);
	LCD_Cmd(0x06, Prikaz);

	LCD_Cmd(0x80, Prikaz);

}
void LCD_Clr(void)
{
	LCD_Cmd(0b00000001,Prikaz);
}
void LCD_GotoXY(unsigned char x, unsigned char y)
{
	if (y>=1)
	LCD_Cmd(192 + x, Prikaz);			//druhy radek zacina na 0xC0
	else
	LCD_Cmd(128 + x, Prikaz);			//prvni radek zacina na 0x80
}
void LCD_WriteStringXY(unsigned char x, unsigned char y, char *text)
{
	char *cptr;

	LCD_GotoXY(x,y);
	for(cptr = text; *cptr; cptr++)
	LCD_Cmd(*cptr, Data);
}
void LCD_WriteString(char *text)
{
	char *cptr;
	for(cptr = text; *cptr; cptr++)
	LCD_Cmd(*cptr, Data);
}
void LCD_WriteInt(int val,unsigned int field_length)
{
char str[5]={0,0,0,0,0};
int i=4,j=0;
while(val)
{
	str[i]=val%10;
	val=val/10;
	i--;
}
if(field_length==-1)
while(str[j]==0) j++;
else
j=5-field_length;

if(val<0) LCD_Cmd('-', Data);
for(i=j;i<5;i++)
{
	LCD_Cmd(48+str[i], Data);
```

```
}
}
void LCD_WriteIntXY(unsigned char x, unsigned char y, int val, unsigned int
field_length)
{
char str[5]={0,0,0,0,0};
int i=4,j=0;
LCD_GotoXY(x,y);

while(val)
{
	str[i]=val%10;
	val=val/10;
	i--;
}
if(field_length==-1)
while(str[j]==0) j++;
else
j=5-field_length;

if(val<0) LCD_Cmd('-', Data);
for(i=j;i<5;i++)
{
	LCD_Cmd(48+str[i], Data);
}
}

char Check_Angles_Range()
{
	char Checked = 1;
	if (Servo6_Angle>180)
	{
		Out_Of_Range("6.Servo 180 deg");
		Checked = 0;
		Servo6_Angle = 180;
		LCD_Write_Angle_Signs();
	}
	if (Servo6_Angle<0)
	{
		Out_Of_Range("6.Servo   0 deg");
		Checked = 0;
		Servo6_Angle = 0;
		LCD_Write_Angle_Signs();
	}
	if (Servo5_Angle>180)
	{
		Out_Of_Range("5.Servo 180 deg");
		Checked = 0;
		Servo5_Angle = 180;
		LCD_Write_Angle_Signs();
	}
	if (Servo5_Angle<0)
	{
		Out_Of_Range("5.Servo   0 deg");
		Checked = 0;
		Servo5_Angle = 0;
		LCD_Write_Angle_Signs();
	}
	if (Servo4_Angle>180)
	{
		Out_Of_Range("4.Servo 180 deg");
		Checked = 0;
```

```
        Servo4_Angle = 180;
        LCD_Write_Angle_Signs();
}
if (Servo4_Angle<0)
{
        Out_Of_Range("4.Servo   0 deg");
        Checked = 0;
        Servo4_Angle = 0;
        LCD_Write_Angle_Signs();
}
if (Servo3_Angle>180)
{
        Out_Of_Range("3.Servo 180 deg");
        Checked = 0;
        Servo3_Angle = 180;
        LCD_Write_Angle_Signs();
}
if (Servo3_Angle<0)
{
        Out_Of_Range("3.Servo   0 deg");
        Checked = 0;
        Servo3_Angle = 0;
        LCD_Write_Angle_Signs();
}
if (Servo2_Angle>135)
{
        Out_Of_Range("2.Servo 135 deg");
        Checked = 0;
        Servo2_Angle = 135;
        LCD_Write_Angle_Signs();
}
if (Servo2_Angle<0)
{
        Out_Of_Range("2.Servo   0 deg");
        Checked = 0;
        Servo2_Angle = 0;
        LCD_Write_Angle_Signs();
}
if (Servo1_Angle>180)
{
        Out_Of_Range("1.Servo 180 deg");
        Checked = 0;
        Servo1_Angle = 180;
        LCD_Write_Angle_Signs();
}
if (Servo1_Angle<0)
{
        Out_Of_Range("1.Servo   0 deg");
        Checked = 0;
        Servo1_Angle = 0;
        LCD_Write_Angle_Signs();
}
if (Arm_Angle+90<0)                     //naklon ramene kolmo dolu je 0°
{
        Out_Of_Range("Arm Tilt  0 deg ");
        LCD_Write_Coordinate_Signs();
        Checked = 0;
        Arm_Angle = -90;
}
if (Arm_Angle+90>180)                   //naklon ramene kolmo nahoru je 180°
{
        Out_Of_Range("Arm Tilt 180 deg");
```

```
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              Arm_Angle = 90;
       }
       return(Checked);
}
char Check_Coordinates_Range()
{
       char Checked = 1;
       if (X<-302)
       {
              Out_Of_Range("X        -302 mm");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              X = -302;
       }
       if (X>302)
       {
              Out_Of_Range("X         302 mm");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              X = 302;
       }
       if (Y<0)
       {
              Out_Of_Range("Y           0 mm");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              Y = 0;
       }
       if (Y>302)
       {
              Out_Of_Range("Y         302 mm");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              Y = 302;
       }
       if (Z<-60)
       {
              Out_Of_Range("Z         -60 mm");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              Z = -60;
       }
       if (Z>302)
       {
              Out_Of_Range("Z         302 mm");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              Z = 302;
       }
       if (Arm_Angle+90<0)
       {
              Out_Of_Range("Arm Tilt  0 deg ");
              LCD_Write_Coordinate_Signs();
              Checked = 0;
              Arm_Angle = -90;
       }
       if (Arm_Angle+90>180)
       {
              Out_Of_Range("Arm Tilt 180 deg");
              LCD_Write_Coordinate_Signs();
```

```
		Checked = 0;
		Arm_Angle = 90;
	}
	if((94+108) < d01)
	//omezeni plynouci z herronova vzorce pri pevne danych dvou stranach
	{
		Out_Of_Range("XYZ4 combination");
		LCD_Write_Coordinate_Signs();
	}
	return(Checked);
}

void LCD_Write_Angle_Signs()
{

	LCD_WriteStringXY(0,0,"  1   2   3   4 ");
	LCD_WriteStringXY(0,1,"                 ");

	LCD_WriteIntXY(1,1,(Servo1_Angle),3);
	LCD_WriteIntXY(5,1,(Servo2_Angle),3);
	LCD_WriteIntXY(9,1,(Servo3_Angle),3);
	LCD_WriteIntXY(13,1,(Servo4_Angle),3);
	LCD_GotoXY(LCD_X,0);
}
void LCD_Write_Coordinate_Signs()
{
	char X_Sign, Y_Sign, Z_Sign, Arm_Sign;


		if (X<0)
		X_Sign = "-";
		else
		X_Sign = " ";

		if (Y<0)
		Y_Sign = "-";
		else
		Y_Sign = " ";

		if (Z<0)
		Z_Sign = "-";
		else
		Z_Sign = " ";

		LCD_WriteStringXY(0,0,"  X   Y   Z   T ");
		LCD_WriteStringXY(0,1,"                 ");

		LCD_WriteStringXY(0,1,X_Sign);
		LCD_WriteStringXY(4,1,Y_Sign);
		LCD_WriteStringXY(8,1,Z_Sign);


		LCD_WriteIntXY(1,1,abs(X),3);
		LCD_WriteIntXY(5,1,abs(Y),3);
		LCD_WriteIntXY(9,1,abs(Z),3);
		LCD_WriteIntXY(13,1,Arm_Angle+90,3);
		LCD_GotoXY(LCD_X,0);
}


ISR(TIMER1_COMPA_vect)
{
```

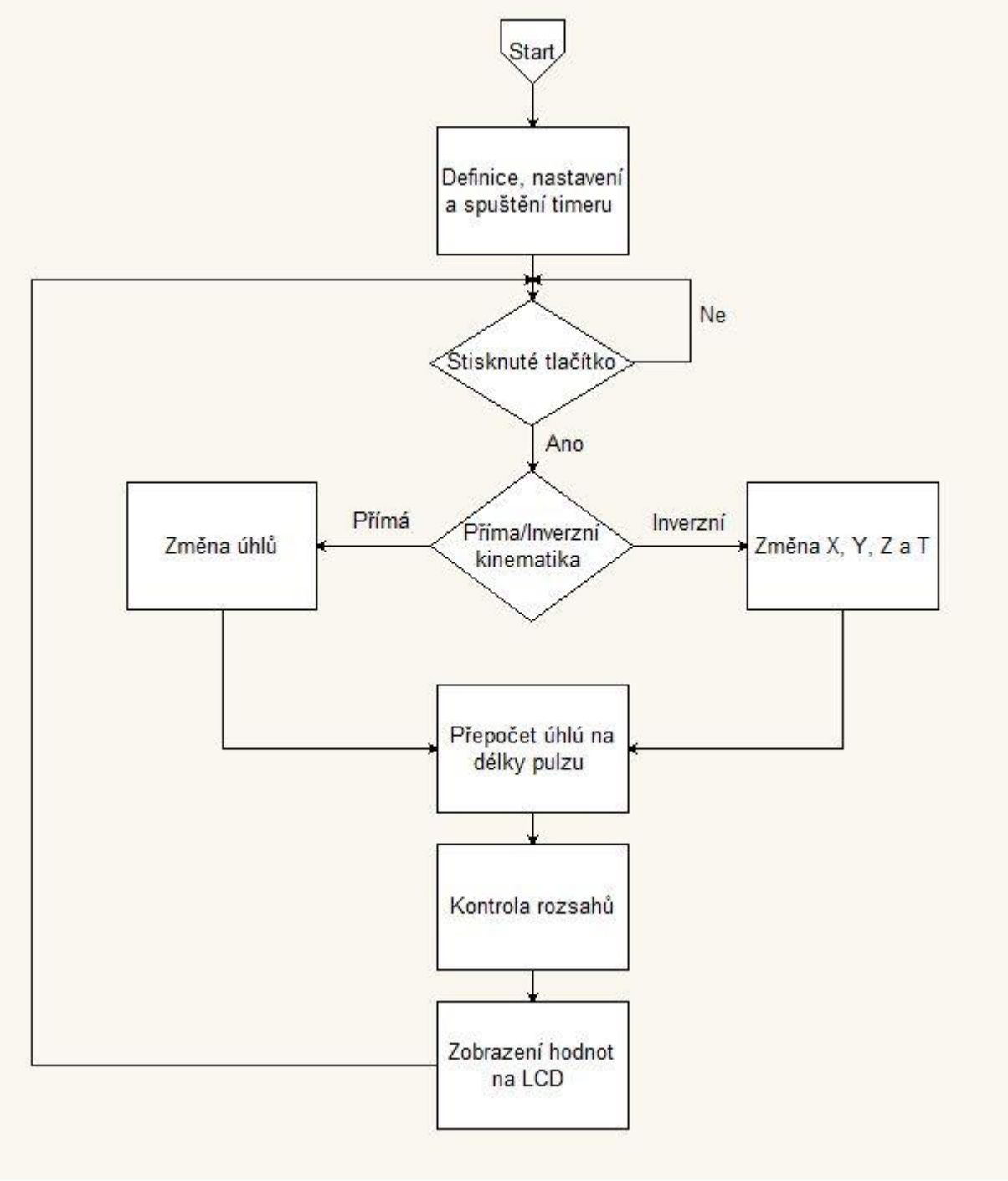
```
	if (TCNT1>=Servo_Pulses[0])			//v Servo_Pulses[0] je ulozena
perioda
		TCNT1 = 0;

	PORTC &= (0<<Servo_index);			//shod predchozi pin
	Servo_index++;
	PORTC |= (1<<Servo_index);			//nastav stavajici

	if (Servo_index>=7)				//pokud prijde sesty interrupt
	{
		Servo_index=0;				//zacni od nuly
		OCR1A = Servo_Pulses[Servo_index];	//dojed do cele periody
	}
	else
		OCR1A = TCNT1 + Servo_Pulses[Servo_index];
			//k aktualnimu casu se pricte delka impulsu dalsiho serva
}

double Deg_conv(double Rad)
{
	return((Rad*180)/M_PI);
}
double Rad_conv(double Deg)
{
	return((Deg*M_PI)/180);
}


void Angles_to_Pulses()
{
	Servo_Pulses[0] = Period;				//ulozeni periody
	if(Servo1_Angle>180)
		Servo1_Angle=180;
	if(Servo1_Angle<0)
		Servo1_Angle=0;
	Servo_Pulses[1] = round(600+(10*Servo1_Angle));

	if(Servo2_Angle>135)
		Servo2_Angle=135;
	if(Servo2_Angle<0)
		Servo2_Angle=0;
	Servo_Pulses[2] = round(600+(10*Servo2_Angle));

	if(Servo3_Angle>180)
		Servo3_Angle=180;
	if(Servo3_Angle<0)
		Servo3_Angle=0;
	Servo_Pulses[3] = round(600+(10*Servo3_Angle));

	if(Servo4_Angle>180)
		Servo4_Angle=180;
	if(Servo4_Angle<0)
		Servo4_Angle=0;
	Servo_Pulses[4] = round(600+(10*Servo4_Angle));

	if(Servo5_Angle>180)
		Servo5_Angle=180;
	if(Servo5_Angle<0)
		Servo5_Angle=0;
	Servo_Pulses[5] = round(600+(10*Servo5_Angle));
```

```
	if(Servo6_Angle>180)
		Servo6_Angle=180;
	if(Servo6_Angle<0)
		Servo6_Angle=0;
	Servo_Pulses[6] = round(600+(10*Servo6_Angle));
}

void Calc_Servo_Angles()
{
	volatile double y0,y01,z01,vd01,s,S;


	Servo1_Angle = Deg_conv(atan2(Y,X));
	y0 = sqrt(square(X)+square(Y));
	y01 = y0 - (100*(cos(Rad_conv(Arm_Angle))));
	z01 = Z -  (100*sin(Rad_conv(Arm_Angle)));
	d01 = sqrt(square(y01)+square(z01));

	s = ((94+108+d01)/2);				//aplikace herronova vzorce
	S = sqrt(s*(s-94)*(s-108)*(s-d01));
	vd01 = ((2*S)/d01);				//vypocet vysky z heronova
vzorce

	if((94+108) > d01)				//pokud je zakladna trojuhelniku
vetsi nez delky zbylych dvou stran, trojuhelnik nepujde sestrojit (S by nebylo cislo)
	{
		Servo2_Angle = Deg_conv(asin((z01/d01)))+Deg_conv(asin((vd01/94)));
		Servo3_Angle = 270 -
(Deg_conv(acos((vd01/94)))+Deg_conv(acos((vd01/108))));
		//korekce… pri soucasnem zpusobu namontovani servo nenahne rameno na
mensi uhel nez 90° coz je zaroven 180° pro servo => 270 minus vypocitany uhel
		Servo4_Angle = Arm_Angle - (((Servo2_Angle +90)-Servo3_Angle)-90);
	}

}
void Calc_Coordinates()				// vypocet promennych XYZ z uhlu
serv
{
	volatile double y0,y01,y02,z01,z02,vd01,s,S;	//promenne


	y02 = 94*(cos(Rad_conv(Servo2_Angle)));
	z02 = 94*(sin(Rad_conv(Servo2_Angle)));
	y01 = y02 + (108*(cos(Rad_conv(((Servo2_Angle+90)-Servo3_Angle)))));
	z01 = z02 + (108*(sin(Rad_conv(((Servo2_Angle+90)-Servo3_Angle)))));
	y0 = y01 + (100*(cos(Rad_conv((((Servo2_Angle+90)-Servo3_Angle)-
90)+Servo4_Angle))));
	Z = z01 + (100*(sin(Rad_conv((((Servo2_Angle+90)-Servo3_Angle)-
90)+Servo4_Angle))));
	X = sqrt(square(y0)/square(tan(Rad_conv(Servo1_Angle)))+1);
	Y = sqrt(square(y0)-square(X));
	Arm_Angle = (((Servo2_Angle+90)-Servo3_Angle)-90)+Servo4_Angle;
}


int main(void)
{
	unsigned char Arrow_Button_Hit=0;
	unsigned char Servo_Button_Hit=0;
	unsigned int Servo_Button_Delay, Arrow_Button_Delay;

	DDRA  = 0b00000000;				//Port A0..7 - vstup
```

```
	DDRB &= 0b11100000;                     //Port B0..4 - vstup
	DDRC = 0b01111110;                      //Port C1..6 - vystup, C0,7 vstup
	DDRD = 0b00111111;                      //Port D0..5 - vystup, D6,7 vstup

	Servo1_Angle = 90;                      //inicializacni poloha robota
	Servo2_Angle = 135;
	Servo3_Angle = 180;
	Servo4_Angle = 0;
	Servo5_Angle = 90;
	Servo6_Angle = 0;                       //otevrene chapadlo


	Calc_Coordinates();

	LCD_Init();

	LCD_Write_Coordinate_Signs();
	_delay_ms(2000);
	Angles_to_Pulses();

	OCR1A = Servo_Pulses[0];                //do prvniho interruptu pockej dobu
rovnou delce periody

	TIMSK |= (1<<OCIE1A);                   //povoleni preruseni od compare A
timeru1
	sei();                                  //globalni povoleni preruseni
	TCCR1B |= (1<<CS11);                    //preddelicka 16bit timeru1 na 8 > 1
takt = 1us pri 8Mhz

    while(1)
    {

	   Arrow_Button_Delay++;

	  if((!(PINB&0b00011111))  || (Arrow_Button_Delay > 20))
	   //povol signal pouze pri nabezne hrane stisku (osetreno na zakmity
trvajici do 12ms), pokud je tlacitko sipek drzeno, povol signal jednou za 300 us
	  {
		Arrow_Button_Delay = 0;
		Arrow_Button_Hit = 0;
	   }

	   if ((OK_Button) && (!Arrow_Button_Hit) )       //OK
		  {
			OK;
			Arrow_Button_Hit = 1;
		  }
	   if ((Left_Button) && (!Arrow_Button_Hit))      //Left
		  {
			Left;
			Arrow_Button_Hit = 1;
		  }
	   if ((Right_Button) &&  (!Arrow_Button_Hit))    //Right
		  {
			Right;
			Arrow_Button_Hit = 1;
		  }
	   if (Down_Button)                               //Down
			Down;
	   if (Up_Button)                                 //Up
			Up;
	   if (Servo1_Plus_Button)           //Servo1_Plus
```

```
                Servo1_Plus;
            if (Servo1_Minus_Button)          //Servo1_Minus
                Servo1_Minus;
            if ((Servo2_Plus_Button))         //Servo2_Plus
                Servo2_Plus;
            if ((Servo2_Minus_Button))        //Servo2_Minus
                Servo2_Minus;
            if ((Servo3_Plus_Button))         //Servo3_Plus
                Servo3_Plus;
            if ((Servo3_Minus_Button))        //Servo3_Minus
                Servo3_Minus;
            if ((Servo4_Plus_Button))         //Servo4_Plus
                Servo4_Plus;
            if ((Servo4_Minus_Button))        //Servo4_Minus
                Servo4_Minus;
            if ((Servo5_Plus_Button))         //Servo5_Plus
                Servo5_Plus;
            if ((Servo5_Minus_Button))        //Servo5_Minus
                Servo5_Minus;
            if ((Servo6_Plus_Button))         //Servo5_Plus
                Servo6_Plus;
            if ((Servo6_Minus_Button))        //Servo5_Minus
                Servo6_Minus;

    }
}
```

Příloha B – Vývojový diagram programu



Příloha C – Vývojový diagram timeru