Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Měřicí modul pro MATLAB

Bakalářská práce

květen 2013 Miroslav Dvořák, DiT.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Miroslav Dvořák, Dipl.tech.**
Osobní číslo:	**I10000**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Měřicí modul pro MATLAB**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem BP je vytvořit zařízení pro sběr dat a ovládání, které bude komunikovat přes rozhraní USB/Ethernet. Toto zařízení bude sloužit zejména jako měřící modul pro MATLAB, např. pro snímání průběhů vstupních signálů apod. Modul ale může být obecněji použitelný i pro jiné programovací jazyky. Zařízení bude realizováno na bázi mikroprocesoru a mělo by obsahovat 4 analogové a 4 digitální vstupy a 4 digitální výstupy. Součástí bakalářské práce bude také několik ukázkových laboratorních úloh s tímto zařízením.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. Doňar B., Zaplatílek K. MATLAB - začínáme se signály. BEN, 276 stran, 2006.
2. Matoušek D. C pro mikrokontroléry PIC. BEN, 368 stran, 2011.

Vedoucí bakalářské práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 1. 5. 2013 Miroslav Dvořák, DiT.

Poděkování

Rád bych poděkoval vedoucímu práce, Ing. Michaelu Bažantovi, PhD., za cenné rady a připomínky, které mi poskytl během vypracování mé bakalářské práce a za jeho ochotný přístup.

Anotace

Tato bakalářská práce se zabývá návrhem a realizací zařízení pro sběr dat a ovládání. Zařízení komunikuje s PC prostřednictvím rozhraní USB/Ethernet a je realizováno pomocí mikroprocesoru. Cílem práce je vytvořit jednoduché a cenově dostupné zařízení pro realizaci jednoduchých laboratorních úloh v prostředí MATLAB.

Klíčová slova

MATLAB, MICROCHIP, MOXA, USB, ETHERNET, PIC18F, RTOS

Annotation

This work describes the design and implementation of devices for data acquisition and control. The device communicates with PC through USB/Ethernet is realized by the microprocessor. The goal is to create a simple and affordably priced devices for the realization of simple laboratory tasks in MATLAB.

Keywords

MATLAB, MICROCHIP, MOXA, USB, ETHERNET, PIC18F, RTOS

OBSAH

Seznam zkratek

ADC	Analog to Digital Converter
CD	Collision Detection
CDC	Communications Device Class
CSMA	Carrier Sense Multiple Access
EAGLE	Easily Applicable Graphical Layout Editor
GUI	Graphical User Interface
HDL	Hardware Description Language
IDE	Integrated Development Environment
IP	Internet Protocol
LAN	Local Area Network
MATLAB	Matrix Laboratory
MPLAB	Microchip PIC Laboratory
NC	Not Connected
PC	Personal Computer
PIC	Programmable Intelligent Computer
PUM	PIC Universal Module
PUM-E	PIC Universal Module – Ethernet interface
PUM-U	PIC Universal Module – USB interface
RS-232	Radio Sector type 232
RTOS	Real-Time Operating System
SDK	Software Development Kit
TCP	Transmission Control Protocol
TTL	Transistor Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem této bakalářské práce je vytvořit zařízení pro sběr dat a ovládání, které bude komunikovat přes rozhraní USB/Ethernet. Toto zařízení bude sloužit především jako měřící modul pro MATLAB, např. pro snímání průběhů vstupních signálů. Jedná se především o úlohy typu monitorování napětí, měření teploty, snímání stavu vstupních kontaktů, či ovládání výstupních zařízení. Pomocí SW nadstavby tak lze realizovat pomaluběžný osciloskop, jednoduchý multimetr, zabezpečovací zařízení, vizualizaci či řízení menšího výrobního procesu nebo vzdálenou správu po síti LAN.

Zařízení bude realizováno na bázi mikroprocesoru a mělo by obsahovat 4 analogové vstupy, 4 digitální vstupy a 4 digitální výstupy. Součástí práce bude i několik ukázkových laboratorních s tímto zařízením. Důležitým požadavkem na zařízení je jednoduchost, nízká cena a možnost napájení zařízení přes rozhraní USB nebo 5 V adaptérem.

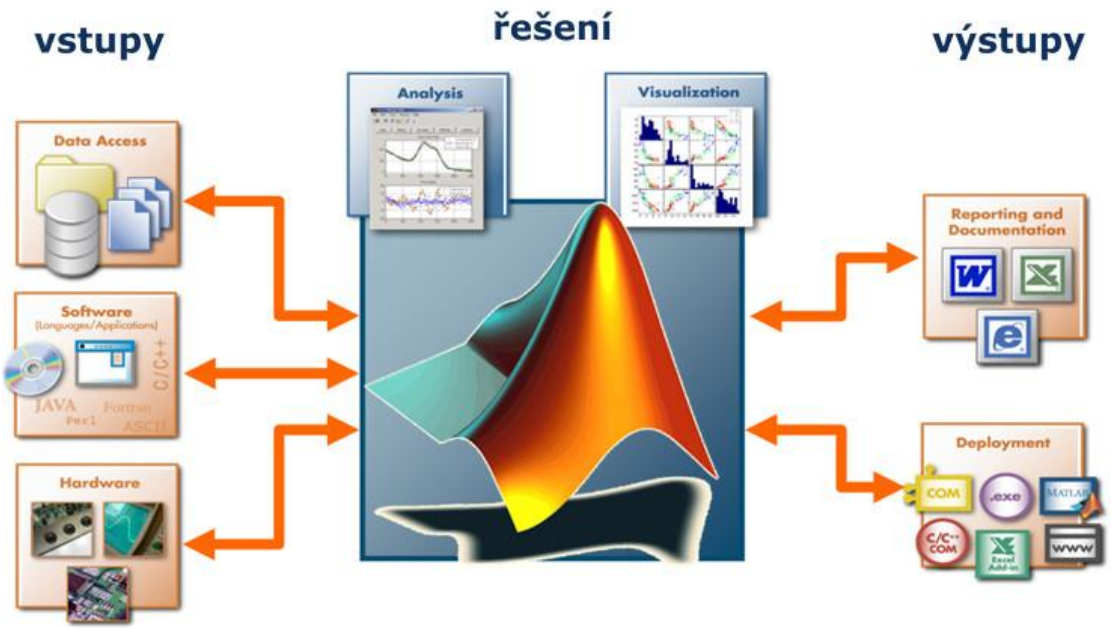
Měřicí karty pro MATLAB vyrábí např. firma Humusoft s.r.o. pro rozhraní PCI, které se cenově pohybují okolo 14 tisíc Kč. Pokud požadujeme rozhraní USB, máme možnost použít např. modul Labjack s cenou okolo 4 tisíc Kč. Pro naše jednoduché laboratorní úlohy však stačí jednoduchý měřicí modul, výše zmiňované moduly bychom z větší části ani nevyužili, nemluvě o jejich ceně. Z těchto důvodů jsem zvolil vlastní realizaci měřicího USB modulu. První volba byl mikroprocesor PIC 18F2455 od firmy Microchip vzhledem k integrovanému USB rozhraní a AD převodníku. U verze Ethernet je využito převodníku Ethernet/seriál od firmy Moxa z důvodu jednoduché implementace, bez nutnosti softwarové implementace protokolu TCP/IP. Tento převodník nám, díky své vlastní programovatelnosti, ještě více zjednoduší programovou část v mikroprocesoru.

2 Použitý software

V této kapitole bude uveden software, který jsem použil v bakalářské práci. Tento software lze rozdělit na tři skupiny, a to na open-source (NetBeans, MPLAB), na druhou skupinu časově omezenou verzi (MATLAB) a poslední třetí skupinu funkčně omezenou verzi (EAGLE).

2.1 MATLAB

MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. MATLAB je nástroj jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací.



Obrázek 1 - Struktura MATLABu. Zdroj [8].

Pro komunikaci s měřicím modulem PUM v obou verzích je využit Instrument Control Toolbox, který umožňuje komunikovat s přístroji, jako jsou osciloskopy, generátory funkčních průběhů a dalšími analytickými přístroji přímo z MATLAB. Komunikace probíhá na základě ovladačů zařízení a běžně používaných komunikačních protokolů, jako jsou GPIB, TCP/IP a UDP. S Instrument Control Toolboxem je možné generovat data v MATLABu a posílat je do přístrojů nebo načítat data z přístrojů do MATLABu pro analýzu a vizualizaci.

2.2 MPLAB IDE

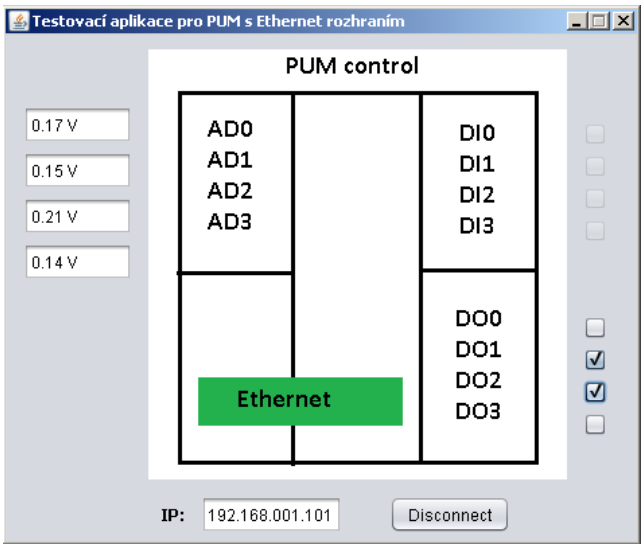
MPLAB je integrované vývojové prostředí (IDE) od firmy Microchip, které je volně ke stažení. Obsahuje integrovanou sadu nástrojů pro vývoj embedded aplikací s procesory PIC a dsPIC. MPLAB běží jako 32-bitová aplikace na platformě MS Windows, je snadno ovladatelný a obsahuje řadu volných softwarových komponentů pro rychlý vývoj aplikací a ladění. Více na www.microchip.com. V současné době Microchip nabízí multiplatformní vývojové prostředí MPLABX, které je založeno na Javě, jmenovitě na IDE NetBeans a umožňuje tak vyvíjet programy i pod operačním systémem Linux. Součástí obou vývojových prostředí je i volně stažitelný kompilátor jazyka C pro mikroprocesory řady PIC18 a balíček Microchip Application Libraries obsahující velké množství vytvořených knihoven a ukázek zdrojových kódů.

Obrázek 2 - Logo prostředí MPLABX. Zdroj [4].

2.3 NetBeans IDE

Vývojové prostředí NetBeans IDE od firmy Oracle je nástroj, pomocí kterého lze psát, překládat, ladit a distribuovat aplikace. Samotné vývojové prostředí je vytvořeno v jazyce Java, podporuje však i ostatní programovací jazyky, např. jazyk C. Existuje pro něj velké množství modulů, které toto vývojové prostředí neustále rozšiřují. Vývojové prostředí NetBeans je bezplatně šířený produkt. Více na www.netbeans.org.

Toto vývojové prostředí je použito pro tvorbu pomocných a testovacích programů používaných při vývoji měřícího modulu PUM. Součástí práce jsou i dvě ukázkové aplikace, které umožňují jednoduchým způsobem zobrazit stavy analogových a digitálních vstupů a současně umožňují i měnit stavy na digitálních výstupech. Obě aplikace využívají nekonečnou smyčku v samostatném vlákně pro komunikaci s modulem PUM. Tyto aplikace ukazují další možné využití měřicího modulu PUM.



Obrázek 3 - Ukázka testovacího programu v Javě

2.4 EAGLE

Editor plošných spojů EAGLE je uživatelsky přívětivý a výkonný nástroj pro návrh desek plošných spojů (DPS, PCB). Název EAGLE je zkratka pocházející z původního názvu Easily Applicable Graphical Layout Editor. Program se skládá ze tří hlavních modulů, které jsou ovládány z jednoho uživatelského prostředí. Proto není třeba konvertovat netlisty mezi schématy a deskami plošných spojů. EAGLE je dostupný pro operační systémy Windows a Linux, více na www.eagle.cz.

Moduly programu:

- Editor spojů
- Editor schémat
- Autorouter

Pro návrh je použita volně dostupná verze Light, která je omezena na plochu desky 100 x 80 mm a podporuje pouze dvě vrstvy spojů a jeden list schématu. Každý návrh se skládá ze dvou souborů, a to soubor s příponou SCH obsahující schéma a soubor s příponou BRD obsahující vlastní návrh desky.

3 Měřicí modul verze USB

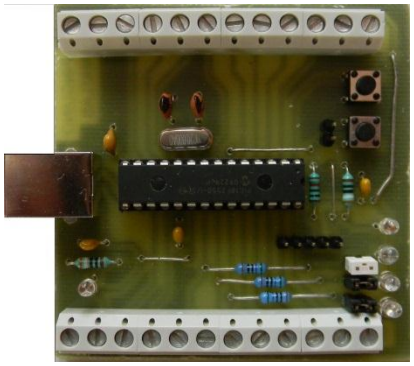
Měřicí modul PUM-U je napájen prostřednictvím rozhraní USB a obsahuje:

- 4 analogové vstupy s rozsahem 0–5 V
- 4 digitální vstupy TTL kompatibilní
- 4 digitální výstupy TTL kompatibilní
- sériovou linku pro komunikaci se servomotorem AX12

Schéma a návrh DPS včetně osazení je uveden v příloze C.

3.1 Popis hardware

Měřicí modul PUM-U je navržen vzhledem k jednoduchosti na jednovrstvé desce plošného spoje. Použitý mikroprocesor je osazen do precizní patice z důvodu snadné výměny v případě poškození, neboť vstupy a výstupy nejsou ošetřeny z důvodu snížení výsledné ceny modulu. Pro snadné programování mikroprocesoru PIC 18F2455 je modul osazen programovacími piny.



Obrázek 4 - Fotografie zkušebního modulu PUM-U

povel		data[0]	data[1]	data[2]	data[3]	data[4]
test připojení PUM		@	T	N	N	#
hodnota napětí (ADC) na vstupu		@	A	vstup	N	#
nastav pin (DIO) na hodnotu		@	E	pin	hodnota	#
hodnota pinu (DIO)		@	D	pin	N	#
posli znak na UART		@	U	znak	N	#
přečti znak z UARTU		@	V	N	N	#
pošli znak na UART-AX12		@	X	znak	N	#

Tabulka 1 - Povely modulu PUM

Každý paket začíná hlavičkou „@“, následuje typ povelu s parametry a konec paketu je zakončen znakem „#“. Délka paketu je pevná 5 bajtů.

svorka	pin PIC	funkce	popis funkce
1		NC	
2		NC	
3	RA0	AD0	analogový vstup
4	RA1	AD1	analogový vstup
5	RA2	AD2	analogový vstup
6	RA3	AD3	analogový vstup
7	RA5	NC	
8	RC0	RX1	SW sériová linka
9	RC1	TX1	SW sériová linka
10	RC2	NC	
11		+5V	
12		GND	
13	PB7	DI3	digitální vstup
14	PB6	DI2	digitální vstup
15	PB5	DI1	digitální vstup
16	PB4	DI0	digitální vstup
17	PB3	DO3	digitální výstup
18	PB2	DO2	digitální výstup
19	PB1	DO1	digitální výstup
20	PB0	DO0	digitální výstup
21	RX0	RX0	sériová linka/Moxa
22	TX0	TX0	sériová linka/Moxa
23		+5V	
24		GND	
	RE3	MCLR	tlačítko TL1
	RA4	LED	tlačítko TL2

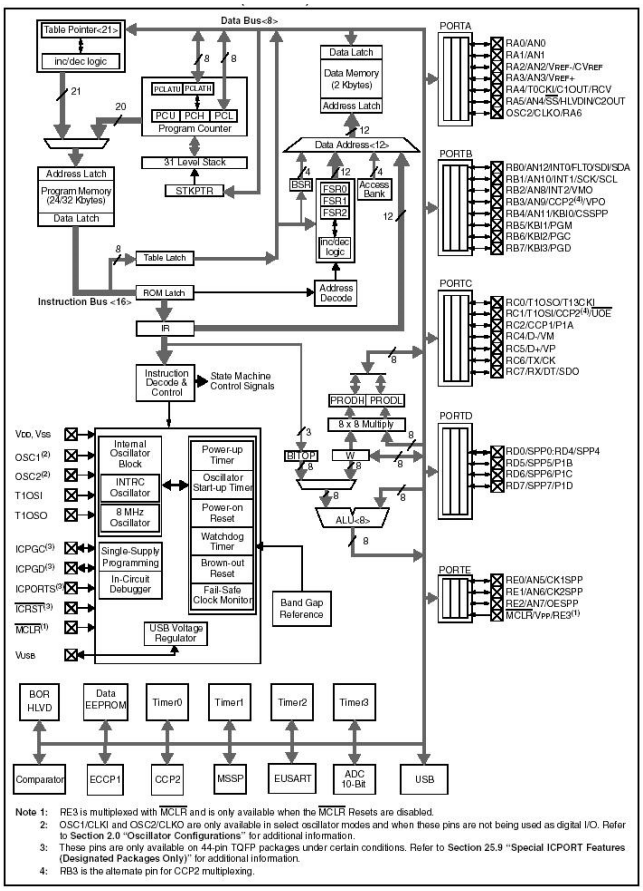
Tabulka 2 - Popis svorkovnice modulu PUM

3.1.1 Popis mikroprocesoru PIC 18F2455

Základní vlastnosti mikroprocesoru:

- 8-bitový mikroprocesor s architekturou RISC, 75 standardních instrukcí
- harvardská architektura, tj. paměť programu oddělená od datové paměti
- datová paměť 2 kB SRAM
- programová paměť typu FLASH 32 kB, 100 000 zápisů

- paměť 256 B EEPROM, 1 000 000 zápisů
- vnitřní fázový závěs 96 MHz
- hardwarová jednocyklová násobička 8 × 8 bitů
- 5 obousměrných 8-mi bitových portů
- 4 časovače /čítače, 2 jednotky CCP
- EUSART
- 8 vstupý 10-bitový A/D převodník
- 2 analogové komparátory
- 3 režimy řízení výkonu (Run, Idle, Sleep)
- rozhraní pro programování a ladění (ICSP)
- variabilita zdrojů hodin (vnitřní RC, krystal, vnější zdroj)
- HW implementace USB 2.0
- napájecí napětí 4,2–5,5 V



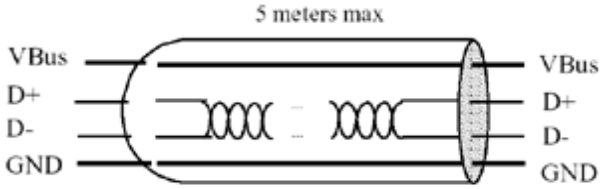
Obrázek 5 - Blokové schéma mikroprocesoru PIC 18F2455. Zdroj[4].

3.1.2 Popis rozhraní USB

Rozhraní USB (Universal Serial Bus) je běžnou součástí každého počítače a z větší části zcela nahradilo rozhraní RS-232. Rozhraní používá dva typy konektorů. Plochý konektor „typ A“, který je součástí každého PC a „typ B“ je určen pro periferní zařízení např. tiskárny.

Základní parametry rozhraní USB :

- komunikační rychlost od 1,5 Mbit/s do 480 Mbit/s
- komunikační vzdálenost do 5 m
- možnost připojení více zařízení
- rozhraní obsahuje 5 V napájení



Obrázek 6 - Struktura USB kabelu. Zdroj [5].

Číslo pinu	Označení pinu	Popis pinu
1	VBus	+5 VDC
2	D-	Data -
3	D+	Data +
4	GND	Ground

Tabulka 3 - Popis pinů konektoru USB typ B

Mikrokontroléry řady PIC18F2x5x/4x5x podporují USB komunikaci rychlostí 1,5 Mbit/s nebo 12 Mbit/s s plně hardwarovým řízením. Výrobce k nim pro tuto komunikaci poskytuje zdarma knihovnu CDC (Communication Device Class) včetně ovladačů pro OS Windows. Pomocí této knihovny může programátor jednoduše doplnit své stávající programy o USB komunikaci.

CDC knihovna obsahuje funkce pro odesílání řetězců hostiteli, přijímání řetězců od hostitele a sledování stavu sběrnice. Celá USB komunikace probíhá ve smyčce v main programu. Uživatel do smyčky ProcessIO pak vkládá vlastní část programu. Vzhledem k jednoduchosti celé aplikace musí uživatel „oželet“ používání přerušení, které by narušilo správný chod smyčky. Druhou volbou je přepsat celou aplikaci včetně USB komunikace na přerušovací systém, což vzhledem k dané aplikaci je zbytečně složité.

```
void main(void)
{
      InitializeSystem();     // inicializace systému
      while(true)             // nekonečná smyčka
      {
         USBTasks();          // volání obsluhy USB
         ProcessIO();         // funkce uživatele
      }                       // konec smyčky while
}                             // konec main
```

Obrázek 7 - Ukázka kódu pro CDC

3.1.3 Popis AD převodníku

Vnitřní modul ADC se skládá ze 4 částí a to vstupní multiplexer, vzorkovač, reference a vlastní AD převodník. Vstupní 8-mi kanálový multiplexer podle hodnot bitů CHS2-CHS0 vybere jeden ze vstupů AN7-AN0 a připojí ho na vstup vzorkovače (viz obr. 5). Pomocí bitů VCFG1- VCFG je nastaven zdroj referenčního napětí pro aproximační AD převodník. Spuštění vlastního převodu se provádí nastavením bitu GO/DONE do logické jedničky. Po ukončení převodu je tento bit automaticky nastaven do nuly. Toho se využívá k rozpoznání ukončení převodu, druhá možnost je povolení přerušení po skončení převodu. Výsledek převodu je uložen v registrech ADRESH, ADRESL.

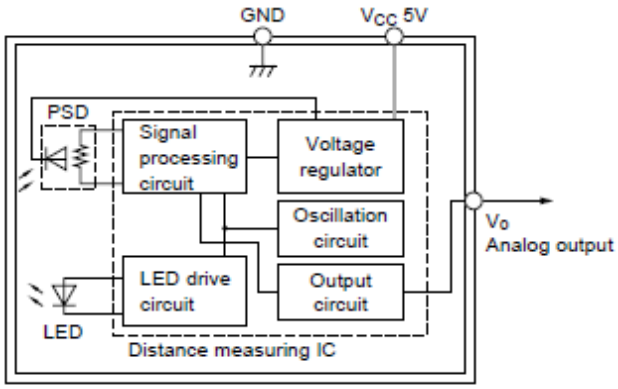
```
SetChanADC(ADC_CH0);                // nastavení vstupu
ConvertADC();                       // Start převodu
while( BusyADC() );                 // čekání na ukončení převodu
vysledek=(unsigned int) ReadADC();  // přečtení hodnoty
```

Obrázek 8 - Ukázka části zdrojového programu komunikace s ADC

3.1.4 Popis odrazového senzoru GP2D120

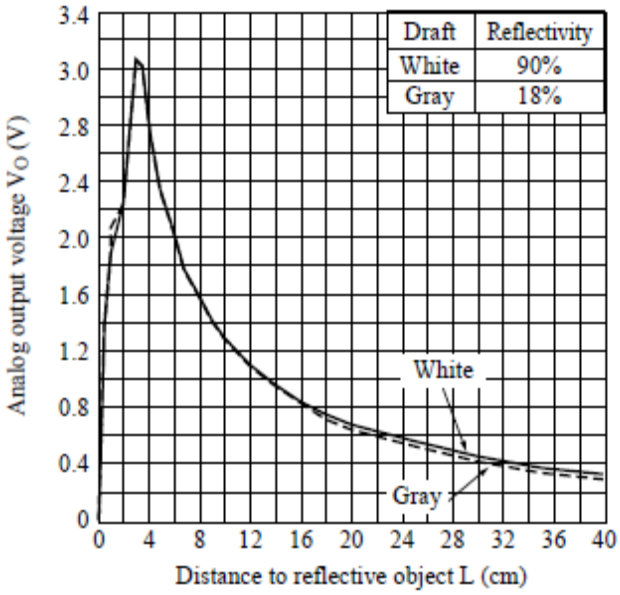
Senzor GP2D120 měří úhlovou odchylku (paralaxu) odraženého paprsku. Obsahuje výkonnou vysílací LED, několik přijímacích prvků, optiku a vyhodnocovací elektroniku. Vysílací LED zobrazí na předmětu intenzivně osvětlený bod, přijímací optika promítne

tento bod na řadu přijímacích prvků. Nejintenzivněji osvětlený prvek z řady pak udává úhel, pod kterým je vidět odražený paprsek. Aby senzor nemusel obsahovat velké množství jednotlivých přijímačů, vyhodnocuje se též intenzita signálu. Vestavěná elektronika vygeneruje příslušné výstupní napětí pomocí digitálně analogového převodníku.

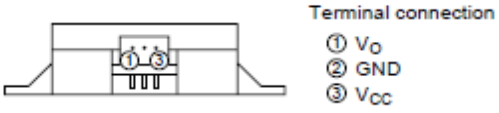


Obrázek 9 - Blokové schéma odrazového senzoru. Zdroj [12].

:



Obrázek 10 - Graf závislosti výstupního napětí na vzdálenosti překážky. Zdroj [12].



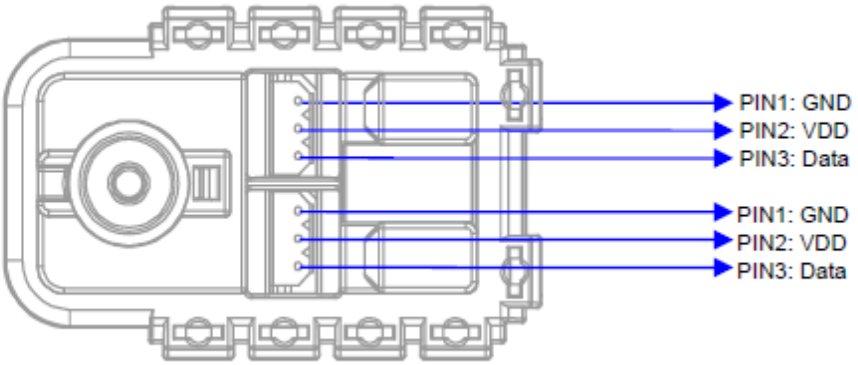
Obrázek 11 - Zapojení konektoru senzoru. Zdroj [12].

3.1.5 Popis servomotoru AX12

Servomotor řady Dynamixel je inteligentní modulární pohon, který obsahuje převodovku, precizní stejnosměrný motor a řídicí obvody s komunikací UART. Navzdory své kompaktní velikosti může produkovat vysoký točivý moment a je vyroben z vysoce kvalitních materiálů, které poskytují potřebnou pevnost a strukturální odolnost, aby vydržely velké vnější síly. Také má schopnost detekovat změnu vnitřní teploty nebo napájecího napětí.

Parametry servomotoru:

- přesné řízení polohy a rychlosti lze ovládat s rozlišením 1024 kroků (rozlišení 0,35 °)
- poskytuje zpětnou vazbu pro natočení, úhlové rychlosti a zatížení točivého momentu
- alarm systém pohonu může upozornit uživatele, když parametry se odchylují od uživatelem definovaného rozsahu (např. vnitřní teplota, točivý moment, napětí, atd.)
- snadné komunikace podporující komunikační rychlosti až 1 Mbit/s
- status LED dioda může zobrazovat chybový stav
- napětí 7 V~10 V (doporučené napětí: 9.6 V)
- maximální proud 900 mA
- provozní teplota -5~+85 °C



Obrázek 12 - Zapojení konektoru servomotoru. Zdroj [11].

3.2 PICKit3

PICkit3 je In-Circuit debugger/programátor od společnosti Microchip využívající obvodové ladící logiky zabudované v každém čipu s Flash pamětí k vytvoření nízkonákladového hardwarového debuggeru a programátoru. Obvodové ladění nabízí výhody minimalizace potřeby dodatečného hardwaru a také v eliminaci potřeby adaptérů. Součástí PICkit3 je 12 ukázkových úloh jak programovat jednotlivé periférie mikroprocesoru PIC v jazyce C.

Základní vlastnosti PICkit3:

- USB
- Podpora vývojového prostředí MPLAB IDE
- Zabudovaný monitoring přepětí a zkratu
- Možnost upgradu firmwaru pomocí PC
- Real-time ladění, breakpointy
- Podpora nízkého napětí (2.0 V až 6.0 V)
- LED Diagnostika (power/busy/error)
- Mazání programové paměti s ověřením

Obrázek 13 - Programátor PICKit3. Zdroj [4].

3.3 Popis modulu PUM

Návrh modulu PUM vychází z materiálu dostupných na webových stránkách firmy Microchip. Zde jsou v aplikačních ukázkách popsány a použity jednotlivé komponenty mikroprocesoru, jako např. AD převodník, USB rozhraní, apod. Vzhledem k požadované jednoduchosti jsem zvolil mikroprocesor PIC 18F2455, který obsahuje integrované komponenty pro splnění požadavků na měřící modul.

V současné době jsou dostupné dva podobné moduly s mikroprocesorem PIC, tyto se však zaměřují na programátory začátečníky a snaží se jim nabídnout jednoduché vývojové prostředí. Jedná se o projekty PINGUINO a AMICUS18. Více o těchto projektech nalezneme na webových stránkách www.hackinglab.org nebo www.myamicus.co.uk.

Pro měřící modul PUM jsem se však rozhodl jít vlastní cestou a začít navrhovat vlastní systém. Návrh schématu a PCB byl realizován v návrhovém systému EAGLE. Tento systém již využívám ve verzi Lite již cca 10 let a jsem s ním velmi spokojen. Pro výrobu PCB byla použita metoda přenosu toneru pomocí laminovačky. Tato metoda umožnuje jednoduchou výrobu DPS v amatérských podmínkách s velmi dobrou kvalitou. Princip spočívá ve vytištění předlohy DPS na lesklý křídový papír na laserové tiskárně. Vystřižený obrazec je pak přiložen na vyčištěnou desku kuprexidu tonerem směrem na měďěnou stranu desky. Potom je obrazec přichycen papírovou lepící páskou a vložen do laminovačky, která je nastavena na teplotu cca 180 ºC. Počet průchodů laminovačkou je přibližně 5 krát a potom je deska vložena do teplé, nejlépe tekoucí vody. Křídový papír se po několika minutách odplaví a na měděné straně kuprexidu zůstane požadovaný obrazec. Desku ponoříme do roztoku chloridu železitého a vyleptáme. Nanesený toner lehce seškrábneme pod vodou a desku ošetříme roztokem kalafuny.

Firmware pro mikroprocesor je napsán v jazyce C a využívá knihovny dodané firmou Microchip. Pro testování firmware jsem nemohl použít windowsovský Hyperterminál, neboť komunikace neprobíhá jen v ASCII, tak jsem si vytvořil vlastní testovací aplikaci. Po ověření funkčnosti firmware jsem se teprve pustil do psaní scriptů v prostředí MATLAB, kde jsem teprve zkušenosti a znalosti získával.

Testovací aplikaci jsem vzhledem k univerzálnosti a rozšířenosti programovacího jazyka Java vytvořil v prostředí Netbeans. Tato aplikace slouží pro čtení AD převodníku a po zvolení příslušného analogového vstupu změří vstupní napětí a výsledek zobrazí v konzoli. Pro USB komunikaci s modulem PUM jsem použil volně dostupnou komponentu GIOVYNET (http://java.giovynet.com/Giovynet). Tato komponenta se mně

vzhledem k jednoduchosti a snadné integraci do prostředí Netbeans zdála být nejvhodnější. Komponenta je dostupná pro obě platformy, a to Windows i Linux. Z komponenty používám tyto funkce:

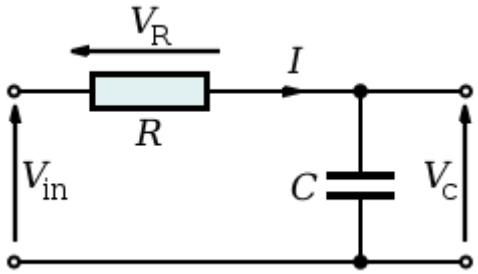
```
//inicializace sériového portu pro windows:
Parameters param = new Parameters();
param.setPort("COM4");
param.setBaudRate("9600");
Com com1 = new Com(param);

// zápis dat na sériový port
char posli[]={'@','A',0x01,'N','#'};    //ADC kanál 1
com1.sendArrayChar(posli,10);

// čtení sériového portu
pum_result = new char[10];
pum_result = comx.receiveToString(4).toCharArray();

//uzavření sériového portu
com1.close();
```

Obrázek 14 - Ukázka zdrojového kódu

Vzhledem k použitému USB portu nepotřebuje měřící modul žádné externí napájení. Modul obsahuje 4 digitální TTL vstupy. Pro jednoduchost modulu nejsou tyto vstupy galvanicky odděleny pomocí optočlenů. Napětí na 4 analogových vstupech je měřeno postupně pomocí integrovaného aproximačního převodníku. Integrovaný ADC využívá jako referenci napětí získané z USB rozhraní, což pro jednoduché měřící úlohy postačuje.

3.4 Měření RC článku

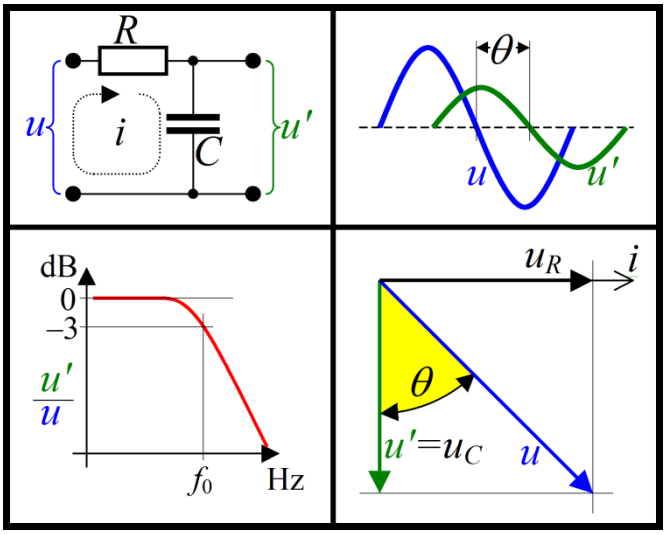
RC článek je lineární a pasivní elektrický obvod složený z rezistoru a kondenzátor, jímž se elektrický signál mění v závislosti na jeho kmitočtu. Užívá se zejména jako frekvenční filtr, například horní nebo dolní propust.



Vin - vstupní napětí,
VR - napětí na rezistoru R,
VC - napětí na kondenzátoru C

Obrázek 15 - Sériový RC článek. Zdroj [6].

Je-li kondenzátor C vybitý a na vstup se přivede skokově napětí V, začne se kondenzátor nabíjet. Rychlost nabíjení závisí na velikosti rezistoru R a kondenzátoru C, takže jeho časovou konstantu lze vyjádřit jako $\tau = RC$. Z toho pak plyne, že je-li vstupní frekvence nižší než frekvence vyjádřená vztahem $f = 1/\tau$, pak tento signál RC článek příliš neovlivní, kdežto pro vyšší vstupní frekvenci představuje RC článek stále větší impedanci.



Obrázek 16 - Ukázka dolní propusti. Zdroj [6].

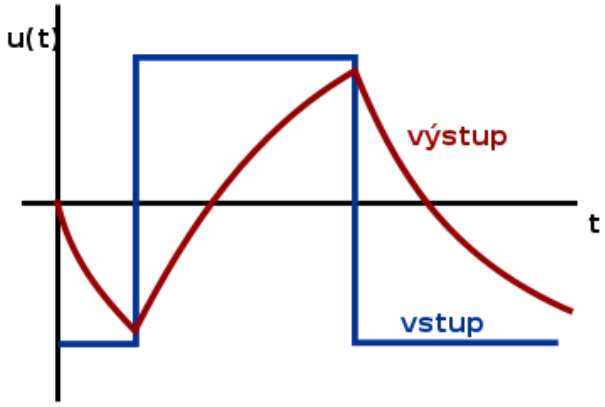
RC článek lze také matematicky popsat jako integrační článek, tj. v obvodu provádí matematickou funkci integrování – napětí na výstupu u_2 je integrálem napětí na vstupu u_1 podle času, viz obrázek č.2, kde K_i je konstanta integrátoru.

$$u_2(t) = K_\mathrm{i} \int_0^t u_1(t)dt$$

Obrázek 17 - Integrál. Zdroj [6].

Příklad výpočtu přechodového děje pro RC článek

Zadání: Vypočtěte výstupní napětí u_2 v čase 1 s, přivedeme-li na vstup RC článku skokově napětí u_1 o velikosti 10 V. Hodnota rezistoru je 1 kΩ a hodnota kondenzátoru je 100 µF. Průběh výstupního napětí u_2 v čase t nakreslete a graficky odečtěte hodnotu časové konstanty τ. Odečtené hodnoty porovnejte s vypočtenými.



Obrázek 18 - Odezva integračního článku RC na obdélníkové pulzy. Zdroj [6].

V prostředí MATLABu se pro výpočet určitého jednorozměrného integrálu využívá funkce *quadl (fce, a, b).* Zmíněnou úlohu vypočteme následujícím skriptem:

```
R=1000;
C=100E-06;
T=1/(R*C);
U1=10;
U2=(R*C)*quadl(U1,0,1);
disp(U2);
```

Obrázek 19 - Ukázka zdrojového kódu

3.5 Komunikace s modulem PUM-U

OS Windows

Po připojení modulu přes rozhraní USB k počítači je operačním systémem požadován ovladač. Ten je standardně dodávám firmou Microchip a nese název mchpcdc. Po úspěšné instalaci ovladače je krátce zobrazen název zařízení, v našem případě „Modul PUM-U“. Ve správci zařízení v sekci „Sériové porty“ vyčteme název virtuálního sériového portu, většinou COM5.

OS Linux

Po připojení modulu přes rozhraní USB k počítači není vyžadován žádný ovladač (platí pro jádro 2.6 a vyšší). Zařízení je detekováno jako CDC RS232 a je vytvořen virtuální port /dev/ttyACM0. Pro snadnější práci s portem si vytvoříme link ln -s /dev/ACM0 /dev/USB0 a dále se odkazujeme na zařízení /dev/USB0. Pokud chceme zobrazit všechny připojené USB zařízení k PC, využijeme linuxového příkazu lsusb.

3.5.1 Komunikace MATLABu přes sériové rozhraní

V prostředí MATLABu pro práci se sériovým portem používáme funkci serial, která nám vrací handle na sériové zařízení. Pomocí funkce fopen toto zařízení otevřeme a zahájíme s ním komunikaci. Ukázka testu komunikace s modulem PUM.

```
s = serial('COM1','BaudRate',9600);
fopen(s)

//test zarizeni
fwrite(s,'@TNN#')
out = fread(s)
//vypis odpovedi
disp(out)

//uzavreni portu
fclose(s)
delete(s)
clear s
```

Obrázek 20 - Ukázka zdrojového kódu

3. 5. 2 Čtení analogových vstupů

V MATLABu byla vytvořena funkce PUM_ADC, která má za vstupní parametry handle sériového portu a číslo analogového vstupu. Funkce vrací změřené napětí na vstupu. Ve funkci je použit přepočet 8-bitové hodnoty na napětí.

```
function napeti=PUM_ADC(PUMhandle,vstup)
% PUM_ADC - precte analogovou hodnotu na vstupu PUM

adc_command = zeros(1,5,'int8');
adc_command(1)='@';
adc_command(2)='A';
adc_command(3)=vstup;
adc_command(4)='N';
adc_command(5)='#';

% komunikace serial_port
fwrite(PUMhandle,adc_command);
adc_result=fread(PUMhandle,5);

% prepocet na napeti
napeti=(5/1024)*(256*adc_result(2) + adc_result(3));
end
```

Obrázek 21 - Ukázka zdrojového kódu

3.5.3 Čtení digitálních vstupů

V MATLABu byla vytvořena funkce PUM_DIO, která má promněný počet vstupních parametrů. Tato funkce je vytvořena jak pro čtení, tak i zápis digitálních vstupů/výstupů.

```
function result=PUM_DIO(varargin)
% PUM_DIO - precti/nastav digitální vstup

data=varargin(:);
% pocet_parametru=nargin;

PUMhandle=data(1);
mod=data(2);
pin=cell2mat(data(3));
% hodnota=data(4);

if strcmp(mod,'W')
    command = zeros(1,5,'uint8');
    command(1)='@';
    command(2)='E';     % zapis
    command(3)=pin;     % cislo pinu
    command(4)=cell2mat(data(4)); % hodnota
    command(5)='#';

else
    command = zeros(1,5);
    command(1)='@';
    command(2)='D';     % cteni
    command(3)=data(3); % cislo pinu
    command(4)='N';
    command(5)='#';

end

% komunikace serial_port
fwrite(PUMhandle,command);
result=fread(PUMhandle,4);
end
```

Obrázek 22 - Ukázka zdrojového kódu

Pro čtení digitálních vstupů se zadávají parametry handle sériového portu, volba čtení "R" a číslo digitálního vstupu.

```
%  precteni dig. vstupu
pin=PUM_DIO(PUMhandle,'R',2);
disp(pin);
```

Obrázek 23 - Ukázka zdrojového kódu

3.5.4 Ovládání digitálních vstupů

Pro ovládání digitálních výstupů se používá shodná funkce PUM_DIO jako pro čtení. U této funkce se zadají následující parametry, a to handle sériového portu, volba zápisu "W" a číslo digitálního výstupu a stav „0" nebo „1".

```
% nastaveni digitálního vystupu
PUM_DIO(PUMhandle,'W',2,1);
```

Obrázek 24 - Ukázka zdrojového kódu

3.5.5 Ovládání servomotoru AX12

V MATLABu byla vytvořena funkce PUM_AX12, která má za vstupní parametry handle sériového portu, typ příkazu a parametr příkazu. Parametrem příkazu volíme možnosti pro úhel natočení, rychlost pohybu a možnost ovládání LED signalizace.

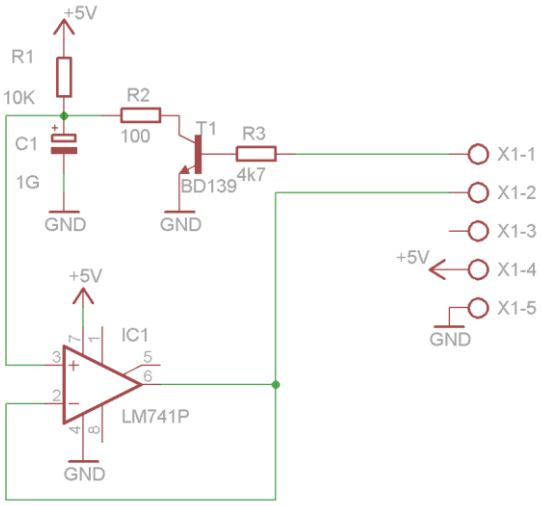
```
function result=PUM_AX12(PUMhandle,typ,param)
% PUM_AX12 - posle data na servomotor AX12

% U-uhel R-rychlost D-LED dioda

switch typ

    case 'U'
    [high,low]=AX_uhel(param);
    paket=AX_paket('1E',low, high);

    case 'R'
    paket=AX_paket('20',int2str(param),'00');

    case 'D'
    paket=AX_paket('19',int2str(param));

    otherwise
    paket={};
end

% vycteme pouze posledni data v paketu tj. delka_paket(2)-5
delka=size(paket);

if(delka~=0)
    cmd=paket(6:delka(2));
    disp(paket);          % zobrazeni paketu
    disp(cmd);            % zobrazeni cmd

% posleme na PUM - UART
    result=PUM_UART(PUMhandle,'AX',cmd); % posli data na UART pro AX
end
end
```

Obrázek 25 - Ukázka zdrojového kódu

3.6 Laboratorní přípravky pro modul PUM

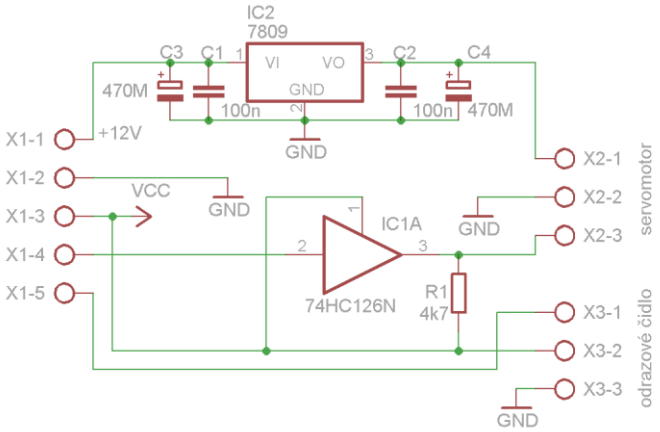
Měřící modul PUM byl rozšířen o pomocné přípravky pro snadnou realizaci laboratorních úloh.

První přípravek má označení P1 a slouží k měření RC obvodu v laboratorní úloze č. 2. Skládá se z integračního článku RC, operačního zesilovače zapojeného jako emitorový sledovač a tranzistoru sloužícího k vybíjení kondenzátoru. Integrační článek je tvořen součástkami R1 a C1 a jeho časová konstanta je zvolena na cca 10 sec. z důvodu získání co nejvíce vzorků z měření a případné eliminace chyby způsobené časovou prodlevou při komunikaci mezi MATLABem a modulem PUM.



Obrázek 26 - Schéma přípravku P1

Druhý přípravek nese označení P2 a slouží pro laboratorní úlohy č. 4 a č. 5. Tento přípravek realizuje rozhraní mezi servomotorem AX12 a měřícím modulem PUM. Dále obsahuje třísvorkový stabilizátor napětí LM7809 pro napájení servomotoru z externího 12 V zdroje. Pro komunikaci se servomotorem byla použita pouze jednosměrná komunikace, tj. odpovědi a chybové události od servomotoru jsou zakázané. Přípravek je také osazen svorkovnicí pro připojení odrazového čidla.

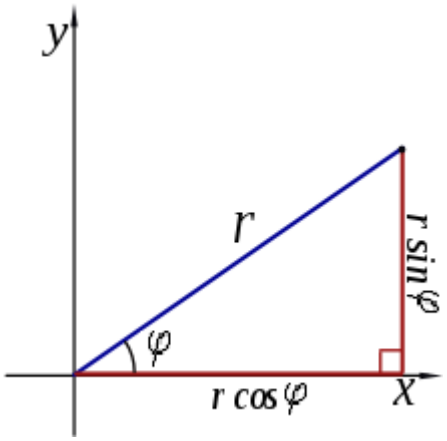


Obrázek 27 - Schéma přípravku P2

Třetí přípravek P3 je ve stádiu návrhu a bude sloužit pro měření stejnosměrného vstupního napětí v rozsahu 0 až 250 V. Zde se plánuje automatické i manuální přepínání měřícího rozsahu. Toto přepínání bude řešeno programovatelným děličem napětí, který bude ovládán digitálními výstupy modulu PUM. Automatického přepínání rozsahu bude řešeno skriptem v MATLABu.

3.7 Polární soustava souřadnic

Polární soustava souřadnic je taková soustava souřadnic v rovině, ve které je každý bod určen úhlem, který svírá spojnice bodu s počátkem a vzdáleností bodu od počátku. Úhel se měří proti směru hodinových ručiček. Pro označení vzdálenosti bodu obvykle používáme značku *r* a pro úhel písmeno řecké abecedy φ.

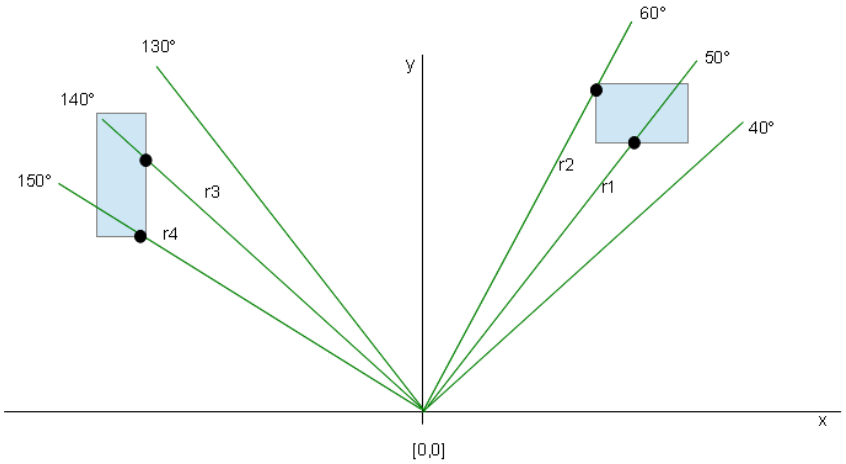


Obrázek 28 - Ukázka převodu polárních souřadnic na kartézské. Zdroj [6].

Převod polárních souřadnic na kartézské se provádí dle následujících vzorců:

$$x = r * \cos \varphi \qquad y = r * \sin \varphi$$

Aplikací této úlohy může být infračervený radar kybernetického vozítka, který vyhledává cestu, tj. vzdálenosti jednotlivých překážek od sebe a nalezení správné šířky pro průjezd vozítka.



Obrázek 29 - Ukázka úlohy pro výpočet

Pro ukázku výpočtu je zvolena situace se dvěma překážkami, vzdálenosti a úhel je uveden v tabulce č. 5. Body odrazu od překážek jsou vyznačeny v obrázku černou tečkou.

	Bod A	Bod B	Bod C	Bod D
úhel [°]	50	60	140	150
vzdálenost [cm]	87	95	96	84

Tabulka 4 - Naměřené hodnoty v polární soustavě

Pro přepočet souřadnic v MATLABu je využit následující skript, úhel a vzdálenost je uložena v promněných *uhel[]*, *vzdalenost[]*, výstupní promněné (kartézské souřednice) jsou uloženy v *x[]* a *y[]*.

```
N=length(uhel);
x = zeros(1,N);
y = zeros(1,N);
for i=1:N
    x(i)=vzdalenost(i)*cos(uhel(i));
    y(i)=vzdalenost(i)*sin(uhel(i));
end
```

Obrázek 30 - Ukázka zdrojového kódu

Pro výpočet šířky mezi dvěma překážkami lze využít následující algoritmus:

- body procházíme zprava doleva
- první bod zůstává
- zredukování počtu bodů – pokud následující bod je od testovaného vzdálen o krokovací úhel, je testovací bod smazán
- pro výpočet šířky použijeme vnitřní dva body

V MATLABu pro výpočet využijeme následující skript:

```
N=length(uhel);
pom = zeros(1,4);
p=1;
pom(p++)=x(1);
for i=2:N
     if((x(i+1)-x(i))>10) pom(p++)=x(i);
end
sirka=abs(pom(2)-pom(3));
```

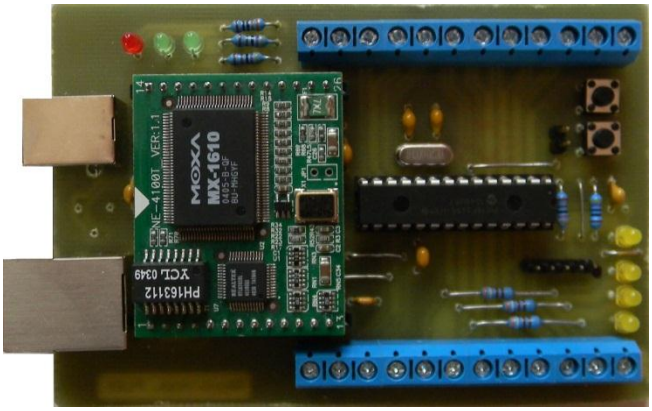
Obrázek 31 - Ukázka zdrojového kódu

4 Měřicí modul verze Ethernet

Měřicí modul PUM-E je napájen prostřednictvím 5 V adaptéru nebo přes konektor USB a obsahuje:

- 4 analogové vstupy s rozsahem 0–5 V
- 4 digitální vstupy TTL kompatibilní
- 4 digitální výstupy TTL kompatibilní

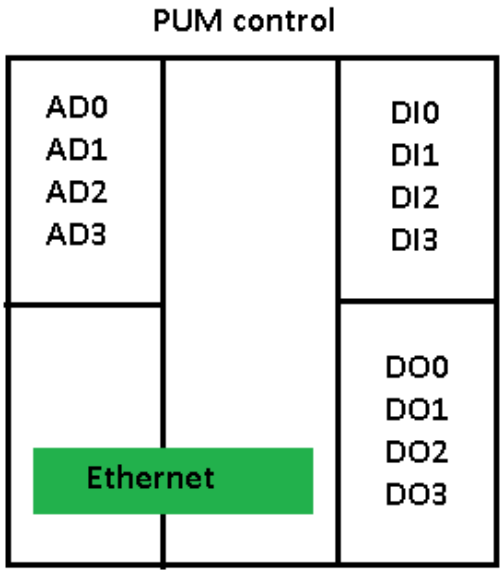
Schéma a návrh DPS včetně osazení je uveden v příloze D.



Obrázek 32 - Fotografie zkušebního modulu PUM-E

4.1 Popis hardware

Modul PUM verze 2 vychází z verze 1 a je rozšířen o převodník rozhraní Ethernet/sériová linka. Dále byl změněn firmware, který byl celý přepsán a běží pod RTOS. Výhodou této verze je možnost vzdáleného měření, popř. ovládání určitého zařízení. Pro převodník Ethernet/sériová linka byl použit modul NE-4100T od firmy Moxa zabývající se výrobou průmyslových převodníků. Výhodou zásuvného modulu je napájení 5 V a možnost programování v jazyce C. Volba RTOS pro mikroprocesor PIC18F vzhledem k omezenému výběru (FreeRTOS, OSA) padla na OSA, vzhledem k jednoduchosti. OSA je kooperativní RTOS (real-time operační systém) pro PIC, AVR, STM8 a snadno se integruje do vývojového prostřední MPLAB.



Obrázek 33 - Modul PUM verze Ethernet

Pro účely vývoje a testování tohoto modulu byl vytvořen jednoduchý program v Javě v prostřední Netbeans. Tento program po zadání IP adresy modulu PUM cyklicky přenáší údaje z/do modulu, proto je možné sledovat zpožděnou reakci na požadovanou změnu. V případě chyby komunikace nebo chybně zadané adresy je zobrazena chyba. Zato aplikace je typu klient-server a komunikuje na portu 4001. Tento port je pevně daný v programu a nelze ho měnit. Uvedený program lze spustit z příkazové řádky zadáním příkazu java -jar PUMControlE.jar.

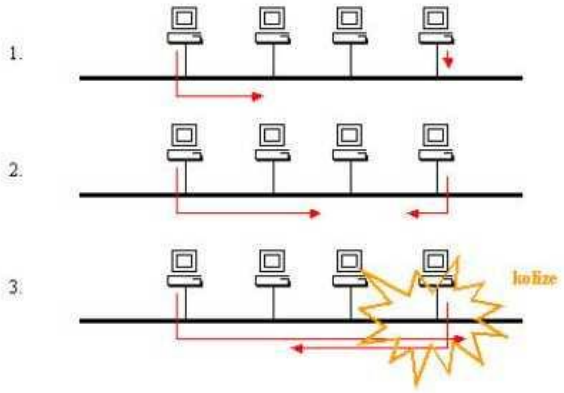
4.1.1 Popis rozhraní Ethernet

V současné době je nejpoužívanější síťovou technologií Ethernet. Tato technologie je, nezávisle na tom, zda jde klasický 10 Megabitový Ethernet nebo jeho rychlejší mutace (Fast a Gigabit Ethernet), založena na velice jednoduchém principu, nazývaném CSMA/CD.

CSMA (Carrier Sense Multiple Access) - stanice připravená vysílat data si "poslechne", zda přenosové médium (kabel) nepoužívá jiná stanice. V případě, že ano, stanice zkouší přístup později až do té doby, dokud není médium volné. V okamžiku, kdy se médium uvolní, začne stanice vysílat svá data.

CD (Collision Detection) - stanice během vysílání sleduje zda, je na médiu signál odpovídající vysílaným úrovním (tedy aby se např. v okamžiku, kdy vysílá

signál 0 nevyskytl signál 1). Případ, kdy dojde k interakci signálů více stanic, se nazývá kolize. V případě detekce kolize stanice generuje signál JAM a obě (všechny) stanice, které v daném okamžiku vysílaly, generují náhodnou hodnotu času, po níž se pokusí vysílání zopakovat.



Obrázek 34 - Ukázka kolize paketu. Zdroj [13].

Popis fází:

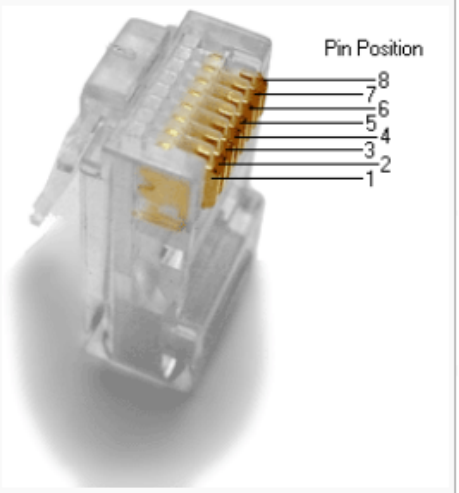
fáze 1 - stanice vlevo si poslechla na „drátu“, zda někdo vysílá, zjistila, že ne a začala sama posílat data, v okamžiku, kdy ještě signál nedorazil ke stanici vpravo, si tato stanice ověřila stav média, zjistila, že je možnost zahájit vysílání

fáze 2 – obě stanice posílají data

fáze 3 – stanice vpravo zjistila kolizi a generuje signál JAM, všechny vysílající stanice zastavují vysílání a generují náhodné číslo pro zpoždění

RJ-45

Koncovka RJ-45 je dnes nejčastěji používaný typ zapojení síťových kabelů UTP a STP. Je to koncovka typu 8P8C (z angličtiny: 8 pozic, 8 vodičů). RJ-45 může mít dvě podoby: zásuvka nebo zástrčka. Pro kabeláž v kategorii 5E se používá kroucená dvojlinka (anglicky Twisted pair cable) a zapojení s označeními T568A nebo T568B, která jsou definována v TIA/EIA-568-B. K nasazení koncovek se používají tzv. krimpovací kleště.

Pin	Standardní zapojení (T568A)	Standardní zapojení (T568B)	Označení pinů na konektoru
1	zeleno-bílý	oranžovo-bílý	
2	zelený	oranžový	
3	oranžovo-bílý	zeleno-bílý	
4	modrý	modrý	
5	modro-bílý	modro-bílý	
6	oranžový	zelený	
7	hnědo-bílý	hnědo-bílý	
8	hnědý	hnědý	

Obrázek 35 - Zapojení konektoru RJ45. Zdroj [13].

Standardní zapojení UTP kabelu

Přímý kabel

U přímého kabelu jsou oba konce zapojeny identicky, dle standardu T568A nebo T568B. Při výrobě „přímého kabelu“ se může stát, že se člověk soustředí pouze na to, aby oba konektory byly zapojeny navzájem stejně (1-1, 2-2, …), a nehledí na správné uspořádání párů kroucené dvojlinky. Při komunikaci vyšší rychlostí však takový kabel nebude fungovat kvůli vysokofrekvenčnímu rušení, které by správně mělo být eliminováno korektním zapojením párů. Když už se tedy v praxi nehledí na přesné rozmístění barev podle výše uvedeného schématu, je důležité alespoň správně zapojit páry podle obrázku vpravo.

Křížený kabel

Kabel, který má na koncích přehozený oranžový pár se zeleným (piny 1+2 versus 3+6) a v případě gigabitového Ethernetu ještě modrý pár s hnědým (piny 4+5 versus 7+8), se nazývá „křížený“ (anglicky crossover). Takovéto kabely jsou většinou označeny nápisem nebo barevně. Používají se k přímému propojení počítače s počítačem a před rozšířením auto-MDI/MDIX (automatické rozpoznání

přímého nebo kříženého kabelu) byl také potřeba k propojení mezi směrovači, rozbočovači nebo přepínači.

4.1.2 Popis komunikačního modulu Moxa

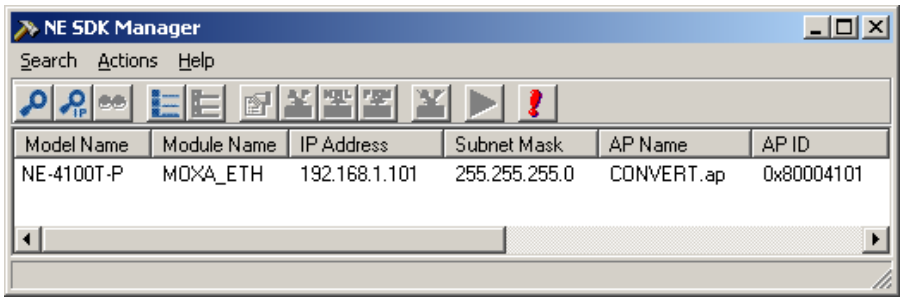
Moxa NE-4100 je embedded sériový server sloužící k připojení libovolného zařízení se standardním sériovým rozhraním na Ethernet. Server podporuje 10/100 Mbit/s Ethernet a nabízí řadu provozních režimů jako TCP server, TCP client nebo UDP komunikaci. Dále nabízí i z důvodu zpětné kompatibility i ovladač pro Real COM. Celé zařízení je napájeno +5 V a sériové rozhraní je typu TTL což umožnuje přímé propojení s mikroprocesorem PIC.

Obrázek 36 - Modul Moxa. Zdroj [16].

Po zakoupení modulu NE4100T lze jednoduchou změnou firmware získat programovatelný modul, což v praxi znamená možnost umístit vlastní program na tento modul a ovlivnit tak jeho chování dle našich požadavků. Tuto možnost jsem v dané konstrukci využil.

Nejdříve musíme na PC nainstalovat dodávané vývojové prostředí NESDK, viz obr. č. 3. Pomocí tohoto software přehrajeme nejen samotný firmware, ale také umístíme vlastní aplikaci do modulu. Dále tento SW umožnuje i ladění naší aplikace přes rozhraní Ethernet.

Výrobce Moxa dodává k tomuto serveru API knihovnu s velkým počtem metod, které umožňují napsat libovolnou aplikaci, která je pouze omezena celkovou velikostí cca 64 kB.



Obrázek 37 - NE SDK Manager

Základ aplikace vychází z ukázkové úlohy ServerTCP2, kterou dodává výrobce. Tako aplikace je rozšířena a upravena pro mé účely. Aplikaci lze rozdělit do několika částí:

- otevření sériového rozhraní modulu a nastavení přenosové rychlosti 56700 b/s
- otevření TCP portu a čekání na zahájení komunikace s klientem
- vlastní komunikace s klientem, resp. převod dat mezi sériovou linkou a Ethernetem
- ukončení komunikace

Popis firmware

Mikroprocesor PIC pracuje jako slave zařízení, proto komunikaci zahajuje vždy počítač. Z tohoto důvodu je modul NE4100T naprogramován jako TCP Server a naslouchá požadavkům od uživatelského software (PC). Po zahájení komunikace jsou TCP pakety přeposílány na sériové rozhraní. Toto řešení je zjednodušeně popsáno v ukázce programu.

```
void main(){
      while(1)
      {     sio_open(sport); //otevření sériového portu
            tcp_open(tcpport);    //otevření socketu
            while (tcp_listen(shd, 20000) ); //čekání na klienta
            //jsou-li v socketu data tak je přepošli na sériový
      port
            if(tcp_iqueue(shd)){  tcp_recv(shd, sbuf, len);
                              sio_write(sport,sbuf,len);
                              }
            //jsou-li v sériovém portu data tak je přepošli na
      socket
            if(sio_iqueue(sport)){      sio_read(sport,sbuf,len)
                  tcp_send(shd,sbuf,len);
            }
            tcp_close(shd);       //uzavření socketu
            sio_close(sport);     //uzavření sériového portu
      }
}
```

Obrázek 38 - Ukázka zdrojového kódu

4.2 Popis RTOS OSA

OSA je real-time operační systém (RTOS) využívající kooperativní multitasking pro mikroprocesory Microchip PIC řady PIC10 a PIC12 a PIC16, PIC18, PIC24, dsPIC, pro Atmel AVR8 a pro STMicroelectronics STM8. RTOS umožňuje programátorovi zaměřit se na řešení úkolů, na řešení problémů v algoritmech nebo matematických výpočtech bez nutnosti řešit sekundární úkoly, jako např. přepínání procesů.

Všechny sekundární úkoly plní jádro OSA a lze je rozdělit do následujících skupin:

- přepínání mezi paralelními procesy (např. skenování klávesnice, výstup dat na displeji, spínací relé)
- kontrolu časových limitů, výpočet zpoždění
- nalezení připraveného úkolu s nejvyšší prioritou a jeho provedení
- výměna dat mezi různými úkoly pomocí semaforů, zpráv, front

Kooperativní multitasking vyžaduje aktivní spoluúčast běžících úloh. Každá úloha je povinna po určitém času prostřednictvím systémového volání předat řízení zpět operačnímu systému, který díky tomu může spustit jinou úlohu, která

se po chvíli opět dobrovolně vzdá procesoru atd. Výhodou řešení je jednodušší implementace operačního systému. Podstatnou nevýhodou je skutečnost, že chybně naprogramovaná úloha, která nevrátí řízení zpět operačnímu systému, způsobí úplné zastavení systému i ostatních úloh.

Základem OSA je C-funkce, která musí obsahovat nekonečnou smyčku tvořenou cyklem for a v ní alespoň jednu službu, která předá řízení zpět operačnímu systému.

Jednoduchá úloha (SimpleTask) může vypadat například takto:

```
void SimpleTask (void) {
unsigned int i=0;

// nekonečná smyčka
for (;;) {
      i++;              //vlastní kód
      OS_Yield ();      //nepodmíněné přepínání kontextu
         }
}
```

Obrázek 39 - Ukázka zdrojového kódu

V rámci řešení vlastních úloh jsou využity v programu následující funkce OSA:

OS_Init – inicializace systémových promněných

OS_Task_Create – vytvoření tasku/úlohy

OS_EI() – povolení přerušení

OS_Run – spuštění scheduleru/plánovače

OS_Timer – inkrementace systémových časovačů

OS_Delay – generování zpoždění

OS_Wait – čekání na událost

4.3 Komunikace s modulem PUM-E

Pro komunikaci s modulem PUM-E je použita funkce tcpip. U této funkce je zvolena syntaxe se dvěma parametry, a to IP adresa hosta a jeho port. Následující obrázek obsahuje ukázku základní síťové komunikace s modulem popisujícím test připojeného modulu PUM-E.

```
t = tcpip('localhost',4012);   % vytvoreni TCPIP objektu
fopen(t);                      % otevreni TCPIP objektu
fwrite(t,65:74);               % zaslani dotazu na modul
rd = fread(t, 10,`uchar`);     % precteni odpovedi
fclose(t)                      % uzavreni TCPIP obj
```

Obrázek 40 - Ukázka zdrojového kódu

5. Laboratorní úlohy s modulem PUM

Součástí bakalářské práce je pět laboratorních úloh zaměřených na využití měřicího modulu PUM.

První úloha je zaměřena na základní seznámení s modulem PUM a obsahuje měření analogového napětí z regulovatelného laboratorního zdroje. Její zadání je obsaženo v příloze A. Pro obě verze modulu PUM je použit stejný skript, liší se pouze v použití odlišných komunikačních funkcí, které jsou zvoleny podle parametru USB_ETH. Tento parametr může nabývat pouze dvou hodnot. Hodnota parametru rovna 0 je určena pro modul s USB rozhraním, hodnota parametru rovna 1 je určena pro modul s Ethernet rozhraním.

```
USB_ETH=0;      %USB=0 ETH=1
x_mereni=11;    %pocet mereni

s_port=OpenPort('COM4',USB_ETH);

for i=1:x_mereni
napeti=PUM_ADC(s_port,0);
%zobraz
str=strcat('Namerene napeti je: ',num2str(napeti),' V');
disp(str);
disp(' ');
input('Stiskni klavesu ENTER');
end

ClosePort(s_port);
```

Obrázek 41 - Ukázka zdrojového kódu

Druhá laboratorní úloha je zaměřena na měření RC obvodu, viz příloha A. Zde se student naučí nejen měřit napětí v závislosti na čase, ale i pro výsledný naměřený průběh vytvořit graf. Součástí této úlohy je i ovládání digitálního výstupu určeného pro ovládání spínacího tranzistoru T. Skript je určen pro obě verze modulu PUM, viz parametr USB_ETH.

```
USB_ETH=0;   %USB=0 ETH=1

s5 = 'Měření probíhá po dobu cca. 1 minuty';
disp(s5);

% mereni charakteristiky
s_port=OpenPort('COM4',USB_ETH);

% vybijeni kondenzatoru
PUM_DIO('W',0,0);
pause (15);


% mereni ADC
x_mereni=50*2;        % 10s...0.5x20
napeti = zeros(1,x_mereni);

% nabijeni kondenzatoru
PUM_DIO('W',0,1);

for i=1:x_mereni
napeti(i)=PUM_ADC(s_port,0);
pause (0.5)
end
ClosePort(s_port);
```

Obrázek 42 - Ukázka zdrojového kódu

Třetí laboratorní úloha je zaměřena na kalibraci čidla vzdálenosti GP2D120 využívajícího se pro detekci vzdálenosti překážek u robotických vozítek. Zadání úlohy je obsaženo v příloze C. Zde si student procvičí výpočty s polynomy, výsledný polynom třetího stupně je pak porovnán s naměřenou závislostí. Úloha je určena pro obě verze modulu PUM, viz parametr USB_ETH.

```
s1 = 'Polynom ma rovnici d=13.268/(U+0.029)-0.7';
disp(s1);
%promnena napeti obsahuje zmerene hodnoty
delka=4:1:x_mereni+3;

rovnice=13./(delka.+0.7).-0.029;
plot(delka,napeti,delka,rovnice);
grid;
title ('Kalibrace cidla GP2D120');
xlabel ('vzdalenost prekazky');
ylabel ('napětí');
axis([4,30,0,3.5]);
```

Obrázek 43 - Ukázka zdrojového kódu

Čtvrtá laboratorní úloha je určena pouze pro verzi s USB rozhraním. Tato úloha je zaměřena na ovládání serva AX12. Zadání úlohy je obsaženo v příloze D. Komunikace se servem probíhá sériově rychlostí 9600b/s. Pro natáčení serva o požadovaný úhel lze použít níže uvedený skript. V modulu PUM-U jsou implementovány dvě funkce pro generování komunikačního protokolu serva AX12. V případě ovládání jiných parametrů než rychlost otáčení a úhel natočení je nutné doplnit firmware PUM-U o další komunikační funkce.

```
USB_ETH=0;                              %USB=0 ETH=1
x_count=11;                             %pocet natoceni
uhel=0;

s_port=OpenPort('COM4',USB_ETH);

for i=1:x_count
% vypocet hodnoty pro natoceni
hodnota=round(uhel*1023/300);
uhel=uhel+30;
PUM_AX12(s_port,'U',hodnota);

input('Stiskni klavesu ENTER');
end

ClosePort(s_port);                      %uzavreni
```

Obrázek 44 - Ukázka zdrojového kódu

Pátá poslední úloha je vychází z laboratorních úloh tři a čtyři. Je určena pouze USB verzi modulu PUM. Zadání úlohy je obsaženo v příloze E. Cílem této úlohy je ukázka použití polárních souřadnic a jejich převod do kartézské soustavy. Tomuto problému je věnována kapitola 3.7.

```
for i=1:x_count

% vypocet hodnoty pro natoceni
hodnota=round(uhel*1023/300)
PUM_AX12(s_port,'U',hodnota);
pause (15);
uhel=uhel+30;

napeti(i)=PUM_ADC(s_port,0);
delka(i)=13.268/(napeti(i)+0.029)-0.7
end
```

Obrázek 45 - Ukázka zdrojového kódu

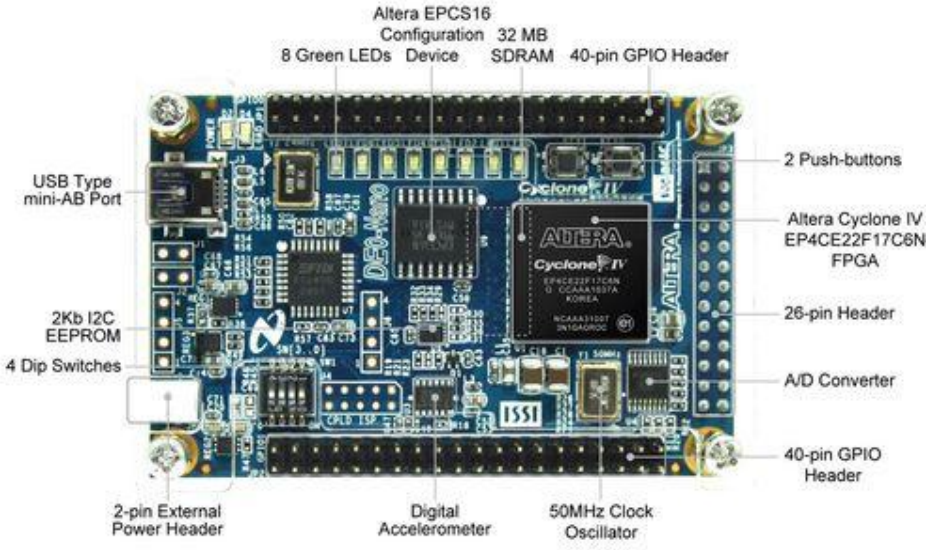
Cílem těchto laboratorní úloh bylo praktické využití MATLABu v laboratořích a procvičení si některých kapitol MATLABu na praktických příkladech. Vzhledem k univerzálnosti měřicího modulu PUM lze oblast těchto úloh rozšířit například o měření teploty, tlaku, vlhkosti nebo pro vysvětlení principu a činnosti čidel, jako jsou například akcelerometr, magnetický kompas.

6 Závěr

Při návrhu tohoto zařízení jsem si prohloubil své znalosti v oblasti mikroprocesorové techniky, zvláště v oblasti USB komunikace. Dále jsem se seznámil s tvorbou skriptů v prostředí MATLAB a s komunikací s externím zařízením pro poskytování dat.

Navrhnuté zařízení je použitelné v širokém spektru aplikací, ve kterých je potřeba změřit hodnoty připojených signálů, ovládat části zařízení, apod. Pro provoz zařízení stačí pouze 5 V napájení získané z USB portu, popř. z externího zdroje. Verze Ethernet je vhodná pro vzdálený sběr dat a ovládání.

V současné době pracuji na verzi realizované pomocí FPGA, jmenovitě na bázi kitu DE0-Nano od firmy Terasic Technologies Inc. Toto řešení mi nabízí větší variabilitu díky softprocesoru NIOS II a dále možnosti vytvářet uvnitř FPGA další moduly jako například digitální filtry, PWM, apod. Zde bych se chtěl seznámit a prakticky vyzkoušet HDL Coder od MATLABu, který přináší možnost generovat HDL kód přímo z algoritmů a funkcí vytvořených v tomto jazyce.



Obrázek 46 - Kit DE0-Nano. Zdroj: [15]

Z mého pohledu je zadání semestrální práce splněno a umožňuje použití modulů při praktických ukázkách využití MATLABu. Přesnost měření pro dané jednoduché úlohy je dostačující. Vzhledem k absenci ochrany vstupů mikroprocesoru je zařízení náchylnější k poškození, avšak cena mikroprocesoru je

zanedbatelná oproti ceně řešení s galvanickým oddělením vstupů/výstupů. Celé zařízení je navrhnuto za použití volně dostupných vývojových nástrojů a je připraveno pro možnou malosériovou výrobu.

Použitá literatura

[1] ZAPLATÍLEK, Karel. *MATLAB pro začátečníky*. 2. vyd. Praha: BEN - technická literatura, 2005, 151 s. ISBN 80-730-0175-6.

[2] ZAPLATÍLEK, Karel. *MATLAB*: *začínáme se signály*. 1. vyd. Praha: BEN - technická literatura, 2006, 271 s. ISBN 80-730-0200-0.

[3] MATOUŠEK, David. *Práce s mikrokontroléry PIC18F452 a PIC18F1220 v jazyce C*. 1. vyd. Praha: BEN - technická literatura, 2011, 368 s. ISBN 978-80-7300-413-2.

[4] *Microchip Technology Inc* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.microchip.com

[5] *HW.cz*: *Vše o elektronice a programování* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.hw.cz

[6] *Wikipedie*: *otevřená encyklopedie* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://cs.wikipedia.org

[7] *Humusoft*: *Technické výpočty, řídicí technika, simulace* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.humusoft.com/

[8] *MATLAB*: *The Language of Technical Computing* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.matlab.com

[9] *Eagle Software*: *Plošné spoje snadno a rychle* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.eagle.cz/

[10] *Oracle*: *Hardware and Software, Engineered to Work Together* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.oracle.com/

[11] *CrustCrawler Robotics*: *The worlds leader in providing cutting edge Robotics kits, Robotic Arms* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.crustcrawler.com/

[12] *Snail Instruments*: *Robotu nechte robotům!* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://shop.snailinstruments.com/

[13] *Svět sítí*: *Informace ze světa počítačových sítí* [online]. c2012 [cit. 2012-07-26]. Dostupné z: http://www.svetsiti.cz/

[14] *OSA*: *Introduction [PIC24]* [online]. 2010 [cit. 2012-07-26]. Dostupné z: http://www. pic24.ru/

[15] *Terasic Technologies: Expertise in FPGA/ASIC Design* [online]. 2012 [cit. 2012-07-26]. Dostupné z: http://www.terasic.com.tw/en/

[16] *Moxa: Device Networking for Industry* [online]. 2012 [cit. 2012-07-26]. Dostupné z: http://www.moxa.com/

Přílohy

Všechny přílohy, včetně zdrojových kódů a bakalářské práce v elektronické podobě, jsou k dispozici na přiloženém CD.

Příloha A – zadání laboratorních úloh

Příloha B – řešení laboratorních úloh

Příloha C – schémata modulů PUM

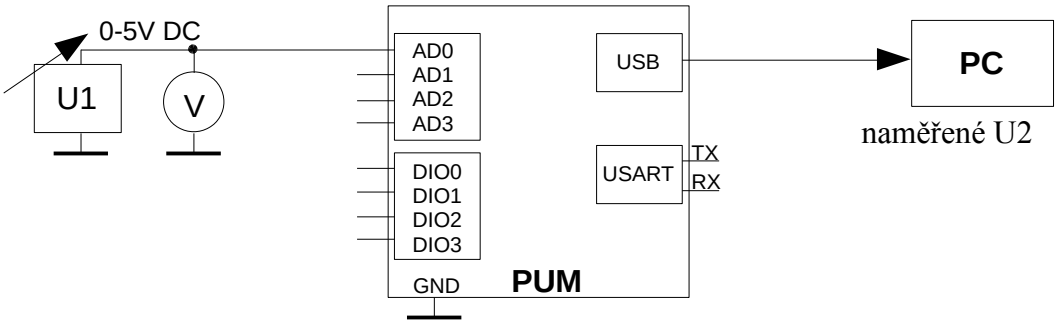
Příloha A – zadání laboratorních úloh

Laboratorní úloha č.1

Zadání:

Sestrojte obvod podle schématu a napište skript v prostředí MATLABu pro měření analogového napětí na vstupu AD0 modulu PUM. Naměřené hodnoty zapište do tabulky a sestrojte graf.

Schéma:



Tabulka:

U1[V]	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
U2[V]											

Script:

Graf:

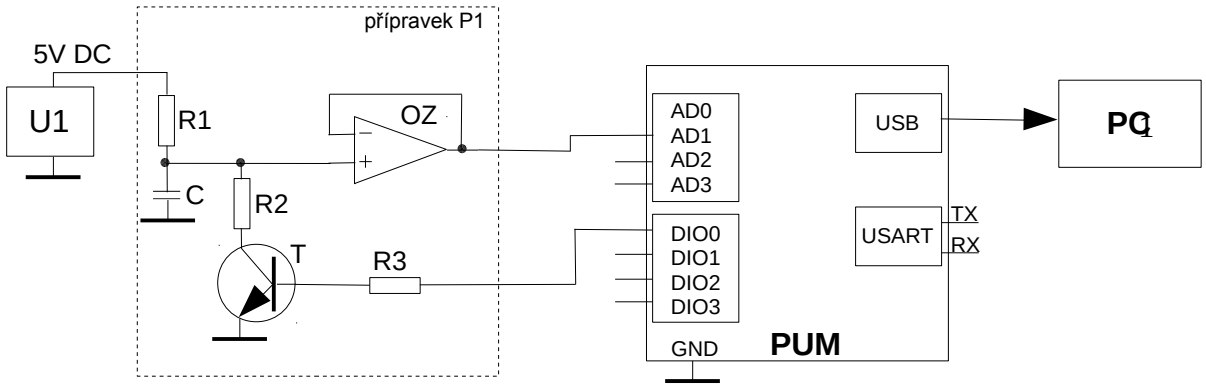
Závěr:

Laboratorní úloha č.2

Zadání:

Sestrojte RC obvod podle schématu s využitím přípravku P1, napište skript v prostředí MATLABu pro měření časového průběhu napětí na kondenzátoru C. Vypočtěte časové konstanty T1 (R1,C) a T2 (R2,C).

Schéma:



R1=10kΩ R2=100Ω C=1000 uF/6 V

Tabulka:

t [s]	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6
U1 [V]											

Script:

Graf:

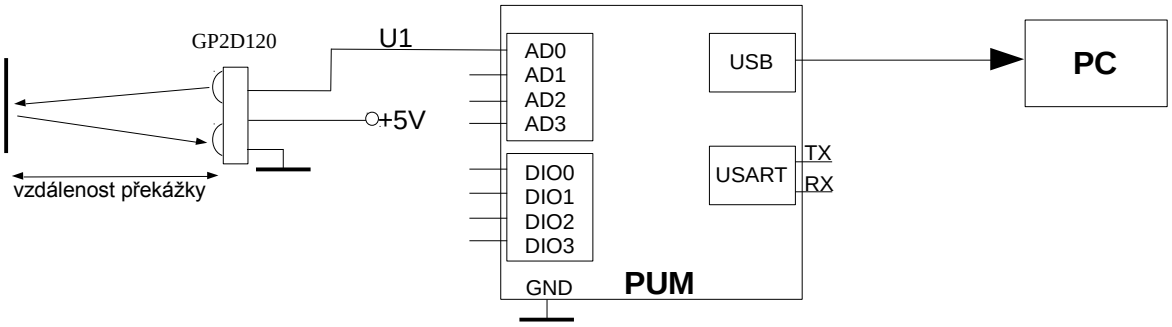
Závěr:

Laboratorní úloha č.3

Zadání:

Sestrojte obvod podle schématu, napište skript v prostředí MATLABu a proveďte kalibraci čidla vzdálenosti GP2D120. Naměřené hodnoty zapište do tabulky a výslednou závislost znázorněte do grafu. Změřenou závislost vyjádřete polynomem třetího stupně a daný polynom zobrazte.

Schéma:



Tabulka:

vzdálenost [cm]	4	5	6	7	8	9	10	11	12	13	14
U1 [V]											

vzdálenost [cm]	15	16	17	18	19	20	21	22	23	24	25
U1 [V]											

Script:

Graf:

Polynom:

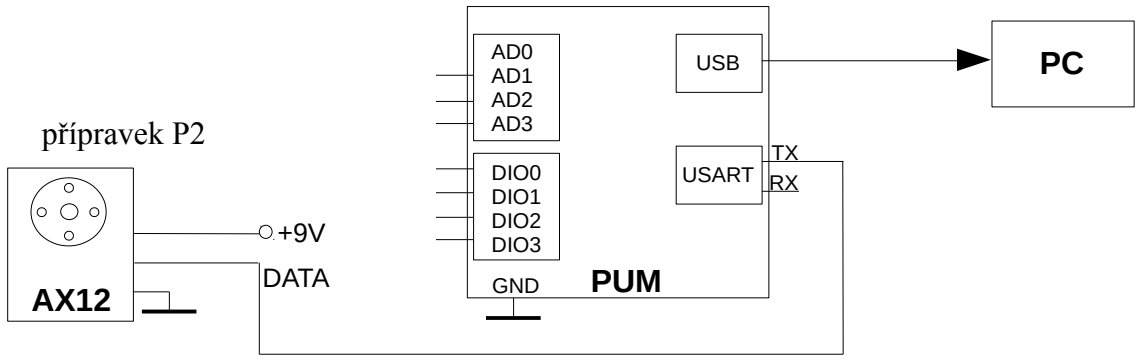
Závěr:

Laboratorní úloha č.4

Zadání:

Sestrojte obvod podle schématu s využitím přípravku P2 a napište skript v prostředí MATLABu pro komunikaci se servem AX12. Nastavená komunikační rychlost serva je 9600 b/s a hodnoty registru Goal Position (GP) pro požadovaný úhel jsou uvedeny v tabulce.

Schéma:



Tabulka:

úhel[°]	0	30	45	60	90	120	135	150	180
reg. GP									

Script:

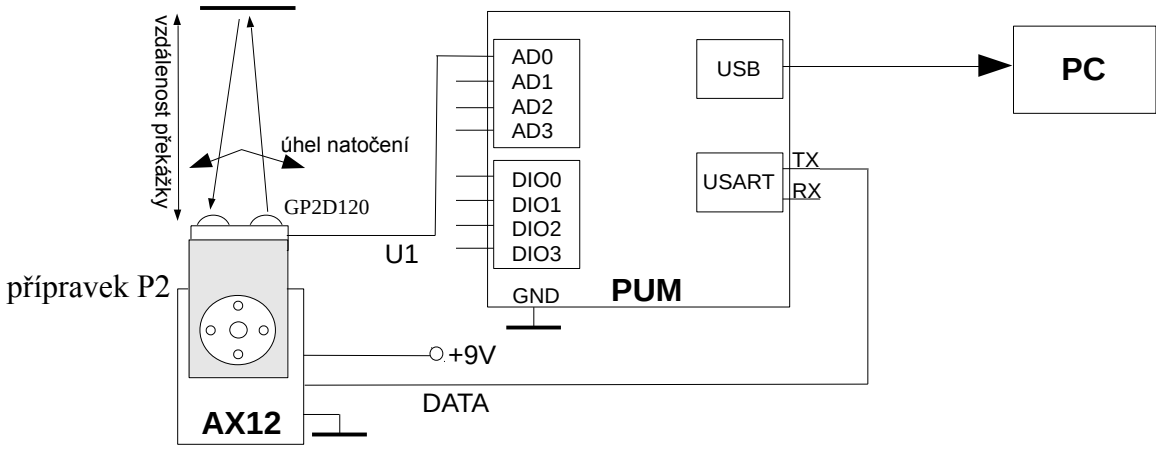
Závěr:

Laboratorní úloha č.5

Zadání:

Sestrojte obvod podle schématu s využitím přípravku P2 a napište skript v prostředí MATLABu pro detekci překážek v závislosti na úhlu natočení. Naměřené hodnoty zapište do tabulky a nakreslete rovinný graf zobrazující detekované překážky.

Schéma:



Tabulka:

úhel[°]	0	30	45	60	90	120	135	150	180
vzdálenost [cm]									

Script:

Graf:

Závěr:

Příloha B – řešení laboratorních úloh

Řešení laboratorní úlohy č.1

Script:

```
% script pro ulohu c.1

USB_ETH=0;        %USB=0 ETH=1
x_mereni=11;      %pocet mereni

s_port=OpenPort('COM4',USB_ETH);

for i=1:x_mereni
napeti=PUM_ADC(s_port,0);
%zobraz
str=strcat('Namerene napeti je: ',num2str(napeti),' V');
disp(str);
disp(' ');
input('Stiskni klavesu ENTER');
end

ClosePort(s_port);
```

Řešení laboratorní úlohy č.2

Script:

```
% script pro ulohu c.2
% vypocte a zobrazi prubeh nabijeni kondenzatoru
% nameri skutecne hodnoty a zobrazi prubeh

USB_ETH=0;   %USB=0 ETH=1

s5 = 'Měření probíhá po dobu cca. 1 minuty';
disp(s5);

% mereni charakteristiky
s_port=OpenPort('COM4',USB_ETH);
% vybijeni kondenzatoru
PUM_DIO('W',0,0);

% mereni ADC
x_mereni=50*2;        % 10s...0.5x20
napeti = zeros(1,x_mereni);

% nabijeni kondenzatoru
PUM_DIO('W',0,1);
pause (15);

for i=1:x_mereni
napeti(i)=PUM_ADC(s_port,0);
pause (0.5)
end

ClosePort(s_port);

for i=1:x_mereni
%view
str=strcat('Namerene napeti je: ',num2str(napeti(i)),' V');
disp(str);
end

%nakresli vypocteny prubeh
[t,u]=ode45('RCdifer', [0:0.1:50], [0]);
plot (t, u,'g');
grid;
title ('průběh nabijení kondenzátoru');
xlabel ('čas [s]');
ylabel ('napětí [V]');
hold on;
%nakresli zmereny prubeh
cas=0:0.5:x_mereni/2-0.5;
plot(cas,napeti,'r');
legend('vypočtený','naměřený',4);
```

Řešení laboratorní úlohy č.3

Script:

```
% script pro ulohu c.3
% kalibrace cidla GP2D120, detekce na vzdalenost 4-30cm
% vyslednou charakteristiku zobrazi

USB_ETH=0;   %USB=0 ETH=1
x_mereni=27;

s_port=OpenPort('COM4',USB_ETH);
napeti = zeros(1,x_mereni);
rovnice = zeros(1,x_mereni);

s1 = 'Nastavte prekazku ve vzdalenosti';

for i=1:x_mereni
s2=num2str(i+3);
s3=strcat(s1,'_',s2,'cm');
disp(s3);

input('Stiskni klavesu ENTER');
napeti(i)=PUM_ADC(s_port,0);
end

for i=1:x_mereni
%view
s1 = 'Namerene napeti je: ';
%disp(s1);
%disp(napeti);
s2=num2str(napeti(i));
s3=strcat(s1,s2,' V');
disp(s3);
end

ClosePort(s_port);

s1 = 'Vypoctený polynom ma rovnici d=13.268/(U+0.029)-0.7';
disp(s1);

delka=4:1:x_mereni+3;

rovnice=13./(delka.+0.7).-0.029;
plot(delka,napeti,delka,rovnice);
grid;
title ('Kalibrace cidla GP2D120');
xlabel ('vzdalenost prekazky');
ylabel ('napětí');
axis([4,30,0,3.5]);
```

Řešení laboratorní úlohy č.4

Script:

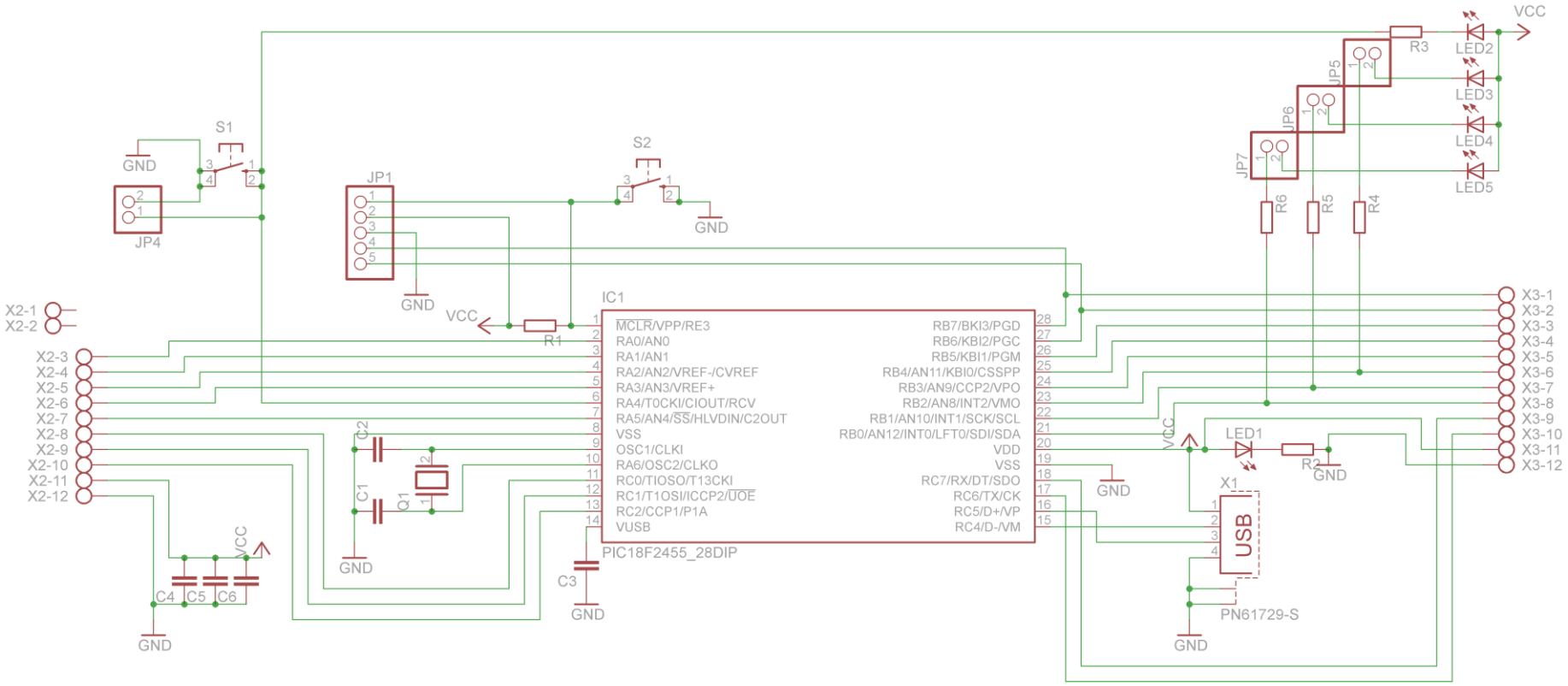
```
% script pro ulohu c.4

USB_ETH=0;  %USB=0 ETH=1
x_count=11; %pocet natoceni
uhel=0;

s_port=OpenPort('COM4',USB_ETH);

for i=1:x_count
% vypocet hodnoty pro natoceni
hodnota=round(uhel*1023/300);
uhel=uhel+30;
PUM_AX12(s_port,'U',hodnota);

input('Stiskni klavesu ENTER');
end

ClosePort(s_port);
```

Řešení laboratorní úlohy č.5

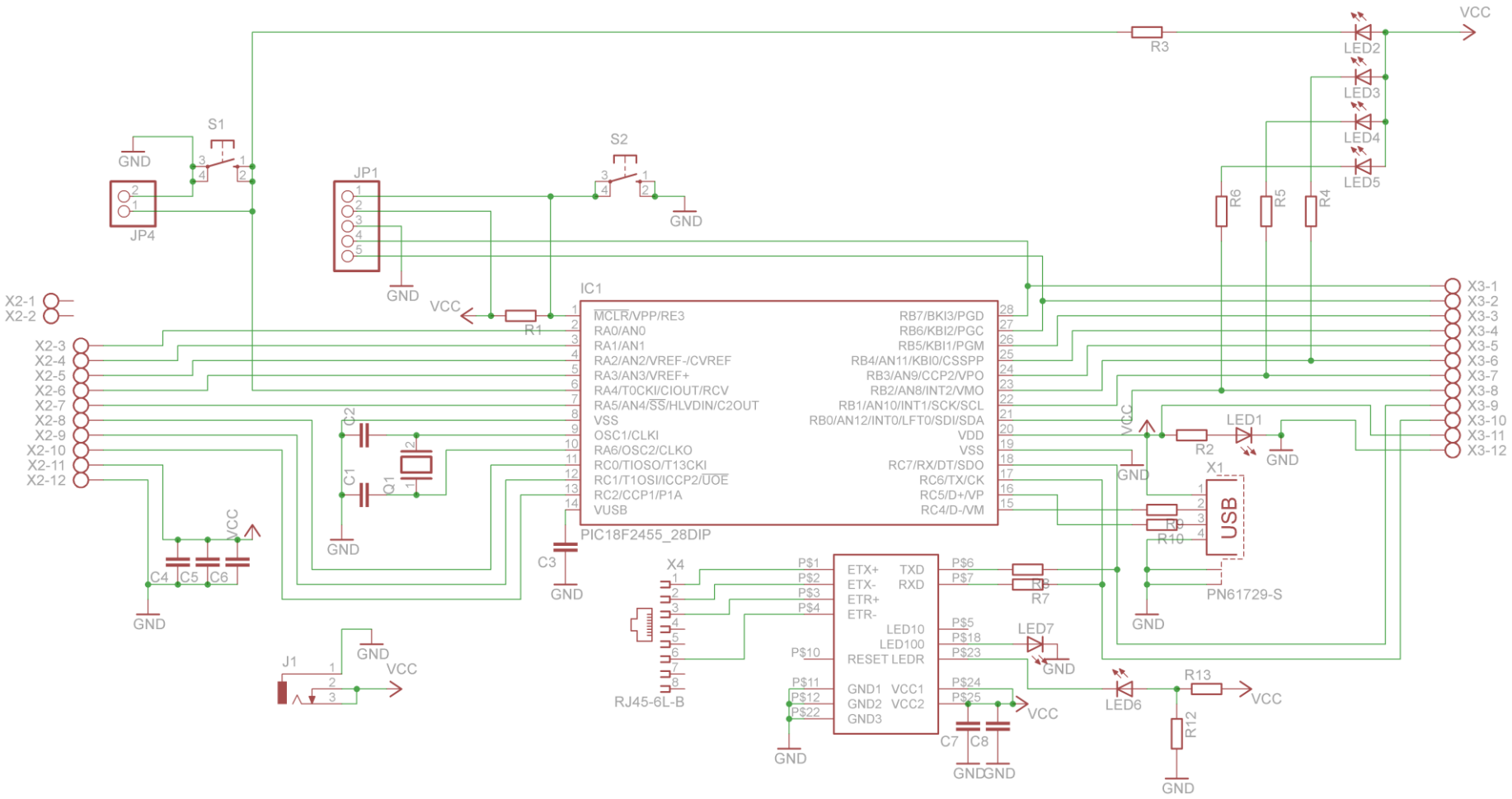
Script:

```
% script pro ulohu c.5

USB_ETH=0;  %USB=0 ETH=1
x_count=9;
uhel=0;

s_port=OpenPort('COM4',USB_ETH);
napeti = zeros(1,x_count);
delka = zeros(1,x_count);

for i=1:x_count

% vypocet hodnoty pro natoceni
hodnota=round(uhel*1023/300)
PUM_AX12(s_port,'U',hodnota);
pause (15);
uhel=uhel+30;

napeti(i)=PUM_ADC(s_port,0);
delka(i)=13.268/(napeti(i)+0.029)-0.7
end

ClosePort(s_port);

uhel=0:30:x_mereni+3;

plot(uhel,delka);
grid;
title ('Snimany prostor');
xlabel ('uhel');
ylabel ('vzdalenost prekazky');
axis([4,30,0,3.5]);

figure(2);
hold on;
x0=delka(4)*cos(uhel);
x=X0.delka.*cos(uhel);
y=delka.*sin(uhel);
plot(x,y);
```

Příloha C – schémata modulů PUM


VCC
S1
S2
GND
JP1
JP4
JP5
JP6
JP7
R3
R4
R5
R6
LED2
LED3
LED4
LED5
IC1
R1
C1
C2
C3
C4
C5
C6
Q1
X1
USB
LED1
R2
PN61729-S
PIC18F2455_28DIP
MCLR/VPP/RE3
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/CIOUT/RCV
RA5/AN4/SS/HLVDIN/C2OUT
VSS
OSC1/CLKI
RA6/OSC2/CLKO
RC0/T1OSO/T13CKI
RC1/T1OSI/ICCP2/UOE
RC2/CCP1/P1A
VUSB
RB7/BKI3/PGD
RB6/KBI2/PGC
RB5/KBI1/PGM
RB4/AN11/KBI0/CSSPP
RB3/AN9/CCP2/VPO
RB2/AN8/INT2/VMO
RB1/AN10/INT1/SCK/SCL
RB0/AN12/INT0/LFT0/SDI/SDA
VDD
VSS
RC7/RX/DT/SDO
RC6/TX/CK
RC5/D+/VP
RC4/D-/VM
X2-1
X2-2
X2-3
X2-4
X2-5
X2-6
X2-7
X2-8
X2-9
X2-10
X2-11
X2-12
X3-1
X3-2
X3-3
X3-4
X3-5
X3-6
X3-7
X3-8
X3-9
X3-10
X3-11
X3-12

PUM-U


VCC
R3
LED2
LED3
LED4
LED5
S1
S2
GND
JP1
JP4
R6
R5
R4
IC1
X2-1
X2-2
X2-3
X2-4
X2-5
X2-6
X2-7
X2-8
X2-9
X2-10
X2-11
X2-12
MCLR/VPP/RE3
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/CIOUT/RCV
RA5/AN4/SS/HLVDIN/C2OUT
VSS
OSC1/CLKI
RA6/OSC2/CLKO
RC0/T1OSO/T13CKI
RC1/T1OSI/ICCP2/UOE
RC2/CCP1/P1A
VUSB
RB7/BKI3/PGD
RB6/KBI2/PGC
RB5/KBI1/PGM
RB4/AN11/KBI0/CSSPP
RB3/AN9/CCP2/VPO
RB2/AN8/INT2/VMO
RB1/AN10/INT1/SCK/SCL
RB0/AN12/INT0/LFT0/SDI/SDA
VDD
VSS
RC7/RX/DT/SDO
RC6/TX/CK
RC5/D+/VP
RC4/D-/VM
PIC18F2455_28DIP
R1
C1
C2
Q1
C3
C4
C5
C6
X3-1
X3-2
X3-3
X3-4
X3-5
X3-6
X3-7
X3-8
X3-9
X3-10
X3-11
X3-12
LED1
R2
X1
USB
R9
R10
PN61729-S
X4
RJ45-6L-B
P$1
P$2
P$3
P$4
P$10
P$11
P$12
P$22
ETX+
ETX-
ETR+
ETR-
TXD
RXD
LED10
LED100
RESET LEDR
GND1
GND2
GND3
VCC1
VCC2
P$6
P$7
P$5
P$18
P$23
P$24
P$25
R8
R7
LED7
LED6
R13
R12
C7
C8
J1

PUM-E