VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

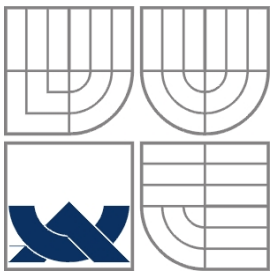
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ŘÍZENÍ ROBOTICKÉHO MANIPULÁTORU FITKITEM

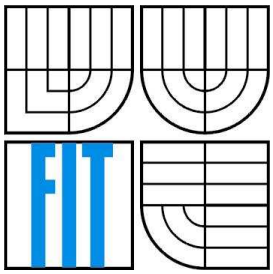
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE LUKÁŠ ŘÍHA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ŘÍZENÍ ROBOTICKÉHO MANIPULÁTORU FITKITEM

ROBOARM CONTROLLED BY FITKIT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ ŘÍHA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RICHARD RŮŽIČKA, Ph.D.

BRNO 2009

Abstrakt

Pojmy *robotika* a *robotické manipulátory* mají v dnešním světě obrovský význam. Tato práce demonstruje ovládání jednoduchého manipulátoru ROB1-3 pomocí FITkitu. K ovládání jsou využity dvě hlavní části FITkitu – mikrokontrolér a FPGA čip. Řízení manipulátoru je možné jak pomocí klávesnice FITkitu, tak pomocí terminálu.

Výkonové členy umožňující pohyb jsou tři serva Hitec HS-311. Serva jsou řízeny pomocí PWM signálu, který generuje mikrokontrolér FITkitu.

Klíčová slova

Robot, robotický manipulátor, FITkit, PWM, servo, ROB1-3, HS-311, VHDL, Jazyk C, FPGA, Mikrokontrolér

Abstract

The terms robotics and roboarms have a huge meaning in the today's world. This thesis describes the simple roboarm ROB1-3 control by the help of FITkit. FITkit has two main parts which are used for control - the microcontroller and the FPGA chip. The controlling of the roboarm is possible by the keyboard of FITkit or by the terminal.

The motion is enabled by three servos Hitec HS-311. The servos are controlled by PWM signal which is generated by microcontroller of the FITkit.

Keywords

Robot, Roboarm, FITkit, PWM, servo, ROB1-3, HS-311, VHDL, C language, FPGA, Microcontroller

Citace

Lukáš Říha: *Řízení robotického manipulátoru FITkitem*. Brno, bakalářská práce, FIT VUT v Brně, 2009.

Řízení robotického manipulátoru FITkitem

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Richarda Růžičky, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Říha
16. května 2009

Poděkování

Poděkování patří všem profesorům podílejících se na mé výuce po celou dobu mého studia na Fakultě informačních technologií VUT v Brně.

Obsah

Úvod

Člověk je neuvěřitelně vynalézavý, pokud jde o to, co nejvíce si ulehčit práci a dopřát si více pohodlí. Už od pravěku vymýšlel finty, aby pracoval méně, ale přesto se měl minimálně stejně dobře jako jeho druzi, kteří pracují více. Přes kolo, pluh, kladku nebo parní stroj se člověk nakonec dopracoval zatím k svým největším pomocníkům, počítači a robotice. A právě robotikou se zabývá i velká část této bakalářské práce.

Slovo *robota* bylo známo již od 17. století, kdy představovalo otrockou práci poddaných. Otroctví bylo postupně ve vyspělých státech rušeno, ale vynalézavý člověk si našel nového otroka, kterým se stal robot. Se slovem *robot* se lidstvo poprvé setkalo ve hře českého dramatika Karla Čapka. V jeho hře R.U.R. představoval robot umělý stroj (android). Karel Čapek tak dal jméno novému fenoménu, který začal používat celý svět. Oficiálně tak vzniklo mezinárodní slovo *robot*.

Prvního robota jménem *Unimate* sestrojil v roce 1954 George Devol. V roce 1961 to byl právě Unimate, který byl jako první nasazen v průmyslu. Jeho úkolem byla, pro člověka nebezpečná, manipulace ze žhavým železem v továrně General Motors. Tímto krokem se v podstatě odstartovala automatice výroby. V roce 2003 byl Unimate právem umístěn do Síně slávy v Pittsburghu.

Po robotovi Unimate se vyvíjeli další roboty a manipulátory. Nejvíce se nové technologie snažila využít vláda USA, zejména pak NASA a ministerstvo obrany.

V dnešní době je již nasazování robotů a robotických ramen zcela běžnou a nepostradatelnou součástí jak masové automatizované výroby, tak i domácího použití. V domácnostech je dnes rovněž samozřejmostí, i když o tom kolikrát ani nevíme, používání vestavěných systémů (embedded system). Vestavěné systémy jsou nedílnou částí této práce. Zejména pak zařízení s názvem FITkit, které obsahuje značné množství komponent a tvoří tak komplexní přípravek pro návrh vestavěných systémů.

V této bakalářské práci se demonstruje ovládání jednoduchého robotického manipulátoru ROB1-3, který je svou konstrukcí a cenou vhodný pro akademické účely. Současně v něm můžeme spatřit i znaky společné vůbec s prvním robotickým manipulátorem Unimate.

Popis mechanických částí manipulátoru se nachází v kapitole 1. Rovněž jsou zde probrány i části, které s manipulátorem neoddělitelně souvisí, jako například pulsně-šířková modulace nebo parametry a využití serv, které manipulátor obsahuje.

Druhá kapitola se zaměřuje na FITkit, pomocí kterého je celý manipulátor ovládán. Jsou zde zmíněny jak jednotlivé komponenty využívané v projektu, tak i techniky, kterými se dá FITkit programovat a využívat (VHDL).

Ve třetí kapitole je pak nastíněna vlastní implementace, se kterou souvisí popis a nastavení časovačů, softwarové generování pulsně-šířkové modulace, ovládání FITkitu pomocí VHDL a další informace vedoucí k úspěšnému zvládnutí celé práce.

Poslední částí této práce je závěr, kde jsou shrnuty dosažené výsledky s požadovanými.

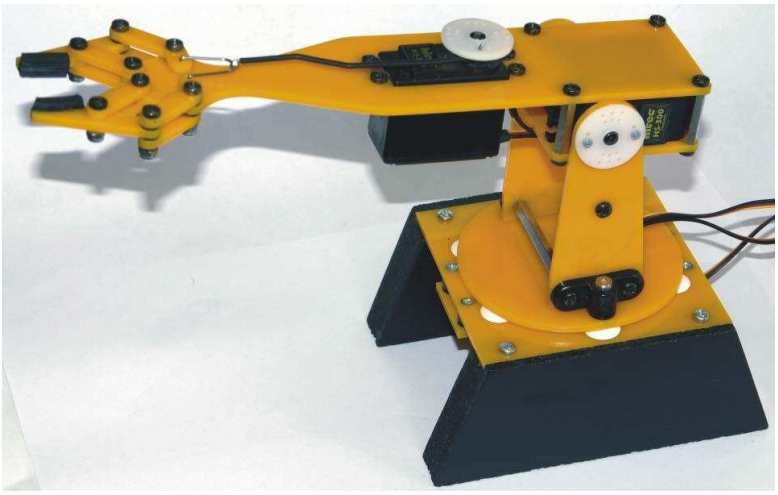
1 Mechanické části

Robotická ramena známe převážně z automobilového průmyslu, kde záběry jejich rychlé a přesné práce jsou doslova fascinující. V této práci je použit manipulátor ROB1-3, kterému se v podstatě věnuje celá tato kapitola. Lépe řečeno se věnuje tomu, jakým způsobem je ROB1-3 konstruován a jakým způsobem lze docílit pohybu.

ROB1-3 je svou příznivou cenou ideální pro akademické účely. Umožňuje pohyb ve třech osách. Manipulátor je zhotovený z levných dílů a i konstrukce naznačuje, že půjde spíše o pochopení a osahání si základních principů a problémů pojících se s ovládáním robotických manipulátorů.

S manipulátorem se musí zacházet velice opatrně. Křehká konstrukce je při neopatrném zacházení velice náchylná k ničení!

1.1 ROB 1-3



obr. 1.1 Robotický manipulátor ROB1-3, zdroj [10]

1.1.1 Konstrukce

Tělo manipulátoru tvoří plastová konstrukce. Plast není nikterak silný a při neopatrném zacházení může snadno dojít k přelomení některé z částí. Rameno se může pohybovat ve dvou směrech s možností úchopu. Pohyb zajišťují tři servomotory s označením HS-311, které jsou v této práci rovněž popsány v kapitole 1.2.

Konstrukce jednotlivých částí byla podřízena použití běžných dílů bez nutnosti většího obrábění. Vyjímkou jsou díly, pro jejichž opracování je zapotřebí lupénkové pily a vrtačky.

Upínací kleština je řešena jako paralelogram skládající se ze šesti shodných ramen jednozvratných a dvou ramen dvojzvratných. Tahem serva za tyto dvojzvratná ramena se kleština rozevírá a svírá. Otočené body ramen kleštiny jsou vytvořeny provlečením šroubů M3, u zadních ramen jsou na šrouby navlečeny podložky IB2, které zde slouží jako distanční podložky a umožňují lehké zasunutí táhla od ovládacího serva. Táhlo serva je vytvořeno z upravené kancelářské sponky. Na dotykových plochách kleštin je nalepeno 15mm pryžového těsnění do oken. Servo pohybu kleštiny je vsazeno do vyříznutého otvoru a upevněno čtyřmi šrouby M3.

Servo pohybu ramene je upnuto mezi desku ramene a přítlačnou desku. Správnou vzdálenost obou desek zajišťují čtyři polyamidové distanční sloupky KDI6M3×20. Jeden otočný bod ramene je vytvořen přímo osou serva; unášející talíř serva je upevněn šrouby M1,6 na bočnici základny. Za servo je vložena destička se zalisovaným distančním sloupkem KDR12, který tvoří druhý otočný bod ramene. Destička s čepem a zadní strana serva je spojena kouskem oboustranné lepící pásky. Tím je zajištěna stabilní poloha serva i pomocné destičky.

Bočnice základny jsou rozepřeny třemi distančními sloupky KDI6M3×50. K otočné desce jsou bočnice upevněny plastovými konzolkami MPJ2621 provlečenými šrouby M3. Otočná hlava klouže po hlavách šesti záslepek F7HP15-08, které jsou zatlačeny do otvorů o průměru 8mm v základní desce.

Pod základní deskou je upevněno čtyřmi šrouby na pomocné desce servo ovládání ramene.

1.1.2 Popis dílů

V příloze 2 je umístěn obrázek, který popisuje součásti pro sestrojení manipulátoru. Při větší zručnosti lze manipulátor sestrojit z laminátu nebo kuprextitu. Pokusy o domácí sestavování bych však příliš nedoporučoval. Manipulátor lze totiž koupit relativně levně jako stavebnici. Seznam součástek je uveden v příloze 1.

Pokud by se přece jen našel kutil, který by si chtěl manipulátor zkonstruovat sám, je mu věnována příloha 2, kde nalezne již zmiňované detailní nákresy jednotlivých dílů.

1.2 Servo

Serva jsou miniaturní motůrky dosahující napříč své velikosti velkého výkonu. Výkon pak přenáší rovnoměrně do mechanické zátěže. Z toho plyne, že čím méně servo-motor zatížíme, tím menší bude i jeho spotřeba energie. Serva můžeme dělit na výkonové (velký kroutící moment) nebo rychlostní.

Poloha hřídele servomotoru je zjišťována elektricky pomocí potenciometru. Toto řešení se však příliš nedoporučuje. V dražších servech se namísto potenciometrů používá fotoelektrický snímač (encoder), rozkladač (selsyn) nebo optické snímání pomocí kódového kotoučku. Signál snímače

polohy je přiveden pomocí zpětné vazby na regulátor, který porovnává skutečnou polohu motoru s žádanou polohou. Na základě rozdílu žádané a skutečné polohy regulátor řídí měnič a tak nastavuje motor na požadovanou polohu.

Některá serva je taky možno upravit pro trvalé otáčení. PWM signálem se pak neřídí poloha výstupního hřídele, ale rychlost a směr jeho otáčení. Takto upravená serva se používají např. v robotech, kde je vyžadován pohon kol.

1.2.1 Servo HS-311

Servomotor s označením HS-311 tvoří základní pilíř standardních serv pro nejširší použití, zejména pak v modelářské technice. Jde o jedno z nejprodávanějších a nejoblíbenějších serv na trhu, a to hlavně díky své nízké ceně, která se v současné době pohybuje kolem 250 Kč za kus.

Ze serva HS-311 jsou vyvedeny tři vodiče. Červený vodič slouží k připojení napájení serva (+Vcc). Na černý vodič je připojená zem. Zbývající žlutý vodič je řídící, kam se připojuje generovaný PWM signál.

V manipulátoru ROB1-3 jsou použita tři serva HS-311 pro ovládání pohybu do tří směrů. Serva se ovládají pomocí pulsně šířkové modulace. Nutno dodat, že servo pro ovládání výšky ramena manipulátoru je takřka stále zatěžováno. Způsobeno je to vlastní hmotností ramene. Servo se snaží udržovat zadanou výšku a musí tak překonat minimálně zmíněnou hmotnost ramene. Toto se projevuje nepřetržitým chrčením serva v zátěži. Celý manipulátor pak při spuštění může vyvolávat dojmy nesprávného zacházení a mást tak uživatele.

1.2.1.1 Parametry serva HS-311

Rozměry:	40×20×36,5 mm
Provozní napětí:	4.8 - 6.0V
Rychlost výchylky:	0.19s/60° při 4.8V
	0.15s/60° při 6.0V
Točivý moment:	42 oz/s (3 kg/cm) při 4.8V
	49 oz/s (4,5 kg/cm) při 6.0V
Operační úhel:	180° (některé typy i 360°)
Typ převodovky:	Nylon
Hmotnost:	43g
Způsob řízení:	PWM signálem
Perioda pro PWM:	15-25 ms
Doba kladného pulsu:	0,5-2,5 ms
Cena:	240 ± 30 Kč (platné 30.2.2009)
Výrobce:	Hitec

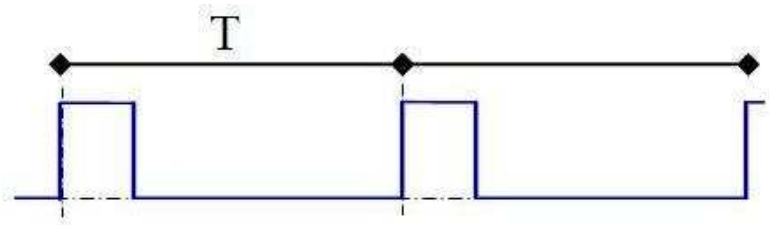
1.2.2 PWM

Pulsně šířková modulace (**P**ulse **W**idth **M**odulation) je diskrétní modulace pro přenos analogového signálu pomocí binárních hodnot. Hodnota analogového signálu je přenášena pomocí střídy. Hodnota střídy nám udává kolik procent z času periody signálu je signál v '1'. Na obrázku 1.2 je znázorněn PWM signál se střídou 20%.

PWM vzniklo z mnoha důvodů. Jedním z nich bylo zpřesnění přenášeného signálu po vodiči (například telefonní komunikace). Dalším důvodem byla kontrola množství přenášeného výkonu do zátěže.

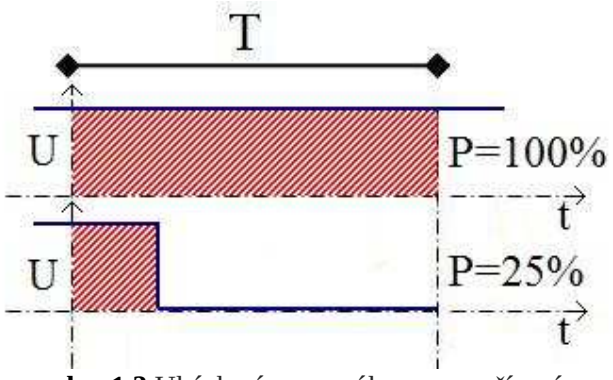
Přenášení komunikace analogovou hodnotou po vodiči není bezpečné. Jednak existuje riziko, že vodič bude vystaven parazitním jevům, jako je indukčnost nebo kapacita. Dále je třeba brát v úvahu, že vodič sám o sobě tyto parazitní jevy obsahuje. Pokud bychom toto nebrali v potaz a serva se pokoušeli řídit např. hodnotou napětí, mohlo by nastat zkreslení napěťové úrovně a tím i nepřesný pohyb manipulátoru.

Pro přenos PWM signálu nemusí sloužit pouze binární hodnota napětí. Lze rovněž použít i hodnotu proudu či světelný tok, který je vzhledem k parazitním jevům nejspolehlivější a nejodolnější.



obr. 1.2 Ukázka PWM se střídou 20%

PWM se ukázalo efektivní i v jiném směru, kterým je ekonomičnost. Příkladem budiž obyčejná žárovka. Na obrázku 1.3 je znázorněn výkon na žárovce.



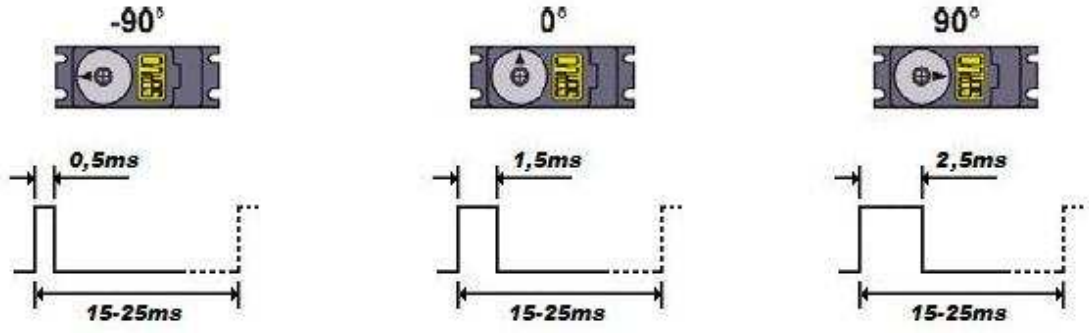
obr. 1.3 Ukázka úspory výkonu na zařízení

Pokud vezmeme v úvahu, že drát žárovky stačí žhavit 25% času s frekvencí elektrické sítě 50Hz aby si udržela svou intenzitu svítivosti, můžeme pomocí pulsně šířkové modulace dosáhnou značných úspor energie. Což je nejen ekonomické, ale v dnešní době hlavně ekologické.

1.2.3 Řízení HS-311 pomocí PWM

Požadovaná poloha výstupního hřídele je do řídící elektroniky serva zadávána jako PWM signál s opakovací frekvencí 50Hz. Na obrázku 1.4 je znázorněna poloha serva v závislosti na střídě.

Signál musí být do serva vysílán nepřetržitě. Pokud ustane, uvolní se zpětná vazba serva a není nijak zajištěna správná poloha výstupního hřídele.



obr 1.4 Vztah mezi polohou hřídele a aktivní dobou PWM

Periodu PWM signálu můžeme volit mezi 15-25 milisekundami. Možnost upravit si délku periody PWM signálu dle vlastních potřeb je užitečné a dává to programátorovi dostatečnou volnost při tvorbě řídícího programu. Doba aktivní úrovně periody PWM signálu je pevně daná rozmezím 0,5-2,5ms a odpovídá rozsahu 0-180°.

1.2.4 Jemnost pohybu serva

Výchozím krokem každé implementace by mělo být zvážení rychlosti a přesnosti pohybu v závislosti na použití manipulátoru. Je jasné, že lékařský manipulátor musí být přesný, ale rychlost vyžadována nebude. Naopak nasazení manipulátoru v průmyslu bude vyžadovat spíše rychlost než přesnost v řádech μm. Budeme-li mít kvalitní servo, můžeme docílit kompromisu - jak rychlosti, tak i nezbytné přesnosti.

Použité servo HS-311 nemá bohužel dostatečně jemný krok. Při použití serva HS-311 je vhodné uživateli umožnit měnit rychlost v závislosti na aktuální potřebě. Pokud bychom přece jenom chtěli použít přesnější nebo silnější servomotor u manipulátoru ROB1-3, není problém nahradit servo HS-311 jiným modelem. Ve stejné modelové řadě nabízí výrobce hned několik serv s lepšími parametry a totožnými rozměry.

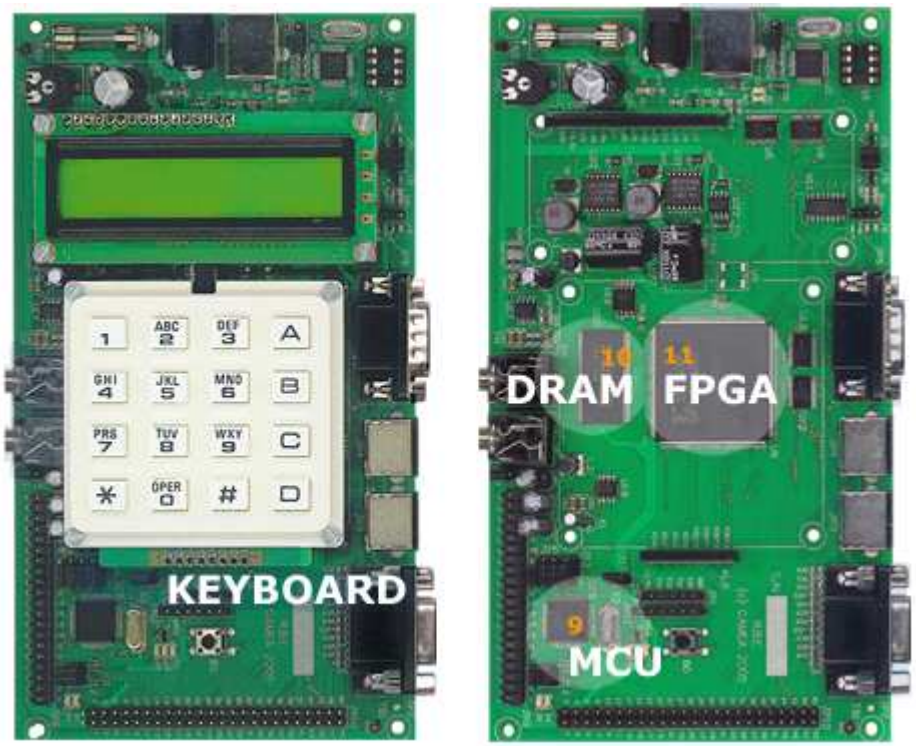
2 FITkit

Platforma FITkit umožňuje dosáhnout značných praktických dovedností. Lze na něm aplikovat a vyzkoušet si teoreticky nabyté znalosti z probrané látky. FITkit obsahuje výkonný mikrokontrolér s nízkým příkonem a řadu periferií. Důležitým aspektem je též využití pokročilého reprogramovatelného hardwaru na bázi hradlových polí FPGA, jenž lze, podobně jako software na počítači, neomezeně modifikovat pro účely dle potřeby – uživatel tedy nemusí vytvářet nový hardware pro každou aplikaci znovu, stačí upravit program.

V současné době vývoj FITkitu na fakultě Informačních technologií pokročil. Tato práce je postavena na FITkitu verze 1.2. Ten byl po dobu mého studia v předmětech vyučován a probíhaly na něm prakticky zaměřená cvičení.

V této práci je využíváno FPGA FITkitu pro dosažení správné funkce klávesnice a LCD displeje. FPGA je programováno jazykem VHDL. Ve velké míře je využito knihovních funkcí, které jsou na fakultě volně dostupné.

Pro hlavní obsluhu programu a generování PWM signálu se využívá mikrokontrolér MSP430F168IPM umístěný na FITkitu. Programování probíhá v jazyce C. Využívá se převážně časových obvodů. Pro připojení FITkitu k ovládacím vodičům serv jsou využity porty MCU na výstupní patici JP9.



obr 2.1 FITkit, zdroj [8]

Podrobné informace o FITkitu jsou dostupné na webových stránkách[8].

Následující kapitoly popisují využité komponenty FITkitu. Je zde zařazen i popis jazyka VHDL, který je nezbytný pro programování FPGA.

2.1 Popis kitu

FITkit v.1.2 obsahuje:

- **MCU MSP430F168IPM firmy Texas Instruments**,
- **FPGA Spartan 3 XC3S50-4PQ208C firmy Xilinx**,
- **USB-UART převodník FT2232C**,
- audio IN / OUT,
- konektory PS2, VGA, RS232,
- DRAM 8x8Mbit,
- **klávesnice,**
- **řádkový LCD displej.**

Tučně zvýrazněné komponenty jsou přímo použity v projektu. MCU je použito pro generování PWM signálu. Na LCD zobrazujeme informace o pohybu a jemnost kroku PWM. Ovládání pohybu manipulátoru provádíme pomocí klávesnice. Pro komunikaci FITkitu s okolím se využívá USB rozhraní.

2.1.1 Klávesnice

obr 2.2 Klávesnice 4x4

Na FITkitu se nachází klávesnice 4x4, obsahující 16 tlačítek zapojených do maticového uspořádání. Výhoda zapojení do matice je především úspora portů. Pro 16 tlačítek je využito pouze 8 portů. Popis funkcí jednotlivých tlačítek je popsán v softwarové části. Maticové uspořádání znázorňuje obrázek 3.5 v softwarové části (kapitola 3).

2.1.2 LCD

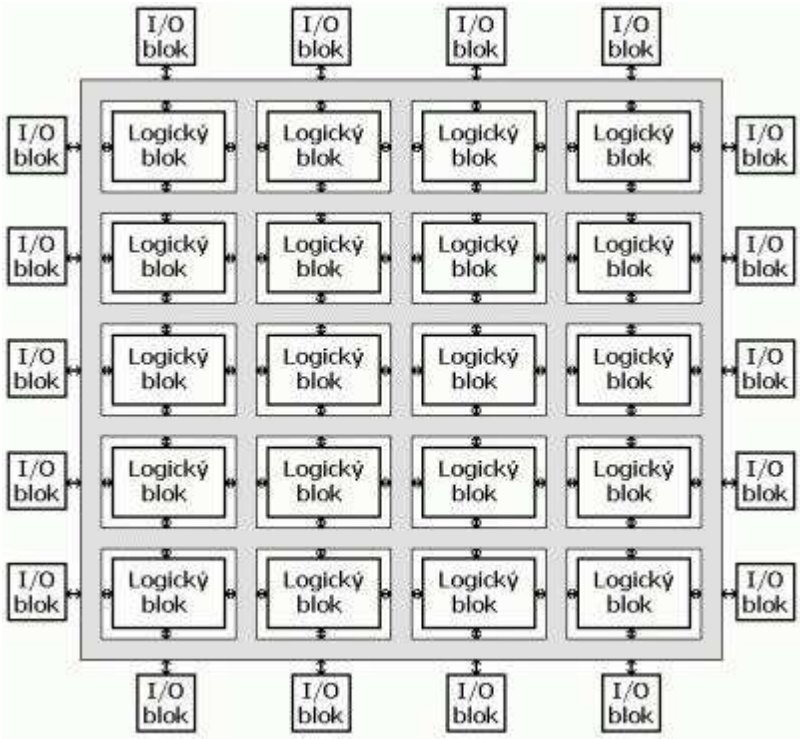
Na FITkitu je umístěn 16-ti znakový jednořádkový LCD(**L**iquid **C**rystal **D**isplay) displej. Displej je připojen přímo na piny FPGA. Modul displeje se skládá z řadiče a vlastního LCD prvku. Řadič obsahuje tři paměti - paměť typu ROM, která obsahuje bodové předlohy ASCII znaků, malé paměti RAM sloužící k definování vlastních znaků a RAM paměti uchovávající aktuální stav.

Model LCD, který se nachází na FITkitu nese označení CM1610NR-j2 a jde o výrobek firmy *DATA IMAGE Corporation*. Displej má v sobě zabudovaný řadič, který je plně kompatibilní s řadičem firmy *Hitachi*, který se používá ve většině LCD displejů.

Displej má v paměti ROM předefinovaných 248 znaků. Umístění znaků v paměti ROM je shodné s umístěním znaků v ASCII tabulce. Uživatel má i možnost definovat si svých 8 znaků a uložit je do paměti RAM. Lze tak docílit např. české diakritiky u některých znaků. Kontrast displeje lze řídit pomocí potenciometru umístěného v levém horním rohu FITkitu.

2.2 FPGA

Programovatelná hradlová pole (**F**ield-**P**rogrammable **G**ate **A**rray**)** jsou známa od roku 1985, kdy je přivedla na trh firma Xilinx. FPGA jsou nejsložitějšími, zároveň však také nejobecnějšími PLD obvody (**P**rogrammable **L**ogic **D**evice). Pomocí těchto obvodů můžeme zrealizovat a ozkoušet navržený hardware. V případě nalezení chyby stačí modifikovat program. To šetří čas a, hlavně firmám, peníze - nemusí se znovu leptat a osazovat deska plošného spoje, stačí pouze přepsat zdrojový kód pro FPGA. Na obrázku 2.3 je znázorněna typická struktura obvodu FPGA



obr 2.3 Struktura bloku FPGA, zdroj[9]

Logické bloky (Logic Cell) představují vlastní programovatelné logické bloky navzájem propojitelné globální propojovací maticí. Některé signály sousedících bloků je navíc často možné propojit přímo a bez použití globální propojovací matice. Takové spoje mají mnohem menší zpoždění

a umožňují realizovat například rychlé obvody šíření přenosu, což je nezbytné například pro čítačky a násobičky. Dále FPGA obvody obsahují samozřejmě vstupně-výstupní bloky a často i řadu dalších speciálních bloků. Například to jsou hardwarové násobičky, dualportové paměti RAM, PLL a DLL obvody a další.

Obvody FPGA obsahují pro svou konfiguraci pouze integrovanou paměť RAM. Je tedy nutné je po každém zapnutí znovu nakonfigurovat. To si však obvody dokážou provést i samy z externí EEPROM paměti, kterou vždy po zapnutí automaticky přečtou.

2.2.1 FPGA XC3S50-4PQ208C

Jedná se o FPGA, které je umístěno na FITkitu s následujícími parametry:

- 124 uživatelských vstupů/výstupů (I/O) s podporou až 23 různých I/O standardů,
- 192 konfigurovatelných logických bloků v matici 16×12, 1728 logických buněk, 50 000 logických hradel,
- paměť RAM 12 kb distribuovaných,
- dvě jednotky pro správu hodin (DCMs),
- čtyři násobičky18×18 bitů.

2.3 VHDL

Zkratka VHDL znamená **V**HSIC(Very-High-Speed Integrated Circuit) **H**ardware **D**escription **L**anguage. Je to typovaný programovací jazyk sloužící pro popis hardware. Používá se pro návrh a simulaci digitálních integrovaných obvodů například CPLD, FPGA nebo zákaznických obvodů (ASIC).

Jazyk byl původně vyvinut pro ministerstvo obrany USA, kde měl nahradit obsáhlé a nepřehledné manuály k elektronickým zařízením. Od roku 1987 je standardem IEEE. Jazyk je použitelný i pro návrh analogových obvodů, má prostředky pro popis paralelismu, konektivity a explicitní vyjádření času.

V současné době podporují jazyk hlavně firmy zabývající se vývojem a výrobou FPGA čipů. Podpora se týká hlavně vývojářského softwaru. Firmy jako *Xilinx* nebo *Blue Pacific* nabízejí vývojářský software zdarma, čímž dosahují značné obliby mezi studenty. V roce 2008 byla schválena nová verze jazyka VHDL 4.0, která je známa též pod názvem VHDL 2008. I tento krok naznačuje, že VHDL je stále populárnějším jazykem a má před sebou jistě velkou budoucnost.

2.3.1 Popis zařízení ve VHDL

VHDL popisuje číslicová zařízení a jednotlivé části zařízení pomocí komponent, které se popisují entitou a architekturou. Entita definuje rozhraní komponent (jednotlivé porty a jejich směr-IN, OUT,

INOUT). Architektura je svázána s entitou a popisuje strukturu nebo chování dané komponenty. Architektura má 3 možnosti popisu:

- strukturální - složení komponenty,
- behaviorální - chování komponenty,
- dataflow - datový tok signálů.

2.3.2 Příklad obvodu typu D ve VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
entity dffx is
port (
   CLK   : in  std_logic;
   RST   : in  std_logic;
   DIN   : in  std_logic;
   DOUT  : out std_logic );
end dffx;
architecture behav of dffx is
begin
   process (CLK,RST)
   begin
      if (RST = '1') then
         DOUT <= '0';
      elsif (CLK'event and CLK = '1') then
         DOUT <= DIN;
      end if;
   end process;
end  behav;
```

Entita definující 3 vstupy a jeden výstup zařízení

Architektura obvodu

tzv. citlivostní seznam, dojede-li ke změně signálu CLK nebo RST je spuštěn proces

vlastní tělo chování obvodu typu D-asynchronního

obr. 2.4 Příklad obvodu D ve VHDL

2.4 MCU MSP430F168IPM

Mikrokontrolér je jednočipový počítač (**M**icro**C**ontroller **U**nit), většinou monolitický integrovaný obvod, obsahující kompletní mikropočítač. Jednočipové počítače se vyznačují velkou spolehlivostí a kompaktností, proto jsou určeny především pro jednoúčelové aplikace jako je řízení, regulace a pod.

Jednočipové počítače často tvoří hlavní a nedílnou součást vestavěných systémů. Jednočipový počítač v sobě zahrnuje jádro mikroprocesoru společně s nonvolatilní pamětí (ROM, FLASH, EEPROM), pamětí RAM a periferními obvody (logické vstupy/výstupy, komunikační linky, A/D převodníky, čítače, časovače, …). Můžeme tedy obsáhnout celou aplikaci, aniž by potřeboval složité podpůrné obvody.

V současné době je k dispozici bohatý sortiment mikrokontrolérů. Je možné si vybrat vhodný typ mikrokontrolérů téměř na míru podle typu a složitosti řešené aplikace. CPU jádro bývá u

jednotlivých členů rodiny stejné, liší se pouze výbavou periferních obvodů a velikostí programové paměti.

Důležitou charakteristikou je šíře datové sběrnice, podle které členíme mikrokontroléry na osmi-, šestnácti- a dvaatřiceti- bitové. Nejčastěji se používají osmibitové mikropočítače. K nejznámějším výrobcům patří Atmel, Motorola, Microchip, NEC, Siemens, Toshiba a Intel.

2.4.1 MCU MSP430F168IPM

MCU na FITkitu je 16-bitový nízkopříkonový mikrokontrolér rodiny MSP430 firmy Texas Instruments s následujícími parametry:

- nízké napájecí napětí (1.8V až 3.6V),
- nízký příkon:
  - 330μA v aktivním režimu při 1MHz a 2.2V,
  - 1,1μA v režimu stand-by,
  - 0,2μA v režimu vypnuto,
- 16-bitová RISC architektura, instrukční cyklus 125ns,
- paměť FLASH 48kB,
- paměť RAM 2kB,
- moduly na čipu:
  - 3-kanálové DMA,
  - 16-bitové časovače Timer_A, Timer_B,
  - komparátor,
  - sériové komunikační rozhraní USART0 (asynchronní UART, synchronní SPI, I2C), USART1 (asynchronní UART, synchronní SPI),
  - 12-bitové A/D a D/A převodníky.

2.4.2 Napájení FITkitu

Většina komponent umístěných na FITkitu pracuje s napětím +3,3V. Pro ostatní případy jsou využita pomocná napětí +1,2V a +2,5V.

Ve většině případů (zvláště při spojení FITkitu s notebookem) je potřeba FITkit připojit k externímu adaptéru +5V. Napájení přes USB port, při větším proudovém odběru při připojení, totiž často nepostačuje ke korektnímu startu. Operační systém pak nerozezná připojované zařízení.

3 Fáze vlastního návrhu

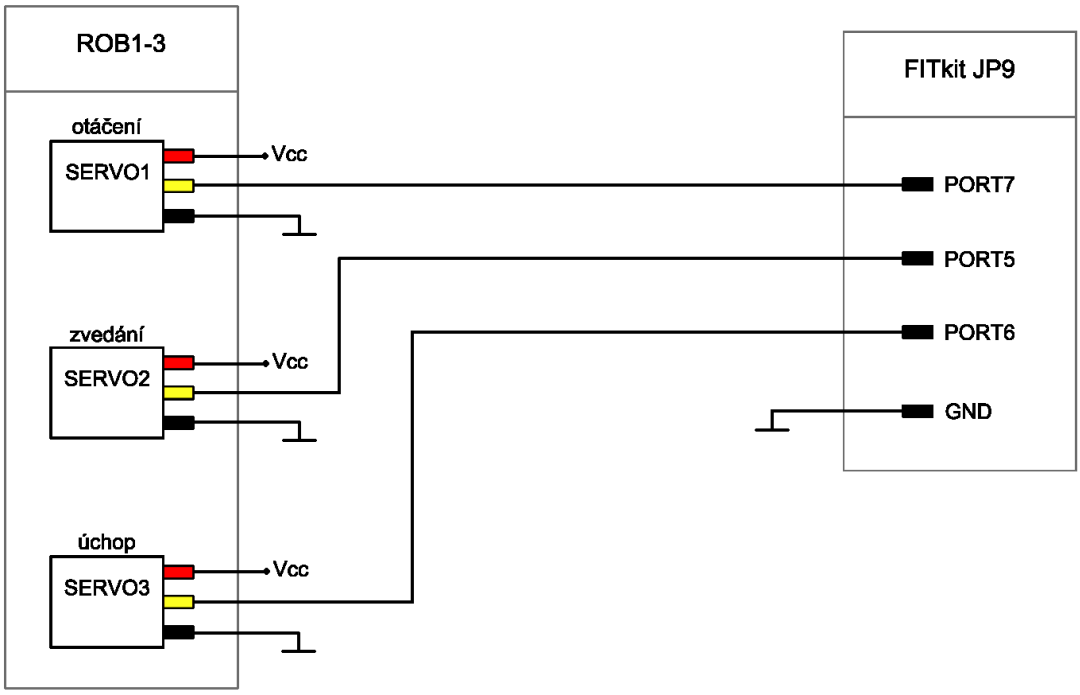
Touto kapitolou se dostáváme jak k propojení FITkitu a robotického manipulátoru ROB1-3, tak k vlastnímu softwarovému řešení této práce. Na začátku bude znázorněno připojení manipulátoru k FITkitu. V další části bude popsán postup při generování PWM signálu pomocí CPU MSP430F168IPM (dále jen CPU). Budou zde nastíněny jednotlivé kroky od matematických výpočtů po vlastní generování.

Dále zde budou popsány možnosti ovládání pohybu pomocí klávesnice, terminálu a následné zobrazování doplňujících informací na displej FITkitu, které bylo zmíněno v kapitole o VHDL.

Závěrem kapitoly zmíníme a zhodnotíme dosažené výsledky ovládání.

3.1 Připojení manipulátoru k FITkitu

Prvotní fází je návrh připojení manipulátoru ROB1-3 k FITkitu. Na obrázku 3.1 je mnou navržené zapojení. Zapojení je jednoduché, není nikterak omezeno.



obr 3.1 Propojení manipulátoru ROB1-3 a FITkitu

Ze strany manipulátoru ROB1-3 jsou připojeny tři ovládací serva. Každé servo má 3 výstupní vodiče. **Červený vodič** je připojen na napájecí napětí *Vcc*. Pro testování bylo zprvu použito napětí +5V, později se pro získání většího točivého momentu přešlo na napětí +6V. Na **žlutý vodič** se přivádí

generovaná střída. **Černý vodič** propojuje země mezi manipulátorem, FITkitem a napájecím zdrojem. Pokud nebude dodrženo propojení zemí mezi všemi zařízeními, nebude systém fungovat.

Ze strany FITkitu jsou vodiče připojeny k patici *JP9*. Konkrétně se jedná o vstup/výstupy MCU. Port 7 na patici JP9 odpovídá portu *P4M0* MCU, port 5 pak *P4M2* a nakonec port 6 *P4M3*.

3.2 Využití FITkitu pro ovládání

Pro ovládání manipulátoru jsem se rozhodl využít FITkit. Byla tu sice možnost pro ovládání využít pákový ovladač (joystick), ale nabízený byl neproporcionální. Tudíž jsem se rozhodl využít komponenty samotného FITkitu. Pro ovládání pohybu manipulátoru je využita klávesnice. LCD FITkitu pak zobrazuje reakci na stisk kláves a funguje převážně jako zpětná vazba, pro kontrolu stisků kláves, mezi uživatelem a FITkitem.

Manipulátor lze rovněž ovládat přes terminál. Tento způsob se ukázal jako rychlejší a hlavně přesnější. Rovněž lze vytvářet textové soubory se souřadnicemi, po kterých se manipulátor bude pohybovat. Ovládání pomocí terminálu bude popsáno v kapitole 3.4.6 .

3.3 Využití FPGA

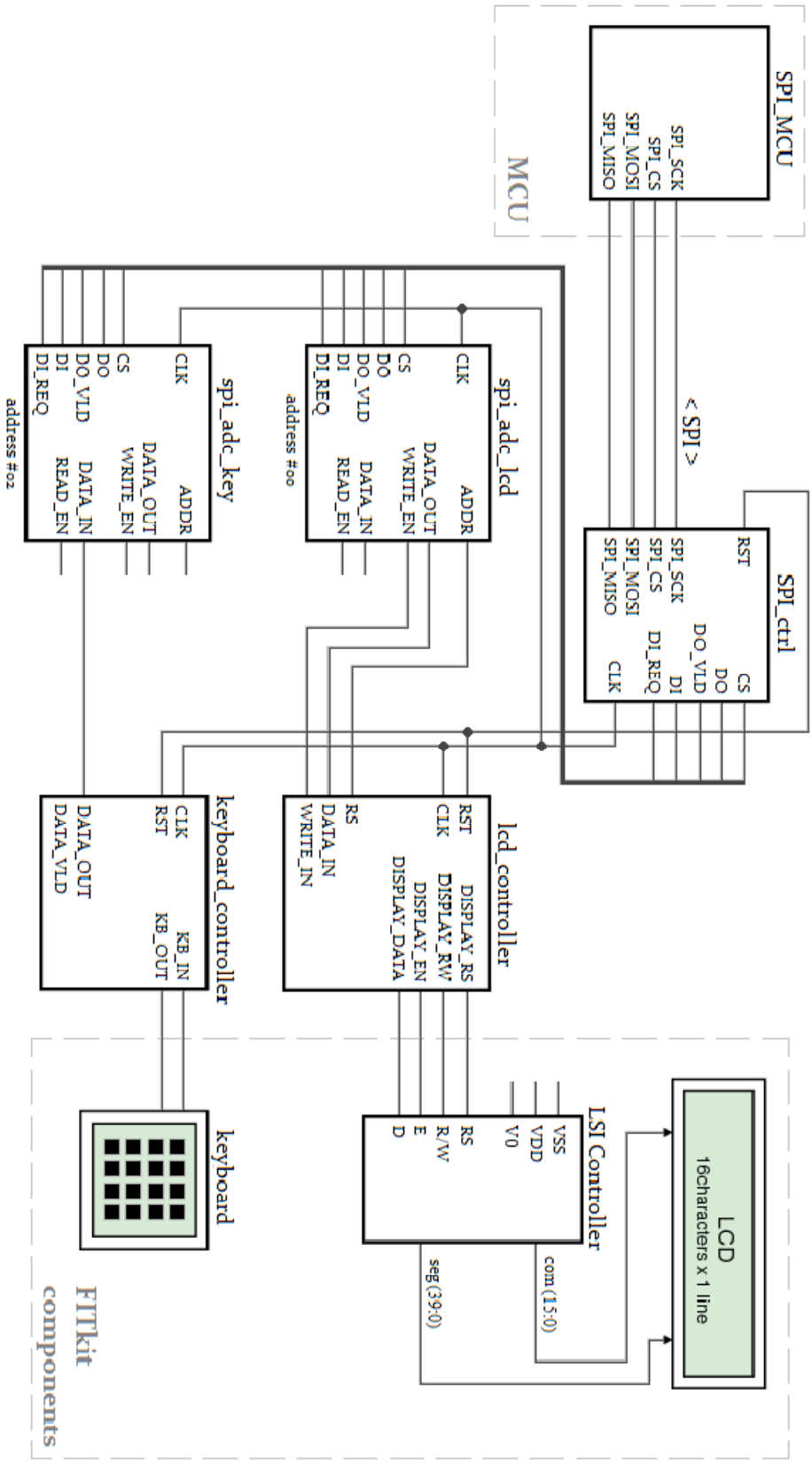
Jak již bylo řečeno, FPGA je využito pro funkčnost LCD displeje, na kterém je vhodné zobrazovat uživateli aktuální stav a dění na FITkitu. Pro ovládání celého manipulátoru se pak využívá klávesnice. Na obrázku 3.2 je znázorněno propojení komponent v FPGA, od kterého se odvíjí následující text.

3.3.1 Struktura zapojení

Hlavní program pro ovládání manipulátoru je uložen v MCU FITkitu. Klávesnice a LCD je ovládáno pomocí FPGA. Je tedy zapotřebí komunikace mezi mikrokontrolérem a FPGA čipem. K tomu účelu se využívá SPI (**S**eriál **P**eripheral **I**nterface), což je sdílená sběrnice, na kterou je mimo jiné připojená i FLASH paměť. Komunikace SPI rozeznává dva typy zařízení. Řídícím (Master) je v tomto případě mikrokontrolér, ke kterému se připojují ostatní periferie (Slaves).

3.3.2 SPI komunikace

SPI komunikace probíhá čtyřmi signály. Prvním signálem je `SPI_SCK`(clock), který generuje master. Jde o hodinový signál, který v závislosti na zvoleném režimu určuje okamžik, kdy dochází k vzorkování dat. Signál `SPI_MOSI`(**M**aster **O**ut **S**lave **I**n) rovněž generuje master a jedná se o data vystupující z master zařízení. Data vstupující do master zařízení jsou přenášeny signálem `SPI_MISO`(**M**aster **I**n **S**lave **O**ut), generuje ho slave. Posledním signálem je `SPI_CS`(**C**hip **S**elect), který generuje opět master. Jde o signál vymezující datový rámec aktivní v logické 0 (při přenosu).



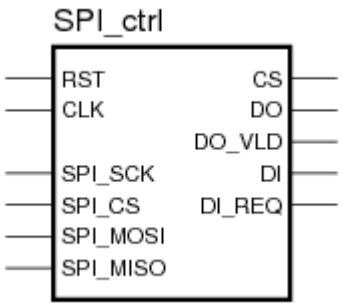
obr. 3.2 Zapojení komponent v FPGA

3.3.3 SPI řadič

Ze strany FPGA je komunikace zprostředkovávána SPI řadičem. SPI řadič (komponenta SPI_ctrl) je jednoduchá komponenta, která má za úkol realizovat převod SPI protokolu na interní sériovou sběrnici. Kromě převodu dekóduje, zda se transakce týká FPGA či nikoliv.

Signál `SPI_SCK` je vzorkován hodinovým signálem `CLK` a detekují se vzestupné a sestupné hrany. Při vzestupné hraně signálu `SPI_SCK` se provádí čtení bitu signálu `SPI_MOSI`, při sestupné hraně se mění hodnota signálu `SPI_MISO`. Veškeré signály vnitřní sběrnice jsou synchronizovány hodinovým kmitočtem `CLK`. Po zahájení komunikace (aktivací signálu `SPI_CS`) řadič postupně načte první čtyři zaslané bity. Pokud se shodují s identifikací přidělené FPGA (0001), je aktivován signál `SPI_CS` a je povoleno generování signálů `DO_VLD` a `DI_REQ`. Pokud se identifikace neshoduje, je signál `SPI_CS` neaktivní a řadič čeká na konec transakce indikované deaktivací `SPI_CS`.

3.3.3.1 Komponenta pro VHDL



obr 3.3 Komponenta pro SPI řadič, zdroj [8]

`RST`	reset
`CLK`	hodinový signál
`SPI_SCK, SPI_CS, SPI_MOSI, SPI_MISO`	signály rozhraní SPI
`CS`	Pokud signál v '1', probíhá komunikace se zařízením uvnitř FPGA
`DO`	seriová data, výstup z MCU
`DO_VLD`	v '1' určuje platnost dat DO
`DI`	seriová data, vstup do MCU
`DI_REQ`	v '1' je nutné v následujícím taktu CLK vystavit na vodiči DI další bit datového slova. Bit vystavený na DI musí zůstat platný až do příchodu dalšího požadavku

tabulka 3.1 Vstup/výstupy komponenty pro SPI řadič

3.3.4 SPI_dekodér

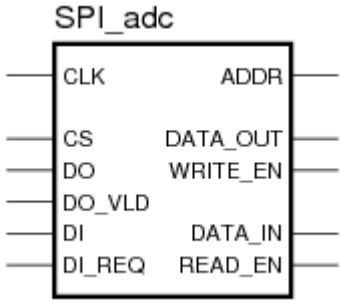
Po přijmutí dat z mikrokontroléru je zapotřebí převést přijatá data na vnitřní paralelní rozhraní, které umožní komunikaci s připojenými periferiemi. To zařizují komponenty spi_adc_lcd a spi_adc_key.

V případě, že dojde k aktivaci signálu SPI_CS, přijme nejprve dekodér 4bity, které odpovídají čtyřem nejméně významným bitům operačního znaku. V těch je zakódován typ operace - čtení, zápis, případně čtení i zápis. V dalším kroku dekodér přijme určitý parametricky zvolený počet adresních bitů. Po přijmutí adresy se provede komparace nejvyšších bitů (generické parametry). Pokud se bity shodují s bázovou adresou, znamená to, že jsou data určena pro zařízení připojené k tomuto dekodéru a pokračuje se přenosem dat.

Vždy, když je přijmut určitý počet bitů (daný parametrem DATA_WIDTH), jsou data vystavena na sběrnici DATA_OUT a je generován signál WRITE_EN. Pokud jsme zvolili režim čtení ze zařízení (nastaven bit RE), pak je vždy před zahájením přenosu generován signál READ_EN, který informuje, že je nutné v následujícím taktu CLK vystavit na sběrnici DATA_IN platná data, která budou přenesena do mikrokontroléru.

3.3.4.1 Komponenta pro VHDL

Na následujícím obrázku je znázorněna komponenta pro FPGA



obr. 3.4 Komponenta pro SPI_dekodér, zdroj [8]

CLK	hodinový signál
CS, DO, DO_VLD, DI, DI_REQ	interní signály FPGA
ADDR	adresa komponenty pro zápis/čtení dat
DATA_OUT	výstup do periferie připojené k dekodéru
WRITE_EN	určuje dobu platnosti dat na výstupu DATA_OUT
DATA_IN	výstup z periferie připojené k dekodéru
READ_EN	určuje dobu platnosti dat na vstupu DATA_IN

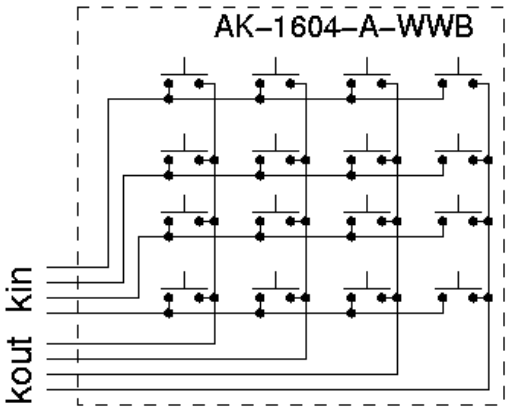
tabulka 3.2 Vstup/výstupy komponenty pro SPI dekodér

3.3.5 Řadič Klávesnice

Existují dva způsoby jak testovat stisk klávesy. Prvním z nich je skenování portů, kde se nestále testujeme stisk klávesy. Druhým způsobem je pak přerušení modulu klávesnice.

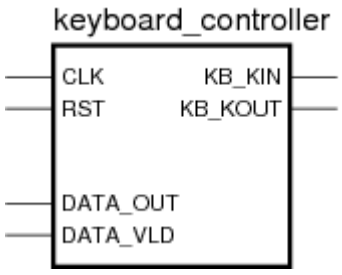
Každý způsob má své výhody a nevýhody. A protože počítám s tím, že manipulátor v této práci zastupuje aktivní člen, rozhodl jsem se využít neustálého skenování.

Program pro skenování stisku klávesnice je psaný v jazyku VHDL a uložen v FPGA FITkitu. Na vstupy klávesnice `KB_KIN` přikládáme signál, který postupně jednoznačně určí jeden řádek klávesnice a na výstupu klávesnice `KB_KOUT` zjišťujeme, která tlačítka jsou ve vybraném řádku zmáčknuta.



obr. 3.5 Maticové zapojení klávesnice

3.3.5.1 Komponenta pro VHDL



obr. 3.6 Komponenta pro obsluhu klávesnice, zdroj [8]

`CLK`	hodinový signál
`RST`	reset
`KB_KIN`	výstupní signál pro připojení ke vstupu klávesnice
`KB_KOUT`	vstupní signál pro připojení k výstupu klávesnice
`DATA_OUT`	stav kláves
`DATA_VLD`	signál určující platnost signálu DATA_OUT

tabulka 3.3 Vstup/výstupy komponenty pro řadič klávesnice

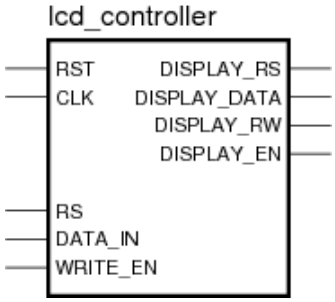
Signál DATA_VLD není v práci použit. Tento signál se používá pro obsluhu klávesnice pomocí přerušení.

3.3.6 LCD řadič

Displej je připojen přímo na piny FPGA. Modul displeje se skládá z řadiče a vlastního LCD prvku. Řadič obsahuje tři paměti - paměť typu ROM, která obsahuje bodové předlohy ASCII znaků, malé paměti RAM sloužící k definování vlastních znaků a RAM paměti uchovávající aktuální stav.

Komponenta lcd_controller je napojena na komponentu LSI_Controller, která je hardwarová. Zbývající signály LSI Controlleru jsou: VSS, na který se připojuje zem. VDD slouží k připojení napájecího napětí a signálem V0 můžeme určit jas displeje.

3.3.6.1 Komponenta pro VHDL



obr 3.7 Komponenta pro ovládání LCD displeje, zdroj [8]

CLK	hodinový signál
RST	Signál, který uvádí řadič do výchozího stavu
DISPLAY_RS, DISPLAY_DATA, DISPLAY_RW, DISPLAY_EN	Signály displeje
RS	Rozlišení mezi daty a řídícími signály
DATA_IN	Vstupní data, která budou zapsána do řadiče displeje
WRITE_EN	Zápis dat do řadiče LCD displeje

tabulka 3.4 Vstup/výstupy komponenty pro LCD řadič

3.3.7 Ovládání pomocí klávesnice

K ovládání dvou hlavních os manipulátoru je využita hvězdice kláves 2, 4, 6, 8. Úchop se ovládá hvězdičkou a křížkem. Pro ovládání pomocí klávesnice je umožněno přepínat mezi citlivostmi. Klávesou 1 zvolíme menší citlivost (docílíme větší rychlosti) a klávesou 3 zvolíme větší citlivost.

Generování pwm se startuje klávesou 5. Klávesou 0 se naopak generování vypíná.

Klávesa A slouží k povolení příjmu dat z terminálu. Povoluje se tak možnost řídit manipulátor pomocí terminálu. Klávesou B naopak FITkit začne vysílat data o změně pozice manipulátoru. Tím je teoreticky umožněno nahrávání pohybu a následná reprodukce z txt souboru. Klávesa C pak ruší aktivity spojené s terminálem a řízení odkazuje pouze na klávesnici FITkitu.

Tlačítko	Reakce manipulátoru	LCD zpráva
1	zvětší se citlivost (zmenší se rychlost) pohybu	sens_high
3	zmenší se citlivost (zvětší se rychlost) pohybu	sens_low
2	pohybu manipulátoru nahoru	mv_up
8	pohyb manipulátoru dolů	mv_down
4	pohyb manipulátoru doleva	mv_left
5	start generování pwm	ready
6	pohyb manipulátoru doprava	mv_right
*	úchop manipulátoru	push
#	uvolnění úchopu manipulátoru	drop
A	povolení příjmu dat z terminálu	listen
B	povolení vysílání dat o pozici do terminálu	send position
C	zakázání příjmu i vysílání dat	cancel

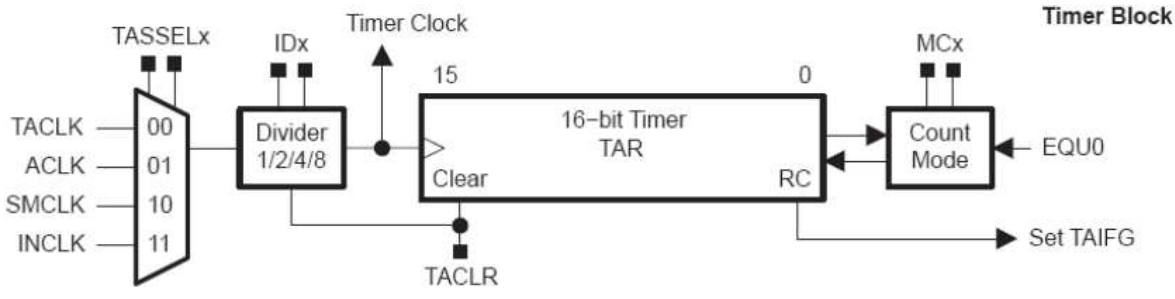
tabulka 3.5 Ovládání manipulátoru pomocí klávesnice FITkitu

3.4 Využití MCU

V MCU je uložen hlavní program. V této kapitole jsou popsány nezbytné součásti pro práci s programem jako časovače nebo přístup k externímu řízení manipulátoru pomocí terminálu.

3.4.1 Časovací systém

V MCU umístěném na FITkitu jsou dva časovače – Timer_A a Timer_B. Na následujícím obrázku je zobrazena struktura jednoho časovače, konkrétně časovače A:



obr. 3.8 Struktura časovače MCU, zdroj [11]

Jedná se o základní obrázek časovacího systému. Rozšířená verze obrázku obsahuje registry a spojení, které v této práci nebudeme využívat.

Registr `TASSELx` umožňuje nastavení vstupní frekvence časovače. Máme na výběr ze čtyř frekvencí. My použijeme ACLK o frekvenci 32,767 kHz, označeno proměnnou TASSEL_1 a SMCLK o frekvenci 7,3728 MHz ukryto v proměnné TASSEL_2. Registr `IDx` určuje dělící poměr vstupní frekvence. Můžeme tak dělit vstupní frekvenci a přímo si tak zvětšovat cyklus čítače.

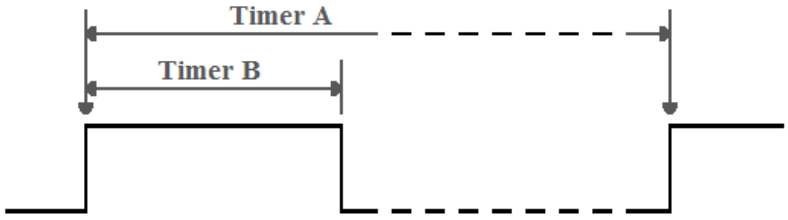
Pomocí registru MCx upravujeme typ čítaní časovače (count mode). MC_0 uvádí časovač do neaktivního stavu. MC_1 určuje inkrementační mód, kdy po dosažení nastavené hodnoty se čítač zastaví. MC_2 oznamuje nepřetržité čítání časovače. Časovač vždy po dosažení přednastavené hodnoty vyvolá přerušení, vynuluje se a čítá znovu. Posledním módem je MC_3. Je to mód čítající nejdříve k přednastavené hodnotě časově a po dosažení této hodnoty odčítá k nule.

Do registru `TACCR0` ukládáme hodnotu, po jejímž dosažení se vyvolá příznak přerušení daného časovače. `TACLR` je příznak, pomocí kterého můžeme resetovat čítač. Příznak je automaticky nastaven (např. při dosažení přednastavené hodnoty count mode MC_2) a poté resetován.

Příznak přerušení TAIFG svou hodnotou říká, zda se žádá o přerušení (0=není požadavek na přerušení, 1=požadavek na přerušení).

3.4.2 Tvorba PWM

Tvorba PWM signálu je základ pro ovládání serv manipulátoru. Existuje vícero způsobů, jak signál generovat. Osobně jsem dal přednost generování PWM pomocí dvou časovačů.



obr 3.9 Použití časovačů při generování PWM

Pro generování signálu PWM jsou použity dva časovače. `Timer_A` určuje periodu signálu. `Timer_B` pak určuje střídu.

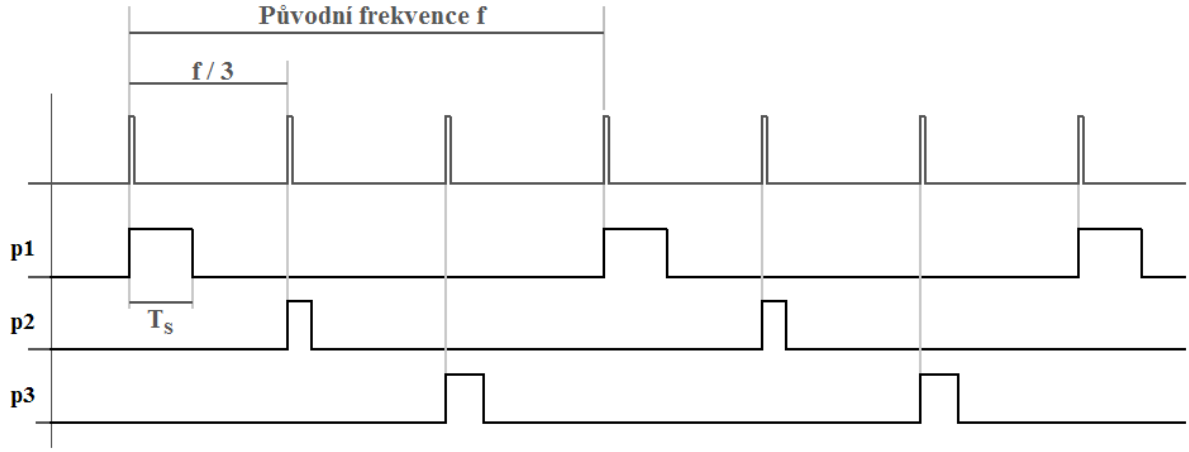
3.4.3 Funkce časovačů

Pro řízení tří serv manipulátoru je zapotřebí generovat tři PWM signály. Metodou ovládání jednoho výstupního portu pomocí dvou časovačů by nám tak zabralo 6 časovačů. V MCU máme k dispozici však pouze dva časovače. Tahle cesta nebude možná stejně tak jako generování PWM v hlavním

programu bez použití časovačů, kde by mohly nastávat nepřesnosti. Budeme-li chtít použít časovače, musíme zvolit jiný přístup k řešení problému.

Začneme tím, že dobu periody PWM signálu si můžeme do jisté míry upravit pro vlastní potřeby, aniž bychom narušili funkčnost serv, a to v rozmezí 15-25ms. Rovněž známe nejdelší možnou dobu trvání aktivní úrovně (logické '1') ve střídě PWM signálu, která je 2,5ms.

`Timer_A` nemusí startovat začátek periody pouze jednoho portu. Vezmeme-li v úvahu předchozí číselné údaje, zjistíme, že pokud třikrát zvýšíme frekvenci `Timeru_A`, můžeme startovat PWM signál na třech portech s dostatečnou rezervou. Z následujícího obrázku pochopíme situaci lépe:



obr 3.10 Průběh PWM na 3 portech

Původní frekvenci jsme ztrojnásobili. Dle vztahu $T = \frac{1}{f}$ jsme dostali třikrát nižší periodu. Dobu maximální střídy znázorňuje T_S. Je tedy třeba nastavit časovač `Timer_A` tak, aby v každém přerušení nastavil aktivní úroveň na odpovídající port a zároveň do counter registru timeru B nastavil požadovanou dobu trvání střídy na daném portu. `Timer_B` pak zajistí přepsání hodnoty na portu do log. 0. Pro lepší pochopení uvedu algoritmus pro obsluhu přerušení časovačů:

TIMER_A:

1. Zjisti, které servo má být dle pořadí obslouženo.
2. Nastav na příslušný port serva aktivní úroveň (logickou '1').
3. Nastav do counter registru pro TIMER_B dobu aktivní úrovně.
4. Spusť TIMER_B.

TIMER_B:

1. Zjisti, které servo má být dle pořadí obslouženo.
2. Nastav na příslušný port serva neaktivní úroveň (logickou '0').
3. Aktualizuj pořadí pro obsluhu serv (přejdi k řízení dalšího).
4. Zastav TIMER_B.

3.4.4 Nastavení časovačů

Pro dosažení správné funkčnosti je nezbytné správně nastavit časovače, které mají za úkol tvořit PWM signál.

Doba periody pro řízení jednoho serva musí být 15-25ms. Doba aktivního stavu pak musí odpovídat rozsahu 0,5-2,5ms. Pro `TIMER_A` zvolíme vstupní frekvenci `f=32,767 kHz`. Tato frekvence je dostatečná pro pokrytí rozsahu citlivostí.

Výpočet pro generování nosné periody signálu PWM:

1. Dobu periody zvolíme 20ms (střed mezi 15-25ms). Perioda jednoho portu $T_P = \frac{20ms}{3} = 6{,}667ms$.
2. Zvolíme frekvenci pro `TIMER_A` 32, 767kHz, perioda $T_č = \frac{1}{f} = 30{,}519\mu s$
3. Počet taktů pro časovač: $\frac{T_P}{T_č} = \frac{6{,}667ms}{30{,}519\mu s} = 218.355 = \mathbf{218}\ takt$ů $časovače$.

Dostaneme tak hodnotu, kterou musíme zapsat do counter registru časovače pro dosažení periody *T=6,667ms*. Tím docílíme toho, že po uplynutí požadované doby přejde program do přerušení. V přerušení střídáme v pravidelném pořadí řízení jednotlivých serv.

Pro časovač TIMER_B je zapotřebí změnit vstupní frekvenci. A to z důvodu přesnosti pohybu serva. Potřebujeme totiž dostatečnou rozlišovací schopnost k pokrytí celého pohybu v rozmezí 0-180°. Při frekvenci 32,767 kHz by nám vycházelo, že na změnu pohybu o 1°, bychom museli nastavit dobu aktivní úrovně (dobu činnosti pro TIMER_B) na $\frac{2ms}{180°} = \mathbf{10,111}\boldsymbol{\mu s}$. Nejmenší možná doba při frekvenci 32,767 kHz nám vyjde $\frac{1}{32{,}767Hz} = \mathbf{30{,}519}\ \boldsymbol{\mu s}$. Z tohoto jasně vyplývá nedostatečné pokrytí rozsahu. Proto jako vstupní frekvenci zvolíme TASSEL_2 o hodnotě 7,3728MHz. Spočítáme si rozsah mezních hodnot pro pohyb v rozmezí 0-180°.

1. Nejmenší hodnota pro polohu 0° = doba aktivní úrovně po dobu 0,5ms, která je potřeba nastavit do counter registru pro TIMER_B je: $\frac{0{,}5ms}{1\ /\ 7{,}3728MHz} = \mathbf{3686}\ takt$ů $časovače$.
2. Největší hodnota pro polohu 180° = doba neaktivní úrovně po dobu 2,5ms, která je potřeba nastavit do counter registru pro TIMER_B je:. $\frac{2{,}5ms}{1\ /\ 7{,}3728MHz} = \mathbf{18432}\ takt$ů $časovače$.

Z dosažených výsledků můžeme vyvodit i další aspekt. Nemusíme se zaobírat nastavováním děliček pro vstupní frekvence časovačů. V obou případech ani zdaleka nedosahujeme hodnoty velikosti registrů časovačů: 2^{16}= 65536 >> 18432, což je největší požadovaná hodnota. Proto není

důvod si vstupní frekvenci dělit. Navíc jak se ukáže, je ve výsledku lepší se v programu pohybovat ve větších číslech. Dává nám to totiž možnost měnit rychlost pohybu.

Chceme-li tedy změnit pohyb o x°, potřebujeme do counter registru připočíst k minimální hodnotě 3686 x-násobek čísla $82 = \frac{max-min}{180} = \frac{18432-3686}{180}$. V programové smyčce skenujeme stisk kláves v jistém intervalu. Kdybychom rovnou přičetli hodnotu 82, manipulátor by změnil sice polohu o jeden stupeň, ale ve výsledku by byl pohyb hodně rychlý určitě by postrádal přesnost. To je zapříčiněno rychlou programovou smyčkou. Tento problém je v programu napraven snížením přičítané hodnoty k aktuální poloze. Ve výsledku přičítáme nebo odčítáme k aktuální poloze hodnotu 17-35 v závislosti na požadované přesnosti, kterou si můžeme definovat. Výsledný pohyb je tedy plynulejší.

Přehled pro nastavení registru časovače tedy bude vypadat následovně:

`TIMER_A:`

`TACCTL0 = CCIE`	povolení přerušení pro časovač TIMER_A
`TACTL = TASSEL_1 + MC_2`	vstupní frekvence pro TIMER_A nastavena na 32,767kHz + opakovací mód
`TACCR0 = 218`	nastavení periody PWM signálu

`TIMER_B:`

`TBCCTL0 = CCIE`	povolení přerušení pro časovač TIMER_B
`TACTL = TASSEL_2 + MC_2`	vstupní frekvence pro TIMER_B nastavena na 7,3728MHz + po dosažení hodnoty counter regisru přestat čítat (`MC_0`)
`TBCCR0 = x`	doba aktivní úrovně

3.4.5 Mechanický limit

Teoreticky mají serva manipulátoru rozsah 0-180°. Pohyb manipulátoru je omezen jeho konstrukcí. V programu jsou tedy softwarově upravené meze pohybu v `ose y` a `stisku`. Softwarové zarážky chrání mechanickou konstrukci manipulátoru. Manipulátor se neničí před rychlými mechanickými dorazy a nemá tak destruktivní účinky sám na sebe – při nárazu na podstavnou plochu. V `ose x` je manipulátor schopen plného rozsahu, nicméně je zhruba o 1° pohyb omezen také.

Spodní část osy y je omezena do výšky 0,5cm na povrch stojné plochy. Vrchní maximum je omezeno asi na 5° od mechanického dorazu. Svírací část je rovněž omezena. Maximální sevření svíracích kleští je omezeno na malou mezírku mezi rameny. Roztažení kleští je taky omezeno, ale nelimituje největší možné roztažení.

Serva manipulátoru jsou poměrně silná a nebýt softwarových zarážek, manipulátor by mohl ohrožovat svou křehkou konstrukci.

V programu jsou softwarové zarážky definované v souboru *main.h* v proměnných `MAX_X, MIN_X, MAX_Y, MIN_Y, MAX_P, MIN_P`.

3.4.6 Ovládání přes terminál

Manipulátor lze řídit pomocí klávesnice, což je interaktivní a intuitivní. Na druhou stranu takovéto řízení postrádá přesnost a jistým způsobem i rychlost provedení mechanického úkonu. Proto lze manipulátor řídit i alternativní cestou a to přes terminál. Toto řízení má i další výhodu. Můžeme si vytvářet krátké prográmky v txt souborech a tyto programy pak libovolně spouštět.

V první řadě je potřeba na FITkitu povolit ovládání pomocí terminálu (viz tab. 3.5). Poté už můžeme zasílat řídící instrukce do FITkitu.

3.4.6.1 instrukce pro žízení

instrukce	popis			rozsah
`X [0..180]`	přesune rameno v x ose na souřadnici číslo			0 = max. vlevo
`Y [0..140]`	přesune rameno v y ose na souřadnici číslo			0 = min. výška
`P [0..15]`	intenzita stisku svíracích kleští			0 = nejvíc stisknuto
`S [1..5]`	rychlost pohybu manipulátoru			1 = nejrychlejší
`MOVE [l|o]`	určuje typ pohybu.			
	l	lineární	může vykonávat vícero pohybů současně	
	o	osový	provádí seznam pohybů postupně. Teprve po dokončení příkazu přejde na vykonávání dalšího příkazu	
`POS`	zašle aktuální informace o pozici všech tří os a vypíše je na terminál			

tab 3.6 Přehled ovládacích instrukcí terminálu

`MOVE` a `POS` jsou instrukce zasluhující delší popis. Instrukcí `MOVE` si můžeme zvolit typ pohybu, přičemž většina aplikací vyžaduje pohyb po osách postupně. Např. při zapalování svíčky knotu sirkou, potřebujeme dosáhnout určité výšky a až poté klesat se sirkou ke knotu. Při lineárním pohybu by mohlo dojít k tomu, že y souřadnice bychom dosáhli rychleji než x souřadnice. Zapálená sirka by pak mohla narazit na hranu svíčky a vypadnout. Na druhou stranu někdy můžeme vyžadovat šikmý pohyb. Oba pohyby lze v programu kombinovat.

Instrukce `POS` je pak užitečná při tvorbě programů. Klávesnicí FITkitu můžeme najet na souřadnice, které požadujeme. Poté zašleme po terminálu POS a FITkit nám vrátí pozice všech tří serv, které můžeme připsat do txt souboru. Tímto stylem můžeme vytvořit celý program.

Programy se do FITkitu odesílají jako ASCII soubory. FITkit si je postupně načte a automaticky přechází na zadané souřadnice. Existuje i možnost odposlechu souřadnic a následné

ukládání souřadnic do txt souboru. V podstatě tak můžeme nahrát pohyb s následnou reprodukcí. Nicméně nastává problém při zpětném přenosu txt souborů do FITkitu. Přijímací buffer FITkitu má omezenou velikost a to na 128 znaků, což nám velikost programu limituje. Toto omezení by šlo překonat využitím softwarového *hand-skake* nebo zprovoznění *RTS/CTS*. Nicméně z hlediska prezentace a demonstračních účelů si vystačíme s limitem 128 znaků.

V programu je postupný pohyb po jednotlivých osách (osový) řešen pomocí zámků. Hlavní strukturou pro parametry pohybu je struktura `move`, která má jako položky `type`, `speed` a `lock`. Do položky `move` definujeme typ pohybu, buď lineární nebo osový (`linear_move/line_move`). Položka `speed` slouží pro definici rychlosti pohybu v rozmezí 1(nejrychlejší) až 5. Poslední položkou je `lock`, která se využívá při osovém pohybu a určuje případné zámky na osách (hodnoty `lock_x, lock_y, lock_p, no_lock`). Pokud je požadavek na změnu polohy v osovém typu pohybu, do položky `lock` se uloží osa, která požaduje změnu. Dokud se aktuální pozice manipulátoru nerovná požadované pozici, zámek je stále platný. Teprve po dosažení požadované polohy se do `lock` nastaví hodnota `no_lock`. Pokud je některá z os zamčená, nejsou vykonávány další příkazy.

3.4.6.2 ukázka programu

Na následujících obrázcích je demonstrován krátký program pro úchop měšce.
Na prvních obrázcích je znázorněn pohyb příkaz po příkazu. Na posledních dvou obrázcích je děj trochu urychlen.

V prvních dvou příkazech je zaslána definice pohybu:

`move o` - budeme vyžadovat osový pohyb
`s 3` - rychlost požadujeme střední

následuje definice samotného pohybu po jednotlivých osách:

START

x 40

p 15

y 6

p 3

y 33

x 80

y 3 , p 16

y 30 , x 60 , p 0

3.5 Dosažené výsledky

Robotický manipulátor ROB 1-3 byl schopen zvládnout téměř veškeré pokusy. Byl schopen pracovat dostatečně rychle, aby zvládal přemístit objekty na jiné místo ve stejném ornamentu. Rovněž byl schopen uchopit sirku, tu si napálit o zapálenou svíčku a zapálit svíčku na druhé straně x osy (video s tímto pokusem je umístěno v příloze 3 na CD). To vše s dostatečnou citlivostí. Serva HS-311 byly velice výkonné. Manipulátor byl schopen zvedat i předměty s větší váhou. Nicméně bylo znát, že softwarově velice jemný pohyb se ukázal jako mechanicky hrubý (sekavý).

3.6 Návod k instalaci

K správné funkčnosti potřebujeme balík knihoven pro FITkit. Do složky `apps` nahrajeme složku s programem.

- Překlad aplikace pro MCU

  ```
  cd sw
  make
  ```

- Nahrát aplikaci do MCU

  ```
  make load
  ```

- Překlad konfigurace pro FPGA

  ```
  cd ../top
  make
  ```

- Nahrát aplikaci do FPGA

  `flash w fpga` v terminálu

  `File transfer -› top/output.bin` přes xmodem

4 Závěr

Cílem této bakalářské práce bylo využití FITkitu pro ovládání jednoduchého robotického manipulátoru. FITkit jsem se snažil využít v co největší míře, protože věřím, že kombinace FPGA a mikrokontrolérů se v budoucnu bude využívat stále častěji. Jsem rád, že naše fakulta má takové zařízení k dispozici a neustále pracuje na jeho vývoji.

K řešení práce jsem využil znalosti získané v předmětech Mikroprocesorové a vestavěné systémy (IMP) a Návrh počítačových systémů (INP).

Práci považuji za úspěšně dokončenou. Jedinou slabostí je absence softwarové HandShake komunikace přes terminál, což limituje velikost programů pro manipulátor.

Vylepšovat a rozšiřovat tuhle práci bych neviděl jako možný přínos. Pohyb je totiž do značné míry limitován a sebelepší ovládání neumožní manipulátoru vykonávat náročnější úkoly. Nicméně tato práce položila dobrý základ pro složitější úkoly na sofistikovanějších typech manipulátorů.

5 Literatura

[1] KLÍR, J., SEIDL, L. *Syntéza logických obvodů*. Praha, SNTL 1996.

[2] PĚCHOUNEK, M. *Toleranční analýza logických obvodů.* Sdělovací technika, 8. vydání, 1973, s.293-298.

[3] Wikipedia, the free encyclopedia, *Servo*, [online 13.2.2009], http://cs.wikipedia.org/wiki/Servomotor

[4] Wikipedia, the free encyclopedia, *Robot*, [online 11.2.2009], http://cs.wikipedia.org/wiki/Servomotor

[5] Wikipedia, the free encyclopedia, *PWM*, [online 26.3.2009], http://cs.wikipedia.org/wiki/Servomotor

[6] Materiály k přednáškám, *Mikroprocesorové a vestavěné systémy*, Brno 2008, https://www.fit.vutbr.cz/study/courses/IMP/private/VYUKA/PREDNASKY/STUDIJNI_OPORA/IMP_opora_pr.pdf

[7] Studijní opora k předmětu, *Mikroprocesorové a vestavěné systémy*, Brno 2008

[8] Fakulta informačních technologií, *FITkit* http://merlin.fit.vutbr.cz/FITkit/.cs

[9] PECH J. *Nebojte se FPGA*, [online 18.2.2009] http://hw.cz/Teorie-a-praxe/Dokumentace/ART365-Nebojte-se-FPGA.html

[10] ROTTA J. *Robot manipulátor 1-3*, http://www.hobbyrobot.cz/PDF/ROB+SOS.pdf

[11] WANG Y. *MSP430 Clock System and Timer*, Northeastern University 2007 http://www.ccs.neu.edu/home/noubir/Courses/CSU610/S07/MSP430-Clock-Timers.pdf

[12] Nagy, CH. *Embedded Systems Design using the TI MSP430 Series*, Boston:Newnes, 2003. ISBN 0-7506-7623-X

6 Seznam příloh

Příloha 1 Seznam součástek manipulátoru ROB1-3

Příloha 2 Nákres dílů manipulátoru ROB1-3

Příloha 3 CD s programem

Příloha 1

Seznam součástek manipulátoru ROB1-3

Distanční sloupek KDR07, 4ks

Distanční sloupek KDR12, 1ks

Distanční sloupek KDI6M3x20, 4ks

Distanční sloupek KDI6M3x50, 3ks

Izolační podložka IB2, 4ks

Záslepka Ø 8mm F715HP-08, 6ks

Upevňovací konzola Ø3mm MPJ 2621, 2ks

Šroub M1,6×8 s maticí, MPJ 0202

Šroub M3×16 se zápustnou hlavou, 6ks

Šroub M3×16s půlkulatou hlavou, 4ks

Šroub M3×12 s půlkulatou hlavou, 8ks

Šroub M3×8 s půlkulatou hlavou, 18ks

Matice M3, 14ks

Podložky Ø 3,2mm, 14ks

Příloha 2

Nákres dílů manipulátoru ROB1-3