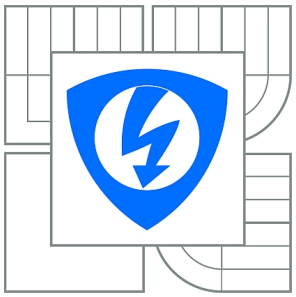
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

REGULACE VÝŠKY LÉTAJÍCÍHO ROBOTA

ALTITUDE CONTROL OF FLYING ROBOT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

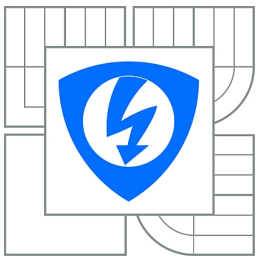
AUTOR PRÁCE — TOMÁŠ PALACKÝ

AUTHOR

VEDOUCÍ PRÁCE — Mgr. RADEK BARÁNEK

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Tomáš Palacký	***ID:***	125578
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Regulace výšky létajícího robota

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s vícerotorovými létajícími stroji známými jako koptery a s možnostmi regulace výšky těchto strojů. S pomocí přípravku vytvořeného v rámci semestrálního projektu navrhněte a experimentálně vyzkoušejte různé způsoby regulace výšky kopteru.

DOPORUČENÁ LITERATURA:

Mann Burkhard - C pro mikrokontroléry. BEN - technická literatura, 2003.
VÁŇA, Vladimír. Mikrokontroléry ATMEL AVR: programování v jazyce C. BEN - technická literatura, 2003
BLAHA, Petr. Stavový regulátor – stavová zpětná vazba, rekonstruktor stavu. Přednášky MTDS, VUT Brno 2005
BLAHA, P., VAVŘÍN, P. Řízení a regulace 1. VUT Brno: 2005. s. 54 (s.)
ŠOLC, F.; VÁCLAVEK, P.; VAVŘÍN, P. Řízení a regulace II. Brno: VUT, 2004. s. 1 (s.)

Termín zadání:	6.2.2012	***Termín odevzdání:***	28.5.2012

Vedoucí práce: Mgr. Radek Baránek
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce se v prví části zabývá návrhem a konstrukcí přípravku pro testování algoritmů regulace výšky létajícího robota. Dále pak vytvářením a testováním algoritmu regulace výšky za účelem aplikování poznatků při tvorbě řídicího systému kopteru. Je uveden výběr a rozbor použitých hardwarových a softwarových prostředků – mikrokontroleru AVR ATmega, ultrazvukového snímače, regulátoru motoru a také využívaných komunikačních rozhraní I²C, USART a SPI. Je zde popsán návrh a realizace elektrického zapojení, plošného spoje a mechanické konstrukce přípravku. Dále je obsahem tvorba programového vybavení mikrokontroleru a aplikace pro PC. S pomocí simulace je navržen regulační algoritmus pro regulaci výšky. Tento je testován a odladěn pro požadované vlastnosti.

KLÍČOVÁ SLOVA

AVR, ATmega, I²C, USART, ultrazvukový snímač, regulace výšky, SFR08, BLDC motor, Dual Bl-Ctrl, C, C# programování, simulace, PID regulátor

ABSTRACT

This bachelor's thesis deals firstly with design and construction of a device for testing of altitude control algorithms of a flying robot. Secondly with creating and testing of an altitude control algorithm in order to apply findings while creating a copter control system. Provided is a selection and analysis of used hardware and software components – microcontroller AVR ATmega, ultrasonic sensor, motor controller and also communication interfaces I²C, USART and SPI. Also design and realization of electrical wiring, circuit board and mechanical structure is being described here. The next section includes creation of software for microcontroller and PC application. The altitude control algorithm is developed using a simulation, furthermore tested and tuned for desired behavior.

KEYWORDS

AVR, ATmega, I²C, USART, ultrasonic sensor, height regulation, SFR08, BLDC motor, Dual Bl-Ctrl, C, C# programming, simulation, PID controller

BIBLIOGRAFICKÁ CITACE:

PALACKÝ, T. Regulace výšky létajícího robota. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav automatizace a měřicí techniky, 2012. 50 s. Vedoucí bakalářské práce byl Mgr. Radek Baránek.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Regulace výšky létajícího robota jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 28. května 2012 .....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Mgr. Radku Baránkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 28. května 2012 .....

podpis autora

OBSAH

Seznam obrázků **9**

Seznam tabulek **10**

Úvod **11**

1 Koptery, regulace výšky **12**

2 Výběr komponent přípravku **13**

2.1 Mikrokontroler ... 13
2.1.1 Mikrokontrolery AVR ... 14
2.1.2 Mikrokontroler ATmega32A ... 14
2.2 USB – USART převodník ... 17
2.3 Snímač vzdálenosti ... 17
2.3.1 Ultrazvukový dálkoměr SFR08 ... 18
2.4 Pohon vrtule ... 20
2.4.1 Motor ... 20
2.4.2 Regulátor Dual BL-Ctrl ... 21

3 Využívaná komunikační rozhraní **22**

3.1 Sériové rozhraní SPI ... 22
3.2 Sběrnice I²C ... 23
3.3 Sériové rozhraní USART ... 24

4 Realizace přípravku **25**

4.1 Blokové schéma ... 25
4.2 Elektrické schéma a DPS ... 26
4.3 Fyzická realizace ... 27

5 Realizace softwaru **28**

5.1 Software mikrokontroleru ... 28
5.2 Software pro PC ... 29

6 Regulace – simulace, testování **33**

6.1 PID regulátor ... 33
6.2 PSD regulátor ... 34

6.3 Simulace 36
6.4 Ladění regulátoru 38

7 Závěr **40**

Literatura **41**

Seznam symbolů, veličin a zkratek **43**

Seznam příloh **44**

SEZNAM OBRÁZKŮ

Obr. 1.1: Principiální znázornění kopteru 12

Obr. 1.2: Reálný quadro-kopter 12

Obr. 2.1: ATmega32 - zapojení vývodů pouzdra TQFP44 (převzato z [4]) 15

Obr. 2.2: Blokový diagram mikrokontroleru ATmega32A (převzato z [4]) 16

Obr. 2.3: Schéma USB – USART převodníku 17

Obr. 2.4: Snímač SFR08 18

Obr. 2.5: BLDC motor RAY C1826/31 20

Obr. 2.6: Regulátor Dual BL-Ctrl 21

Obr. 3.1: Typické zapojení sběrnice SPI (převzato z [8]) 22

Obr. 3.2: Příklad připojení několika zařízení na sběrnici I^2C (převzato z [6]) 23

Obr. 3.3: Časový průběh logických úrovní sběrnice I^2C (převzato z [6]) 24

Obr. 4.1: Blokové schéma přípravku 25

Obr. 4.2: Schéma zapojení mikroprocesoru 26

Obr. 4.3: Schéma DPS – napájecí část 27

Obr. 4.4: Hardwarový přípravek při levitaci 27

Obr. 5.1: Vývojový diagram programu mikroprocesoru 28

Obr. 5.2: Ovládací prvky programu 30

Obr. 6.1: Blokové schéma regulace v uzavřené smyčce 33

Obr. 6.2: PID regulátor - odezva na skokovou změnu žádané veličiny 34

Obr. 6.3: Regulační smyčka s PSD regulátorem 36

Obr. 6.4: Graf závislosti tahu motoru na řídicím slově 37

Obr. 6.5: Model regulované soustavy 38

Obr. 6.6: Průběh regulace výšky 1 38

Obr. 6.7: Průběh regulace výšky 2 39

SEZNAM TABULEK

Tab. 2.1: Registry snímače SFR08 19

ÚVOD

Cílem této práce je vyvinout hardware a software pro testování regulace výšky, implementovat a otestovat regulační algoritmy, které naleznou uplatnění při implementaci do systémů vícemotorových létajících robotů typu quadro-kopter, hexa-kopter apod. Tyto typy strojů nejsou běžně zastoupeny v reálném civilním provozu, ale pro jejich jednoduchost konstrukce a relativně jednoduché řízení jsou velice rozšířené na poli experimentálních zařízení a mobilní robotiky.

Z praktického a bezpečnostního hlediska není vhodné provádět tyto experimenty na samostatně se vznášejících strojích. Pro testování a ladění stabilizace výškové či směrové se je většinou zapotřebí vytvořit složité matematické modely a s pomocí simulačního softwaru jako je například MATLAB/Simulink udělat vizualizaci. Ani takto není dostatečně kvalitní výsledek zaručen, protože nelinearity, pronikající rušení a jiné fyzikální vlastnosti se často nedají popsat s dostatečnou přesností a je těžké je vystihnout.

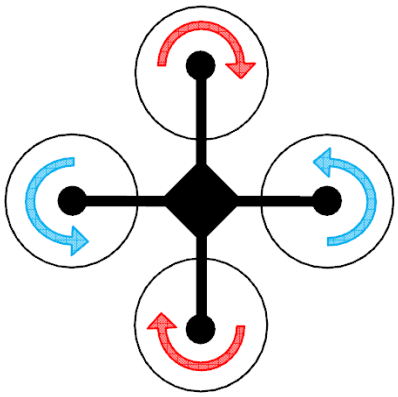
S pomocí tohoto přípravku, bude možné provádět experimenty s regulací výšky na reálném zařízení bez nutnosti složitých výpočtů a simulací. Samotné vnášející se zařízení bude mít mechanicky vymezený rozsah pohybu, aby se předešlo nehodám při experimentech. Nejedná se tedy o robot vznášející se volně v prostoru. Zařízení bude propojeno s počítačem a na počítači běžící aplikací se bude ovládat. Tato aplikace umožní nastavovat konstanty implementovaného regulátoru, interaktivně měnit požadovanou výšku letu, vykreslovat grafy reálných průběhů a zjednoduší tak vyhodnocování kvality regulace. Dosažených výsledků bude poté možno využít při tvorbě systému řízení kopteru.

1 KOPTERY, REGULACE VÝŠKY

Koptery jsou vícerotorové létající stroje či roboty, principielně podobné stavbě vrtulníku. Typicky mají čtyři nebo šest nezávisle řiditelných rotorů se svislou osou rotace. Podle počtu těchto rotorů se označují jako quadro-koptery (quad-rotory), hexa-koptery apod. Výhoda těchto kopterů oproti klasickým jednomotorovým vrtulníkům je vysoká schopnost manévrovat, konstrukční jednoduchost, jejich odolnost a spolehlivost daná na rozdíl od klasických vrtulníků absencí pohyblivých mechanických částí.

Na obrázku 1.1 je principielně znázorněna konstrukce quadro-kopteru. Základ tvoří 4 rotory, tedy elektromotory s připevněnými vrtulemi umístěné v jedné rovině na čtyřech ramenech. Tyto ramena dohromady tvoří jakýsi kříž, do jehož středu se umisťuje řídicí elektronika a napájecí baterie. Vždy dvě protilehlé vrtule se otáčí ve stejném směru, který je opačný než směr dalších dvou. Tímto se vyruší nežádoucí reakční moment, což je největším problémem obyčejných vrtulníků.

Veškeré řízení kopterů, tedy náklony, zatáčení, stoupání atd. se děje změnou rychlostí vrtulí. Toto obstarává většinou řídicí mikroprocesor na základě vzdálených podnětů obsluhy. Aby bylo možné autonomně takovýto robot řídit, je často vybaven nejrůznějšími snímači neelektrických veličin. Jsou to především ultrazvukové snímače vzdálenosti pro výšku a pro stabilizaci gyroskopy a akcelerometry, které případně doplňují senzory tlaku, nebo magnetometry. Na obrázku 1.2 je reálný quadro-kopter i s veškerou řídicí elektronikou.



Obr. 1.1: Principiální znázornění kopteru

Obr. 1.2: Reálný quadro-kopter

Pro zpracování dat ze senzorů, regulaci pohonů a řízení dle pokynů obsluhy přijatých rádiem je třeba v řídicím mikroprocesoru implementovat sofistikovaný řídicí systém. Základem správně fungujícího kopteru je dobrá regulace výšky ať už při statické levitaci, či při pohybu. Právě touto problematikou se zabývá tato práce.

2 VÝBĚR KOMPONENT PŘÍPRAVKU

Hardware přípravku bude představovat jakýsi improvizovaný mechanicky upevněný létající robot. Mechanická část musí umožňovat volný pohyb ve vertikálním směru do výšky zhruba 75 centimetrů s vymezením dovoleného rozsahu pohybu. Pro regulaci je zásadně důležité měřit výšku modulu nad podložkou vhodným snímačem vzdálenosti. Vztlak bude zajišťovat jedna vrtule poháněná elektrickým motorem podobně, jak se využívá v systémech kopterů. Pro vyhodnocování kvality regulace a například nastavování parametrů regulátoru bude potřeba implementovat komunikační protokol umožňující připojení k PC. Programové řízení celého přípravku a veškerou komunikaci bude zajišťovat vhodný mikrokontroler. Napájení bude třeba dimenzovat s ohledem na proudovou náročnost použitého elektromotoru.

2.1 Mikrokontroler

Mikrokontrolerů pro aplikace, jako je tato existuje na trhu celá řada. Především jsou to 8-bitové mikrokontrolery typu i8081, PIC, AVR, HCS12 a 32-bitové ARM a AVR. Tyto typy využívají odlišné soubory instrukcí. Vyrábí se nepřeberné množství různých modifikací podle velikosti vnitřních pamětí (FLASH, SDRAM, EEPROM), počtu vstupně/výstupních portů, komunikačních rozhraní a dalších podpůrných obvodů obsažených na čipu.

Při výběru mikrokontroleru pro tento přípravek byly zásadní tyto požadavky:

- Implementace rozhraní I^2C (komunikace s SFR08 a BL-Ctrl)
- Možnost komunikace s PC
- Snadné programování v aplikaci
- Dostatečně velká paměť programu
- Malé rozměry pouzdra

Vzhledem k principu snímání výšky ultrazvukovým snímačem s periodou vzorkování v řádu několika milisekund, nebyly nároky na výpočetní výkon nijak velké. Při výběru také hrála roli znalost dané architektury a programovacího jazyka.

2.1.1 Mikrokontrolery AVR

Jádro AVR [3] je typu RISC (Reduced Instruction Set Computer). Skládá se ze 32 stejných 8bitových registrů, které mohou obsahovat jak data, tak i adresy. Vzhledem k propojení registrů s ALU (Arithmetic Logic Unit) provede ALU za jeden hodinový cyklus jednu operaci. Mikrokontrolery AVR využívají koncepci Harwardské architektury. To znamená, že mají oddělenou paměť pro program a pro data.

Mikrokontrolery AVR je možné programovat jak paralelně, tak i sériově a to přímo v systému. Při paralelním programování se využívá toho, že obvod je navržen tak, aby po připojení programovacího napětí na určitý pin obvodu bylo provedeno přepnutí vývodů z normálního režimu I/O portů na adresové a datové vývody vnitřní paměti. Pak je možné do paměti paralelně zaznamenat data. Po naprogramování se obvod opět přepne zpět. Nevýhoda tohoto programování je, že je nutné mikroprocesor odpojit od jakýchkoliv obvodů a umístit ho do programátoru. Tato nevýhoda odpadá při sériovém programování. Při tomto programování mikroprocesor zůstává v aplikaci a pomocí signálů MOSI, MISO, SCLK a RESET připojených na programátor ho lze jednoduše naprogramovat. Tomuto programování se říká ISP (In System Programming). Většina mikrokontrolerů z řady ATmega obsahuje ještě JTAG rozhraní. To je pro ladění softwaru přímo v aplikaci.

Mikrokontrolery se liší také použitými periferiemi, které jsou obsaženy na čipu mikrokontroleru. Je to například velikost paměti SDRAM, FLASH a EEPROM. Dále počtem čítačů/časovačů a jejich rozlišením (8 nebo 16 bit), počtem portů. Většinou obsahují USART. Některé procesory řady Mega jich mají i více. Obsahují analogový komparátor, obvod Watchdog a některé i A/D převodník a spousty dalších obvodů.

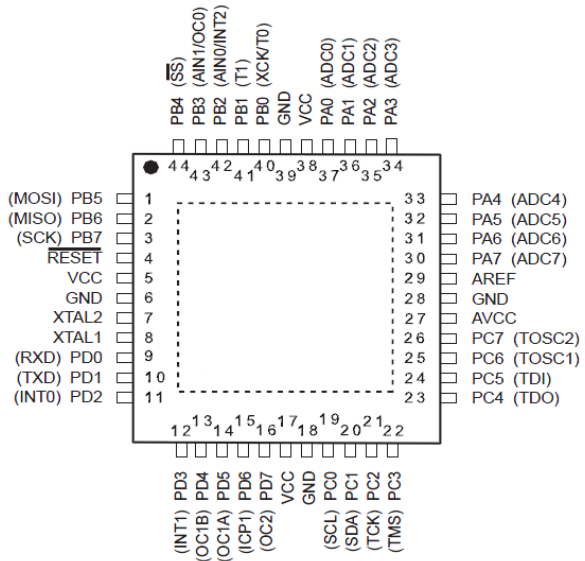
2.1.2 Mikrokontroler ATmega32A

Jedním z mnoha zástupců architektury AVR řady Mega je mikrokontroler ATmega32A [4]. Zde je uveden výčet jeho důležitých parametrů:

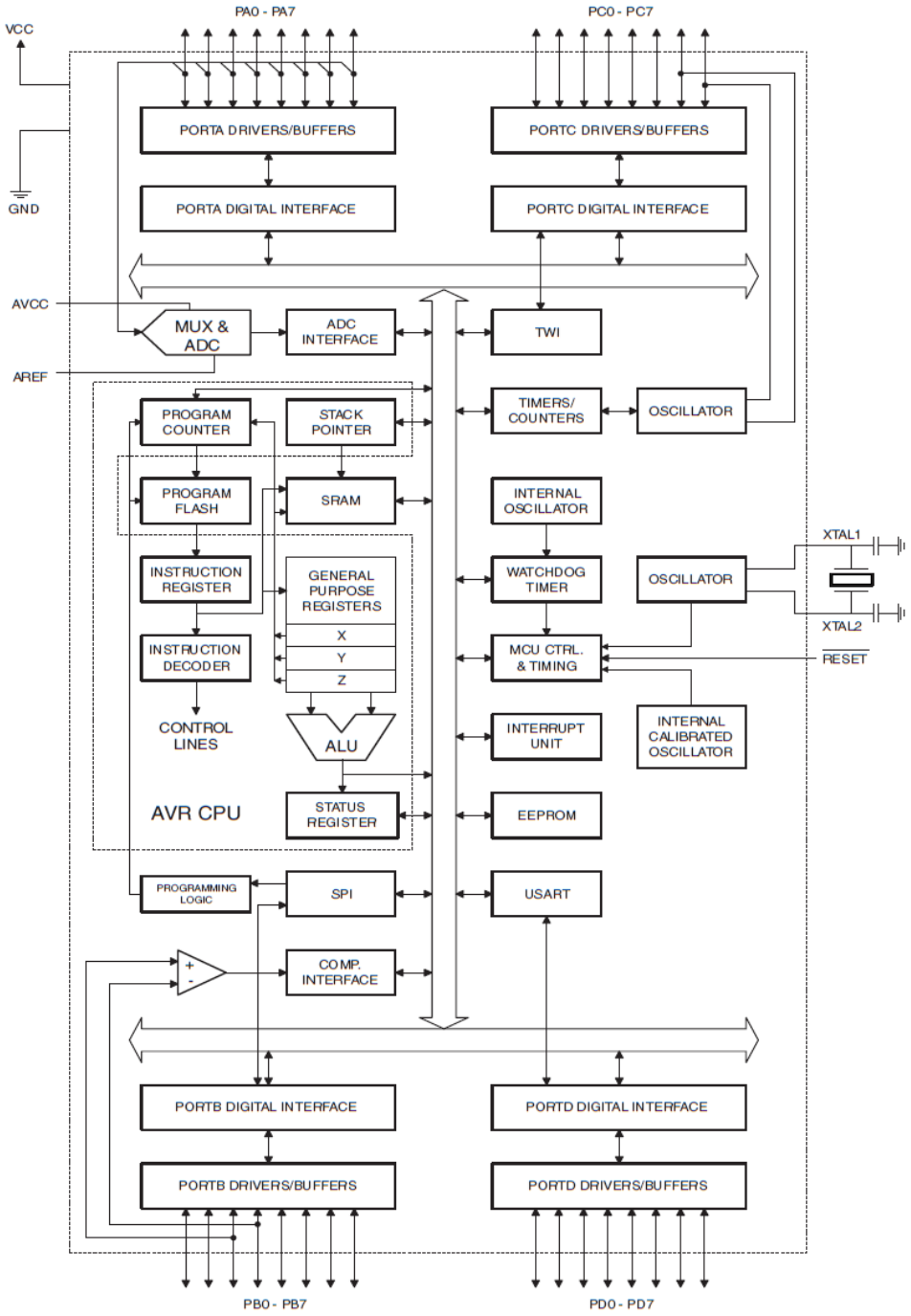
- Napájecí napětí 2,7 V až 5,5 V
- Hodinový kmitočet až 16 MHz
- Instrukční soubor obsahuje 131 instrukcí
- Výpočetní výkon až 16 MIPS při 16 MHz
- 32 registrů o šířce 8 bitů
- 4 vstupně/výstupní porty o šířce 8 bitů
- 32 KB Flash paměti programu
- 1024 B datové paměti EEPROM

- 2 KB datové paměti SRAM
- Paměti Flash, EEPROM a Lock, Fuse bity programovatelné prostřednictvím SPI nebo JTAG
- Dva 8-bitové časovače/čítače, jeden 16-bitový časovač/čítač
- Čtyři PWM kanály
- Osmikanálový 10-bitový A/D převodník
- Jednotky USART, SPI, TWI (I²C)
- Vnitřní RC oscilátor, obvod Watchdog, analogový komparátor

Jako diskrétní součástka se vyrábí ve dvou provedeních podle typu pouzdra: PDIP 40 a TQFP/MLF 44 (viz obr. 2.1). Blokové schéma vnitřní struktury tohoto mikrokontroleru je na obr. 2.2.



Obr. 2.1: ATmega32 - zapojení vývodů pouzdra TQFP44 (převzato z [4])

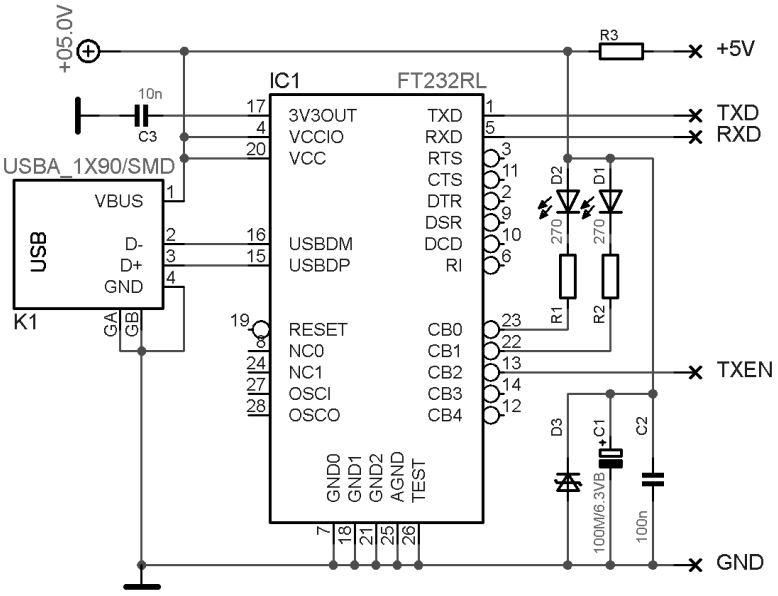


Obr. 2.2: Blokový diagram mikrokontroleru ATmega32A (převzato z [4])

Tento mikroprocesor byl pro realizovaný přípravek zvolen především pro jeho velkou paměť programu FLASH o velikosti 32KB, což do budoucna umožní implementovat i velice složité algoritmy. Vyhovuje také ostatním výše uvedeným požadavkům (dostatečný výpočetní výkon, hardwarová implementace rozhraní I^2C, USART, ISP programování).

2.2 USB – USART převodník

Použitý Převodník komunikačních rozhraní USB a USART, jejichž vlastnosti jsou popsány v kapitole 3, je kompaktní modul realizovaný na oboustranné desce plošných spojů s konektorem USB 1.x/2.0 Standard A. Připojuje se tedy přímo do USB portu Počítače. Funkci převodníku obstarává integrovaný obvod FT232RL. Schéma převodníku je na obrázku 2.6. S nainstalovanými FTDI ovladači se jeví jako virtuální hardware sériového portu COM. Z hlediska software na PC se tedy ke komunikaci skrze tento převodník přistupuje jako při běžné komunikaci po sériovém portu. FTDI ovladače jsou obsahem přílohy č. 2 na DVD.



Obr. 2.3: Schéma USB – USART převodníku

2.3 Snímač vzdálenosti

Pro bezkontaktní snímání polohy (vzdálenosti) se běžně používají dva základní typy senzorů: laserové a ultrazvukové. Ultrazvukové sonary mají na rozdíl od laserů širokou směrovou charakteristiku. Obě skupiny jsou založeny na principu vyslání signálu do prostoru a následného vyhodnocování jeho odrazu. TOF (Time Of Flight) snímače vypočítávají vzdálenost objektu z času, za který se odražený signál vrátil do přijímače. Laserové TOF senzory vynikají svou přesností, velkým měřitelným rozsahem a rychlostí měření. Nejsou závislé na teplotě prostředí, ani na úhlu natočení předmětu. Tyto výhody mají i laserové senzory vyhodnocující vzdálenost podle rozdílu fáze vysílaného a přijímaného paprsku, či podle jeho úhlu dopadu s využitím principu triangulace. S rozlišením a přesností jsou na tom ještě lépe.

Ultrazvukové TOF sonary s piezoelektrickými měniči využívají signál o frekvenci okolo 40kHz, tedy zvuk, jehož rychlost šíření ve vzduchu závisí na teplotě (viz.(2.1)). Některé snímače proto mají zabudovanou teplotní korekci. Díky relativně nízké rychlosti šíření zvuku ve vzduchu dané vztahem

$$c = 331{,}57 + 0{,}607 \cdot \vartheta\,[\mathrm{m/s,°C}], \qquad (2.1)$$

kde ϑ je teplota vzduchu, je každé měření mnohokrát pomalejší (desítky ms) než u laseru. Tento typ snímačů je však konstrukčně jednoduchý a levný. Proto se hojně používá v robotice i v průmyslu tam, kde jeho parametry dostačují.

Další skupinou bezkontaktních snímačů vzdálenosti jsou IR senzory, ty využívají odrazu paprsku infračerveného světla většinou vysílaného v impulzech s frekvencí 38kHz. Jedná se o levné senzory se schopností detekce do 1 m. Pro přesné určování vzdálenosti nejsou vhodné, protože na ně působí parazitní vlivy, jako okolní osvětlení a odrazivost materiálu.

2.3.1 Ultrazvukový dálkoměr SFR08

Pro použití na přípravku byl vybrán snímač SFR08 [1]. Má dostatečný měřicí rozsah s jemným rozlišením a za nízkou cenu dostačující parametry. Výstup měřených dat probíhá po sériové sběrnici I²C, která je velmi často implementována v nejrůznějších mikroprocesorech.

Modul ultrazvukového snímače SFR08 zobrazený na obr. 1.1. je kompaktní zařízení s relativně malými nároky na napájení (5 V / 15 mA, 3 mA v nečinnosti). Základem jsou dva piezoelektrické měniče. Jeden pro vysílání, jeden pro příjem odraženého ultrazvukového signálu o frekvenci 40 kHz. Měřitelný rozsah vzdáleností je od 3 cm do 6 m. Mimo to modul zahrnuje také snímač osvětlení.

Obr. 2.4: Snímač SFR08

Funkci obstarává mikrokontroler PIC, který navenek obstarává komunikaci po sběrnici I^2C. Implicitně je zařízení přidělena adresa 0xE0, tu lze však měnit. Na sběrnici se jeví jako paměťový blok o 36 registrech. Zápisem do příkazového registru 0 se

nastavuje pracovní mód, jednotka výstupní čtené veličiny (cm, palce, milisekundy) a spouští se samotné měření. Je možné provozovat v módu ANN (Artificial Neural Network) pro implementaci v systému s umělou neuronovou sítí. Zápisem do registru 2 se nastavuje horní hranice měřicího rozsahu, tedy časový interval, po který se čeká na příjem echa. Tato hodnota rozsahu je dána vztahem

$$\text{Rozsah} = 43 + (43 \cdot \text{registr2}) \ [\text{mm}]. \qquad (2.2)$$

Implicitně (po připojení napájení) je v registru 2 hodnota 255, což dává rozsah 11m a dobu měření 65ms. V případě potřeby měřit jen kratší vzdálenosti je vhodné rozsah snížit a zkrátit tak dobu měření, po kterou je nutné čekat na výsledek. Je poté možné zjišťovat naměřenou vzdálenost v kratších intervalech. V tomto případě je třeba snížit hodnotu maximálního analogového zesílení v registru 1, čímž se eliminuje šance zachycení odrazu z předchozího měření a tím pádem nesprávného výsledku. Na adrese 1 lze číst hodnotu světelného snímače sejmutou A/D převodníkem v okamžiku vysílání zvuku. Na adrese 2 a 3 je zapsána naměřená hodnota prvního odrazu ve zvolených jednotkách: centimetrech, palcích nebo milisekundách představujících čas mezi vysláním a příjmem signálu. Toto číslo je rozděleno na vyšší a nižší byte. Dalších 16 odrazů je stejným způsobem zapsáno postupně až po adresu 35. Paměťový adresní prostor snímače přístupný na I²C sběrnici znázorňuje tabulka 1.1.

Tab. 2.1: Registry snímače SFR08

Lokace	**Čtení**	**Zápis**
0	Verze firmware	Příkazový registr
1	Světelný senzor	Max. zesílení (implicitně 31)
2	1. echo – vyšší byte	Max. rozsah (implicitně 255)
3	1. echo – nižší byte	-
~	~	~
34	17. echo – vyšší byte	-
35	17. echo – nižší byte	-

2.4 Pohon vrtule

2.4.1 Motor

DC motor

Stejnosměrný (DC) motor je jednoduchý typ elektrického motoru s vinutím na rotoru. Je schopen dosahovat vysokých otáček, které se snadno řídí hodnotou napájecího napětí nebo PWM modulací. Velkou nevýhodou je komutátor spínající proudy do vinutí za přítomnosti jiskření, což přináší problémy s rušením a spolehlivostí.

BLDC motor

Synchronní bezkomutátorový motor BLDC (Brushless DC) je z hlediska mechanického uspořádání nejjednodušší. Vinutí motoru jsou součástí statoru, rotor je tvořen permanentními magnety. Komutace, tedy spínání jednotlivých vinutí musí být zajištěna vhodným budičem. Tento typ motoru se často používá v modelářské technice a systémech copterů. Výhodou je vysoká životnost a účinnost. Cena je však vyšší.

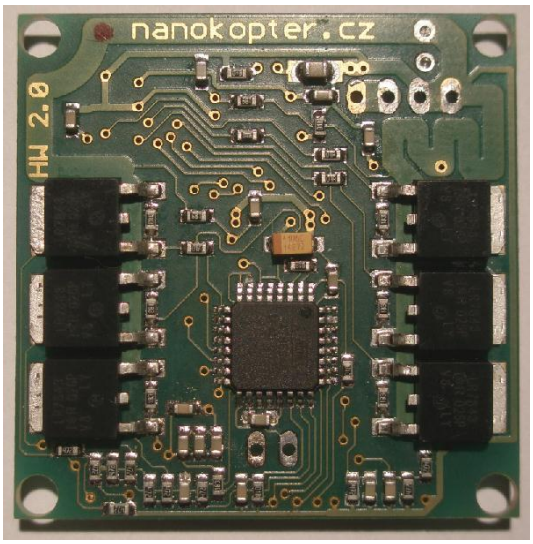
Pro pohon vrtule a zajištění potřebného vztlaku, který udrží zařízení ve vzduchu byl vybrán třífázový BLDC motor s označením RAY C1826/31 (viz. obr. 1.2) s jmenovitými otáčkami 1800 ot/min/V, výkonem 55 W, maximální zatížitelností 10 A a hmotností 18 g.

Obr. 2.5: BLDC motor RAY C1826/31

K motoru je gumovým o-kroužkem připevněna dvoulistá vrtule GWS EP-6030 s průměrem 152 mm a stoupáním 76 mm.

2.4.2 Regulátor Dual BL-Ctrl

Pro napájení a regulaci otáček motoru byl zvolen regulátor Dual BL-Ctrl (Brushless Controller) z konstrukce Nanokopter, která vychází z německého projektu MikroKopter [2]. Oproti původnímu BL-Ctrl obsahuje na desce o rozměrech 40 x 40 mm dva regulátory (viz obr. 1.3). Výkonovou část pro každou ze tří fází motoru tvoří dvojice MOSFET tranzistorů schopných dodávat kontinuálně proud 10 A, výkon 55 W. Modul mimo jiné obsahuje kondenzátory pro odstranění rušení a informační LED diody pro signalizaci správné funkce a chybového stavu. Chod tohoto frekvenčního měniče/regulátoru řídí mikroprocesory ATmega8 navenek komunikující po I²C sběrnici. Každý z měničů má pro komunikaci po této sběrnici firmwarově stanovenou vlastní adresu. Na tuto adresu se zápisem hodnoty s určitou konstantou nastavují požadované otáčky. Toto je třeba provádět alespoň zhruba každých 200 ms, jinak regulátor automaticky přejde do chybového stavu a motor zastaví. Čtením z dané adresy je možné zjišťovat hodnotu aktuálně odebíraného proudu.



Obr. 2.6: Regulátor Dual BL-Ctrl

3 VYUŽÍVANÁ KOMUNIKAČNÍ ROZHRANÍ

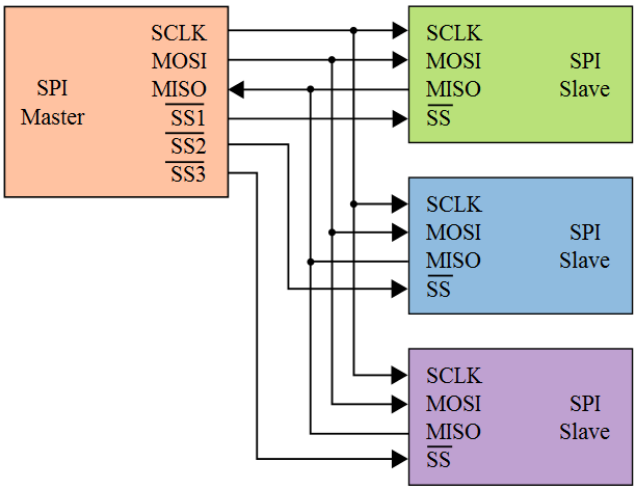
3.1 Sériové rozhraní SPI

SPI [8] (Serial Peripheral Interface) je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody (paměťmi EEPROM, A/D převodníky, displeji apod.). Komunikace je realizována pomocí společné sběrnice. K ní je možno připojit teoreticky libovolný počet obvodů. Zařízení na SPI sběrnici jsou rozdělena na:

Master - řídí komunikaci pomocí hodinového signálu a určuje, se kterým zařízením na sběrnici bude komunikovat pomocí SS (Slave Select) nebo CS (Chip Select).

Slave - vysílá podle hodinového signálu, pokud je aktivován pomocí SS/CS

Sběrnice je fyzicky realizována dvěma datovými vodiči MOSI (Master Output Slave Input), MISO (Master Input Slave Output) a vodičem pro hodinové impulzy SCLK (Synchronous Clock). V případě, že je připojeno více než jedno Slave zařízení, je třeba použít adresaci. Ta se provádí pomocí zvláštních vodičů připojených k pinům SS nebo CS (viz obr. 1.8). Zařízení je aktivováno signálem v log. 0. Délka vyslaných dat je buď jeden nebo dva bajty (8/16 bitů).

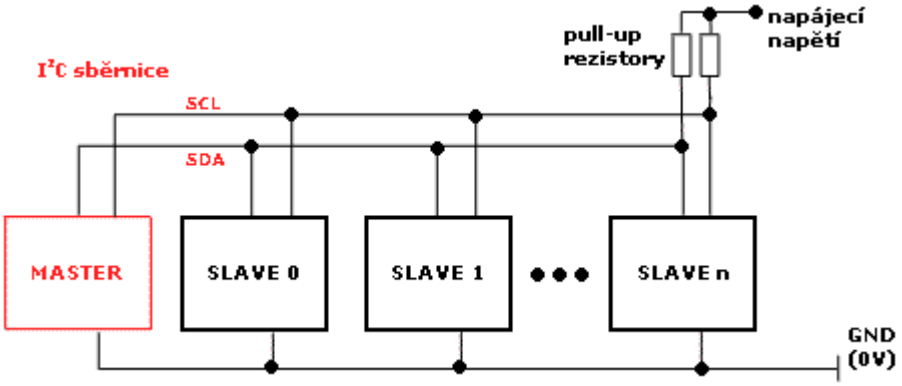


Obr. 3.1: Typické zapojení sběrnice SPI (převzato z [8])

V realizovaném přípravku je prostřednictvím rozhraní SPI zajištěno spojení mezi mikroprocesorem a ISP programátorem.

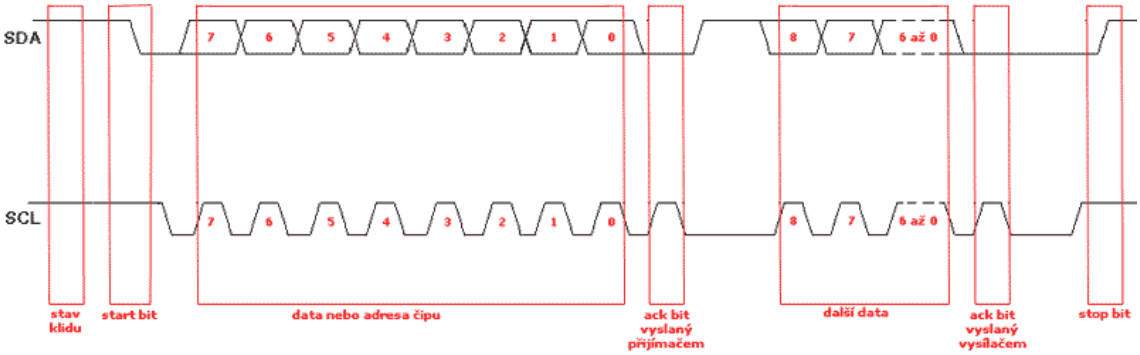
3.2 Sběrnice I²C

I²C [5][6] (Inter-Integrated Circuit), někdy také označováno TWI (Two Wire Interface), je sériová sběrnice pro komunikaci mezi na ni připojenými zařízeními. Připojení na tuto sběrnici podporuje spousta integrovaných obvodů: RAM a EEPROM paměti, budiče displejů, D/A převodníky a další speciální obvody pro audio a video techniku. Je fyzicky realizovaná dvěma vodiči: datový SDA a hodinový SCL. Na jednu sběrnici je možné připojit až 128, případně 1024 zařízení, podle toho, jestli je použito adres o délce 7 bitů nebo 10 bitů. Každé z připojených zařízení musí mít individuální adresu. Pokud jsou připojovány například dva integrované obvody s od výrobce nastavenou stejnou adresou, je třeba, pokud je to možné, tuto adresu předem u jednoho z nich změnit. Ukázka zapojení sběrnice je na obr. 1.6. Připojená zařízení se rozlišují na Master a Slave. Zařízení Master je většinou jen jedno, řídí komunikaci a generuje hodinový signál na SCL.



Obr. 3.2: Příklad připojení několika zařízení na sběrnici I²C (převzato z [6])

V klidovém stavu jsou zajištěny úrovně logické 1 připojenými pull-up rezistory. Každý přenos zahajuje Master vysláním start bitu úrovně 0 na SDA při SCL stále v úrovni 1. Se začátkem hodinových impulzů následuje vysílání sedmi nebo deseti bitů adresy příjemce spolu s R/W bitem, který indikuje požadovanou operaci - čtení/zápis dat. Další bit ACK vysílá přijímací zařízení. Úrovní 0 potvrzuje správný příjem a připravenost vysílat/přijímat data. Dále jsou po bajtech přenášena data ve směru určeném bitem R/W. Úroveň na SDA se může změnit jen, pokud SCL v úrovni 0. Každý poslaný byte je ze strany přijímače potvrzován bitem ACK. Po ukončení přenosu je vyslán stop bit a oba vodiče uvedeny do klidového stavu. Časový průběh komunikace na sběrnici s adresami o délce 7 bitů ukazuje obr. 1.7.



Obr. 3.3: Časový průběh logických úrovní sběrnice I²C (převzato z [6])

Frekvence signálu SCL je standardně 100 kHz, pro Fast Mode 400 kHz a High Speed Mode až 3.4 MHz. Sběrnice I²C neumožňuje duplexní přenos, v jednom okamžiku vysílá jen jedno zařízení. Každá ze stanic může zahájit vysílání, je-li předtím sběrnice v klidovém stavu. Používá se metoda s detekcí kolize Vysílač při vysílání kontroluje stav SDA a porovná s odeslanými bity. Je-li zjištěn rozdíl mezi očekávaným a skutečným stavem linky SDA, je indikována kolize.

V realizovaném přípravku po sběrnici I²C komunikuje řídicí mikrokontroler ATmega se snímačem výšky SFR08 a s frekvenčním měničem Dual BL-Ctrl.

3.3 Sériové rozhraní USART

USART [7] (Universal Synchronous / Asynchronous Receiver and Transmitter) je synchronní / asynchronní sériové rozhraní pro sériovou komunikaci, které lze nastavit buď pro asynchronní režim (SCI - např. pro linky RS232 resp. RS485), anebo pro synchronní režim (běžně označovaný jako SPI). USART vysílá data na pinu označovaném jako TXD (Transmit Data), přijímá na pinu RXD (Receive Data). Jako jeden ze sériových I/O modulů je USART implementován ve spoustě mikrokontrolerů a integrovaných obvodů. Je určen ke komunikaci mezi dvěmi zařízeními, odpadá tedy problematika adresování. Fyzicky se spojení realizuje propojením TXD pinu prvního s RXD pinem druhého zařízení. V případě využití obousměrné komunikace je třeba spojit i RXD pin prvního s TXD pinem druhého zařízení.

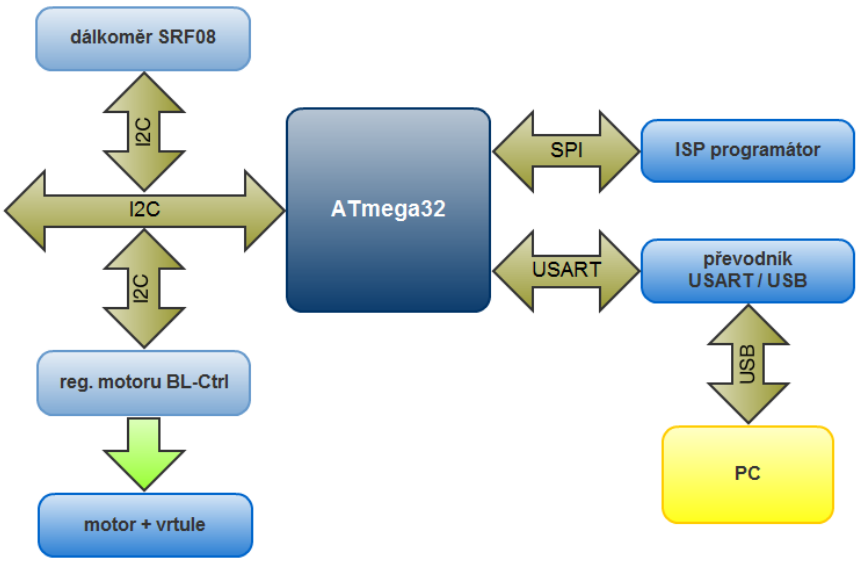
Přenos dat probíhá po bajtech. Klidová úroveň signálu je log. 1. Vysílání je zahájeno start bitem - hodnotou signálu log. 0 po dobu jednoho bitu. Následně se posílají datové bity od nejnižšího po nejvyšší. Poté následuje stop bit, který má úroveň log. 1. Po odvysílání stop bitu může začít přenos dalšího bajtu.

V realizovaném přípravku prostřednictvím rozhraní USART a převodníku USART/USB komunikuje řídicí mikrokontroler ATmega s PC.

4 REALIZACE PŘÍPRAVKU

4.1 Blokové schéma

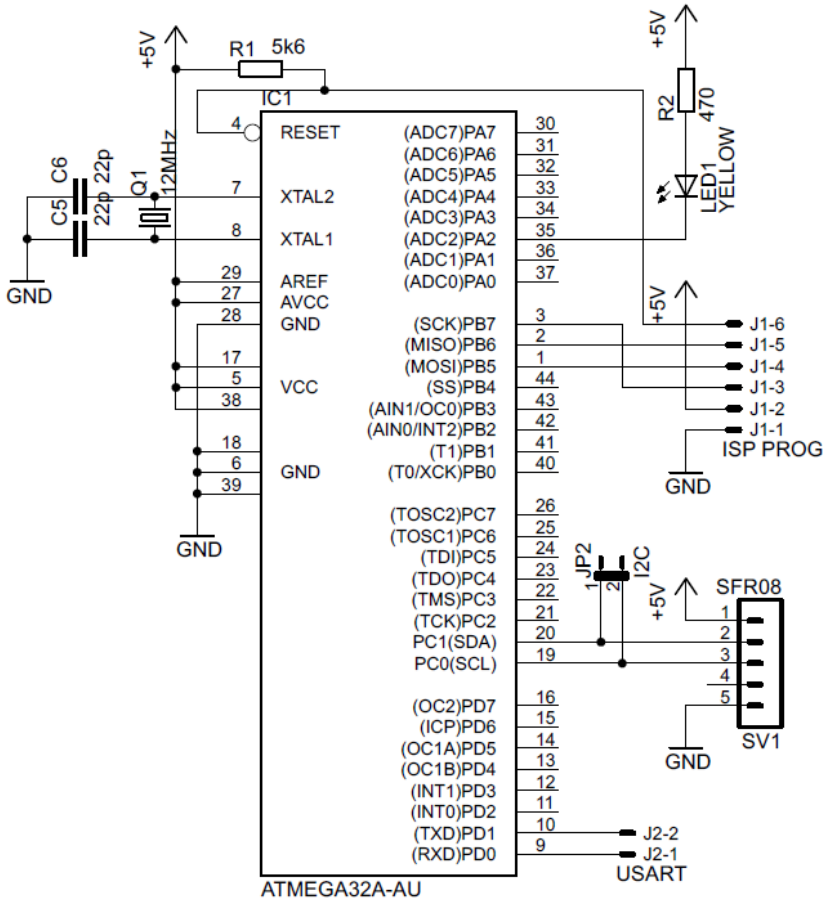
Po ujasnění, které komponenty budou na přípravku použity, bylo sestaveno blokové schéma propojení jednotlivých součástí. Zobrazeno je na obr. 2.1. Jakýmsi srdcem přípravku je mikrokontroler ATmega32. Frekvenční měnič - regulátor BL-Ctrl, jež napájí motor s vrtulí, se ovládá datovými příkazy posílanými po I²C sběrnici. Stejně tak i ultrazvukový snímač výšky. Prostřednictvím této společné sběrnice jsou tedy tyto dva moduly připojeny k mikrokontroleru. Toto propojení v principu dostačuje pro základní samostatnou funkčnost přípravku. Aby bylo možné měnit program nahraný v paměti mikrokontroleru, bylo třeba zahrnout jeho propojení s ISP programátorem a sice rozhraním SPI. Ke komunikaci s PC je využito rozhraní USART mikrokontroleru a převodníku na rozhraní USB. Toto řešení je jednoduché na implementaci, nenáročné na propojení (dva signálové vodiče a zem) a dostatečně efektivní z hlediska rychlosti datového přenosu.



Obr. 4.1: Blokové schéma přípravku

4.2 Elektrické schéma a DPS

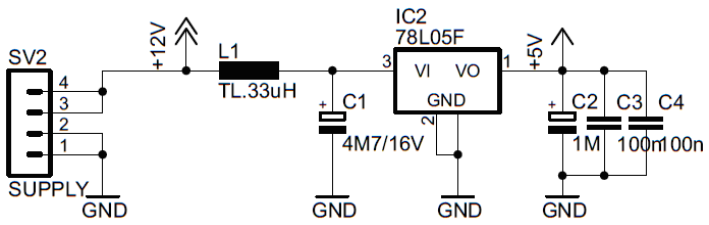
Z blokového schématu se vycházelo při tvorbě schématu elektrického zapojení. Pro připojení mikrokontroleru k jednotlivým periferiím byly dané vývody vyvedeny na konektory či pájecí plošky. Piny konektoru pro připojení ISP programátoru jsou zapojeny ve stejném pořadí, jako u často používaného programátoru USBasp. Za zmínku stojí propojení vodičů sběrnice I²C bez pull-up rezistorů, protože tyto rezistory je možné vnitřně připojit mikrokontrolerem. Schéma zapojení mikroprocesoru a k němu připojených periferií je zobrazeno na obrázku 4.2.



Obr. 4.2: Schéma zapojení mikroprocesoru

K napájení byl vzhledem k proudové náročnosti motoru zvolen PC zdroj typu AT s maximálním výkonem 200 W a proudovým odběrem 8 A z větve napětí +12 V. Ostatní nevyužívané větve (+5 V, -12 V apod.) byly odstraněny a na 5 V větev byly připojeny tři výkonové 20Ω rezistory, aby se zdroj rozběh naprázdno. Napětí 12 V ze zdroje je přivedeno na svorkovnici v DPS a přes kolíkovou lištu také na modul regulátoru motoru. Přes tlumivku L1 eliminující rušení od motoru či spínaného zdroje je

napájen stabilizátor napětí +5 V (viz. obr 4.3). Toto stabilizované napětí je napájecí pro mikrokontroler, modul SFR08 a jednu LED diodu pro signalizaci stavu mikrokontroleru.



Obr. 4.3: Schéma DPS – napájecí část

Kompletní schéma zapojení je obsahem přílohy A.1.

Součástky byly voleny převážně pro povrchovou (SMD), ale i klasickou montáž. Jednostranná deska plošných spojů byla navržena na stejné rozměry, jako má modul regulátoru motoru (40 x 40 mm), a optimalizována pro výrobu na CNC frézce.

Obsahem příloh A.2 až A.5 jsou: předloha desky plošných spojů, osazovací výkresy, seznam součástek.

4.3 Fyzická realizace

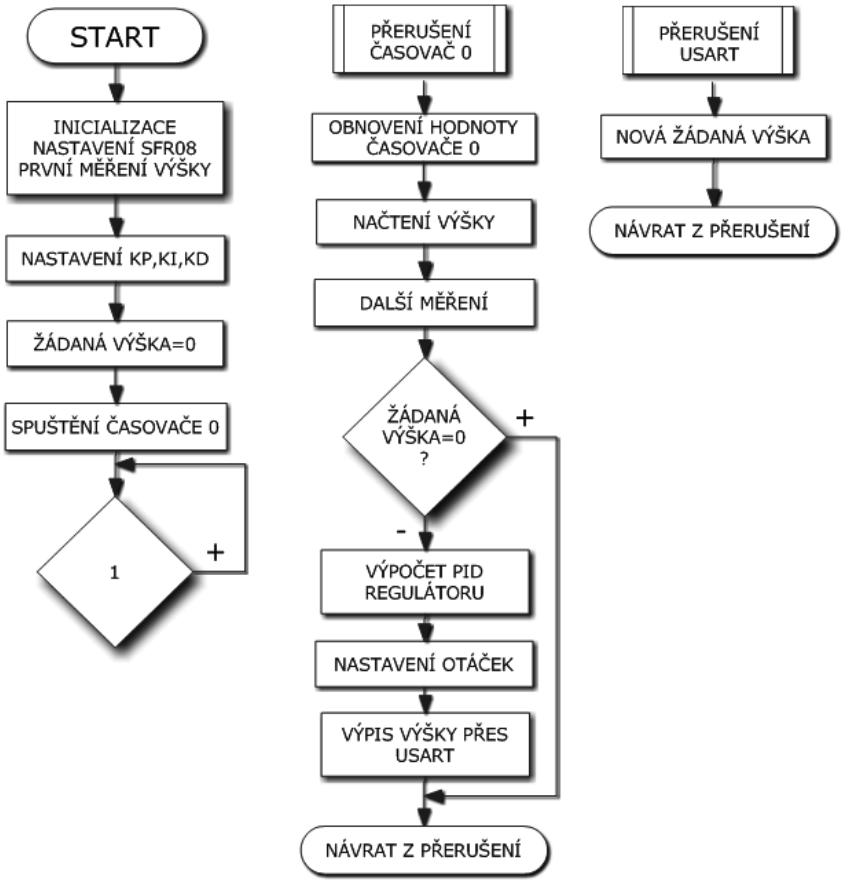
Na frézce vyrobená DPS byla osazena součástkami a propojena s dálkoměrem SFR08 a regulátorem motoru BL-Ctrl. K zajištění mechanické pevnosti byly DPS namontovány vedle sebe s použitím distančních sloupků. Elektronika a motor s vrtulí byly namontovány na mechanickou konstrukci z dílů stavebnice Merkur. Dvě vodicí tyče průměru 6 mm a délce 80 cm upevněné na podstavci vedou přípravek ve vertikálním směru. Tato konstrukce umožňuje pohyb od 3 do 73 cm výšky. Plochý kabel z přípravku je vyveden do krabičky s konektory pro připojení napájecího zdroje a USB-USART převodníku. Kompletní sestava přípravku je zobrazena na obr 4.4.

Obr. 4.4: Hardwarový přípravek při levitaci

5 REALIZACE SOFTWARU

5.1 Software mikrokontroleru

Zdrojový kód pro mikrokontroler ATmega32 byl psán v jazyce C s použitím vývojového prostředí a kompilátoru CodeVision C Compiler [9][10]. Byly napsány funkce pro komunikaci po I²C sběrnici (zápis a čtení ze zařízení) pro regulátor motoru i snímač SFR08. Program spolupracuje s aplikací běžící na PC (viz. kapitola 5.2) a komunikuje prostřednictvím sériového rozhraní USART. Jeho vývojový diagram je na obr. 5.1.



Obr. 5.1: Vývojový diagram programu mikroprocesoru

Na začátku běhu programu se inicializují časovače/čítače, rozhraní USART a I²C (TWI), vstupně/výstupní porty a přerušení. Ultrazvukovému snímači výšky je nastaven rozsah 903 mm a nízké zesílení. Je spuštěno první měření výšky. Dále se čeká na nastavení konstant Kp, Ki, Kd regulátoru (viz. kapitola 6.2) obslužnou aplikací. Implicitně je nastavena hodnota žádané výšky 0 cm. Je spuštěn časovač 0, hlavní chod

programu přechází do nekonečné smyčky. Perioda vzorkování byla vzhledem k možnostem ultrazvukového snímače zvolena 10 ms, což odpovídá frekvenci 100 Hz. Časování bylo vypočteno následovně: Zdroj hodinového signálu externí krystal osciluje s frekvencí 12 Mhz. S před-děličkou 1024 dostáváme frekvenci 11719 Hz. Tuto frekvenci zpracovává osmibitový časovač 0 čítáním impulzů nahoru. Abychom dostali frekvenci 100 Hz, je třeba vstupní frekvenci dělit ještě cca 117. Do registru čitače 0 TCNT0 proto zapisujeme hodnotu 255-117+1=139. Až časovač počítá do hodnoty 256, je vyvoláno přerušení přetečení časovače 0.

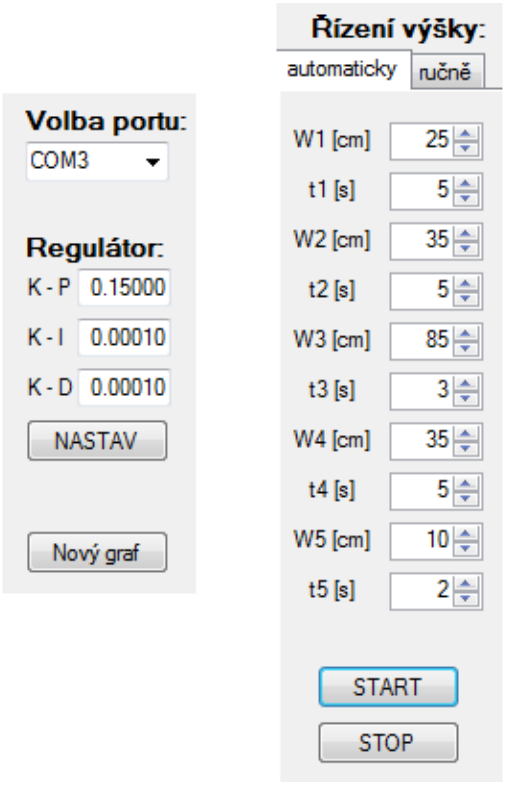
Je-li vyvoláno přerušení od USART, byla přijata data od obslužné aplikace. Hodnota přijatého bytu je uložena jako žádaná výška v centimetrech.

V obsluze přerušení časovače se nejprve obnovuje hodnota jeho registru na 139. Načte se změřená výška a iniciuje se další měření. Pokud není nastavena žádaná výška, rutina přerušení končí. Pokud je žádaná výška nenulová, je volána funkce pro výpočet akčního zásahu regulátoru PSD, její výstup nastaven regulátoru motoru a přes USART vypsána hodnota aktuální skutečné výšky v milimetrech se znakem # na konci. Při běžící regulaci tedy mikroprocesor posílá každých 10 ms novou hodnotu měřené výšky. Algoritmus pro výpočet PSD regulátoru bude odvozen v kapitole 6.2.

5.2 Software pro PC

Aby bylo možné regulaci výšky patřičně otestovat a porovnat vlastnosti regulátorů s různým nastavením, byla vytvořena aplikace pro operační systém Windows s uživatelským prostředím, které umožňuje intuitivní obsluhu a řízení hardwarového přípravku a také vykreslování grafů průběhu výšky. Aplikace byla psána v jazyce C# s využitím vývojového prostředí Microsoft Visual Studio 2010 a literatury [11],[12],[13],[14]. Spustitelný soubor je v příloze 3 na přiloženém DVD. Zdrojové kódy jsou obsahem přílohy 5 na DVD.

Velikost okna po spuštění maximalizovaného na celou obrazovku byla optimalizována pro rozlišení obrazovky 1600x900 px. Snímek obrazovky je obsahem přílohy B. Na obr. 5.2. jsou vyobrazeny všechny ovládací prvky. V nabídce pod označením *Volba portu:* je třeba zvolit číslo sériového portu, který byl přiřazen USB-USART převodníku připojenému do USB portu počítače. V kolonkách *K-P, K-I, K-D* se zadávají konstanty zesílení jednotlivých složek PID regulátoru s přesností na maximálně 5 desetinných míst. Tlačítkem *NASTAV* se tyto konstanty zapíší do příslušných proměnných v paměti mikrokontroleru.



Obr. 5.2: Ovládací prvky programu

Vyhodnocování kvality regulace budeme provádět na základě průběhů přechodových dějů při změně požadované výšky. Program pro toto nabízí dvě možnosti řízení: **ručně**, kdy se po stisku *START* žádaná výška v centimetrech zadává do kolonky *Výška* a zůstává konstantní až do další změny či stisku *STOP*, nebo **automaticky**. V automatickém režimu si nejdříve může uživatel navolit hodnoty výšky *W1* až *W5* pro časové úseky *t1* až *t5* nebo je ponechá ve výchozím stavu. Graf žádané výšky v čase je zobrazován dle aktuálního nastavení těchto parametrů. Stiskem START se spustí automatický proces testování. V nastavených okamžicích je mikrokontroleru posílána nová hodnota výšky. Křivka měřené výšky je okamžitě vynášena do grafu. Po dovršení nastaveného času nebo po stisku STOP je možné provést nové testování s jiným nastavením. Takto je možné porovnávat až 5 grafů, které jsou kresleny různými barvami.

Jak bylo zmíněno výše, s mikroprocesorem program komunikuje skrze USB-USAR převodník (viz. kapitola 2.2). Z programátorského hlediska se tomuto přistupuje jako ke komunikaci po sériové lince. V následující části zdrojového kódu je inicializace proměnných třídy `Form1`, okna programu a virtuálního sériového portu. Kompletní zdrojové kódy jsou obsahem přílohy 5 na DVD.

```
public partial class Form1 : Form
{   //deklarace a inicializace proměnných třídy
    SerialPort sp = new SerialPort();     //vytvořena instance sériový port
    List<byte> bBuffer = new List<byte>();//buffer typu byte pro příjem dat
    uint time = 0;                        //čitač času ve stovkách ms
    UInt16 W = 0;                         //žádaná výška
    bool pidSet = false;                  //příznak nastavení konstant regulátoru
    UInt16 YlineCount = 1;                //čitač kreslených grafů výšky
public Form1()
{
    InitializeComponent();
    FormBorderStyle = FormBorderStyle.Fixed3D;     //styl rámečku okna
    this.WindowState = FormWindowState.Maximized;  //maximalizuje okno
    DrawChartW();                                  //volání metody vykreslující
                                                      graf žádané výšky
    List<String> tList = new List<String>();       //seznam pro aktivní COM porty
    foreach (string s in SerialPort.GetPortNames())//získá aktivní COM porty
      tList.Add(s);                                //a přidá je do seznamu
    tList.Sort();                                  //uspořádá seznam
```

```
    comboBoxPort.Items.Clear();                    // a předá jej do vysouvací
    comboBoxPort.Items.AddRange(tList.ToArray());  // nabídky
    comboBoxPort.SelectedIndex = 0;                //
    sp.PortName = comboBoxPort.SelectedItem.ToString();//port s nejnižším číslem
    sp.BaudRate = 9600;             //následuje nastavení pro komunikaci po portu
    sp.Parity = Parity.None;
    sp.DataBits = 8 ;
    sp.StopBits = StopBits.One;
    sp.DataReceived += new SerialDataReceivedEventHandler(port_DataReceived);
                                                   //metoda pro příjem dat
    ErrorLabel.Text="init "+sp.PortName;//název portu vypsán v levém horním rohu
}...}
```

Při běžící regulaci mikroprocesor posílá každých 10 ms novou hodnotu měřené výšky ve formátu `456#` jako ASCII znaky. Číslo je v jednotkách mm, znak # slouží pro oddělení po sobě jsoucích hodnot. Data přijatá po sériovém portu jsou zachycována do bufferu jako hodnoty typu byte. Jednotlivé byty reprezentují číslo ASCII znaku dané číslice. Např. 0 je reprezentována hodnotou 48, 1 hodnotou 49 atd. Dekadickou hodnotu získáme postupem, jak je ukázáno níže. Metoda `timer1_Tick` je při běžící regulaci volána každých 100 ms.

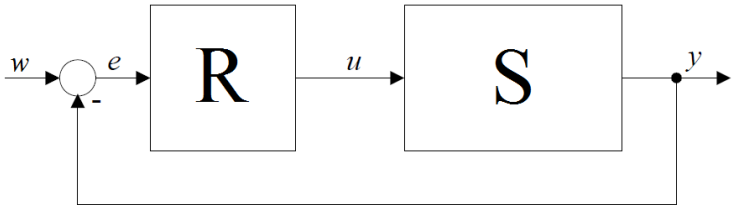
```
private void timer1_Tick(object sender, EventArgs e)
{
    int index = 0;                        //index první hodnoty výšky v bBuffer
    int endIndex = 0;                     //index poslední hodnoty výšky v bBuffer
    List<int> pBuffer = new List<int>();   //buffer hodnot v dekadickém vyjádření
    List<byte> bCache = new List<byte>();  //pro nekompletní poslední hodnotu
    time++;                          //čas aktuálního testování ve stovkách ms
    if ( bBuffer.FindIndex(item => item == 35) != 3)//vyhledání prvního #
    index = bBuffer.FindIndex(item => item == 35) + 1;//číslo před # není
                                                  //kompletní, index na následující
    endIndex = bBuffer.LastIndexOf(35) - 3;      //index hodnoty před posledním #
    if ( endIndex != bBuffer.Count-4 )           //není-li # poslední v bufferu,
    for (int i = endIndex+4; i <= bBuffer.Count-1; i++)//hodnoty za ním uchováme
    {
       bCache.Add(bBuffer[i]);                   //do bufferu bCache
    }
    ErrorLabel.Text=index.ToString()+(bBuffer.Count-1-endIndex-3).ToString();
    //zobrazení informací o počátečním a konečném indexu
    for ( ; index <= endIndex; index = index+4)//hodnoty z bBuffer v podobě ASCII
                                                  //znaků číslic
    {
       pBuffer.Add(100*(bBuffer[index]-48)+10*(bBuffer[index+1]48)+
       (bBuffer[index+2]-48)); //převedeme do pBuffer na skutečné hodnoty
    }
    for (int i=1; i<=(pBuffer.Count); i++)       //jednotlivé hodnoty výšky
    {
    if (pBuffer[i-1]<750)                        //pokud nejsou chybové
       chart1.Series[YlineCount].Points.AddXY((double)(time-1+(double)i/
       pBuffer.Count)/10,(double)pBuffer[i-1]/10);//přidáme do grafu
       na posledních 100 ms. Takto je ošetřen i případ, kdy změníme periodu
       vzorkování regulátoru. Hodnoty budou rozprostřeny na posledních 100 ms
    }
    bBuffer.Clear();                             //vymazání bufferu portu
    bBuffer.AddRange(bCache);//a přidání znaků z konce bufferu pro příští volání
...}
```

V nastavených časových intervalech je mikrokontroleru posílána nová hodnota žádané výšky následovně:

```
if (time == 10 * t1numericUpDown.Value & t2numericUpDown.Value != 0)
//je-li konec intervalu t1 a není-li t2 nula
{
       W = (UInt16)W2numericUpDown.Value;       //žádaná výška je W2
       sp.Write(new byte[] { (byte)W }, 0, 1);  //nastavení výšky
}
```

6 REGULACE – SIMULACE, TESTOVÁNÍ

Regulace, tedy řízení se zpětnou vazbou, probíhá v uzavřené smyčce. Na řízenou soustavu *S* s výstupem *y* působí regulátor *R* akční veličinou *u*. Řídící veličina *w* je v rozdílovém členu porovnávána s hodnotou regulované veličiny *y* a výsledná regulační odchylka *e* je vstupní veličinou regulátoru. Regulátor tak může reagovat nejen na změnu řídící veličiny, ale i na skutečnou hodnotu regulované veličiny a důsledky poruch působících na různých místech regulačního obvodu [15]. Schéma tohoto základního regulačního obvodu je na obr 8.1.



Obr. 6.1: Blokové schéma regulace v uzavřené smyčce

6.1 PID regulátor

V průmyslu a automatizační technice je nejrozšířenější typ regulátoru PID (proporcionálně-integračně-derivační). P složka zesiluje regulační odchylku. I složka ji integruje, je tak schopna zajistit nulovou ustálenou odchylku tam, kde P složka nestačí, ale zvyšuje překmit výstupní veličiny a prodlužuje tak regulační děj. D složka derivuje průběh odchylky, urychluje přechodový děj a zmenšuje překmity, ale na skokové změny reaguje velikým akčním zásahem, což může být u některých systémů nežádoucí. Základní rovnice PID regulátoru udávající hodnotu akčního zásahu *u* v čase *t* je dána vztahem

$$u(t) = K\left(e(t) + \frac{1}{T_{\mathrm{I}}} \int_0^t e(\tau)\mathrm{d}\tau + T_{\mathrm{D}} \frac{\mathrm{d}e(e)}{\mathrm{d}t}) \right), \qquad (6.1)$$

kde *K* je zesílení PID regulátoru,

T_I integrační časová konstanta,

T_D derivační časová konstanta,

e(t) regulační odchylka v čase *t.*

S použitím Laplaceovy transformace za předpokladu nulových počátečních podmínek dostáváme z (8.1) přenosovou funkci ideálního PID regulátoru

$$F_R(s) = \frac{U(s)}{E(s)} = K\left(1 + \frac{1}{T_I s} + T_D s)\right), \qquad (6.2)$$

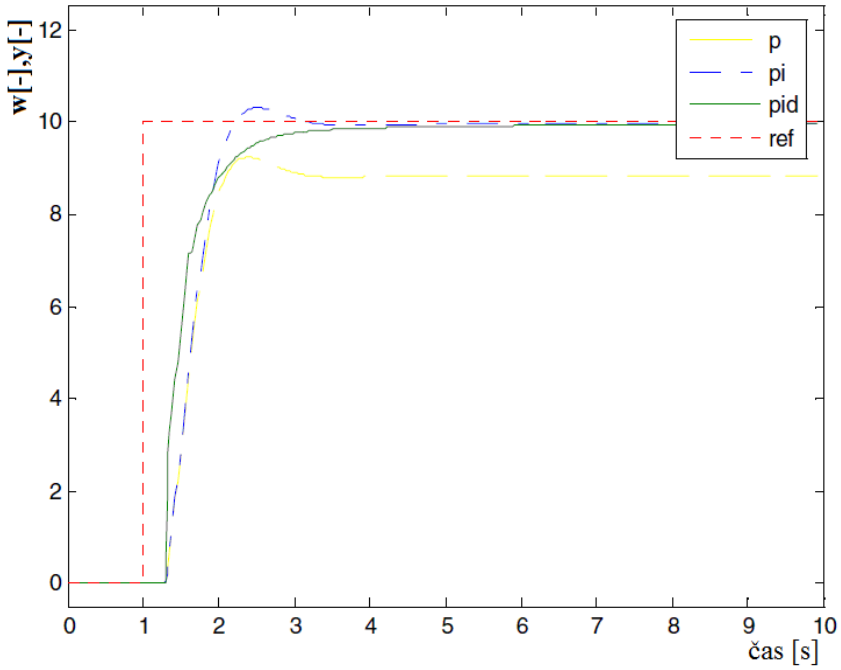
kde s je operátorem Laplaceovy transformace,

$F_R(s)$ operátorový přenos regulátoru,

$U(s)$ Laplaceův obraz výstupu,

$E(s)$ Laplaceův obraz vstupu (regulační odchylky).

Na obr. 8.2 je zobrazen typický průběh odezvy systému na skokovou změnu žádané veličiny při použití P, PI a PID regulátoru. S dobře určenými konstantami jednotlivých složek je obecně možné docílit velice dobrých výsledků.



Obr. 6.2: PID regulátor - odezva na skokovou změnu žádané veličiny

6.2 PSD regulátor

V číslicovém řízení se používá diskrétní ekvivalent regulátoru PID, regulátor PSD (proporcionálně-sumačně-diferenční). V literatuře je často označován jako diskrétní PID [16]. Pro získání diskrétní verze spojitého PID regulátoru vyjdeme z rovnice (8.1). Integrační a derivační složku rovnice je třeba převést na diskrétní funkce. Zavedeme konstanty T_0 pro periodu vzorkování a k pro pořadové číslo vzorku. Integrál nahradíme prostou sumací a to tak, že spojitou funkci aproximujeme po úsecích T_0 konstantní funkcí. Dostáváme takzvanou náhradu obdélníky zprava.

$$\int_0^t e(\tau)\mathrm{d}\tau \approx T_0 \sum_{i=0}^{k} e(i) \tag{6.3}$$

Derivaci nahradíme diferencí 1.řádu podle vztahu

$$\frac{\mathrm{d}e(e)}{\mathrm{d}t} \approx \frac{e(k)-e(k-1)}{T_0} \tag{6.4}$$

Rovnice pro výstup PSD regulátoru *u* v kroku *k* tedy bude mít tvar

$$u(k) = K\left(e(k) + \frac{T_0}{T_\mathrm{I}} \cdot \sum_{i=1}^{k} e(i) + T_D \frac{e(k)-e(k-1)}{T_0} \right) \tag{6.5}$$

S použitím Z-transformace dostaneme z (8.5) přenos PSD regulátoru

$$F_R(z) = \frac{U(z)}{E(z)} = K\left(1 + \frac{T_0 z^{-1}}{T_\mathrm{I}(1-z^{-1})} + \frac{}{T_0} T_\mathrm{D} s) \right), \tag{6.6}$$

kde *z* je operátorem Z-transformace,

$F_R(z)$ operátorový přenos regulátoru,

$U(z)$ Z obraz výstupu,

$E(z)$ Z obraz vstupu.

Podrobný teoretický rozbor různých typů diskrétních regulátorů je v literatuře [17].

Pro implementaci PSD regulátoru do algoritmu v jazyce C ještě upravíme rovnici (6.5) na tvar

$$u(k) = Kp \cdot e(k) + Ki \cdot T_0 \sum_{i=1}^{k} e(i) + Kd \cdot \frac{e(k)-e(k-1)}{T_0}, \tag{6.7}$$

kde Kp je zesílení proporcionální složky,

Ki zesílení integrační složky,

Kd zesílení derivační složky,

e(k) regulační odchylka s pořadovým číslem k

a pro přepočet konstant platí

$$Kp = K \tag{6.8}$$

$$Ki = \frac{K}{T_\mathrm{I}} \tag{6.9}$$

$$Kd = K \cdot T_\mathrm{D} \tag{6.10}$$

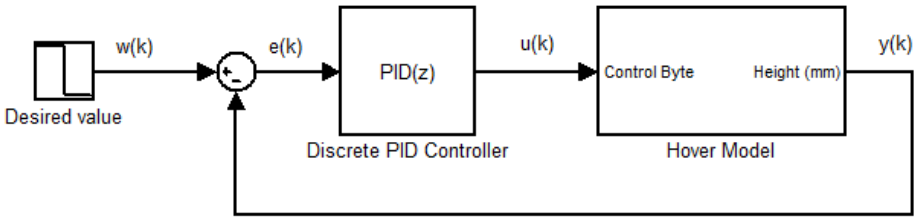
Jednoduchá funkce v jazyce C pro výpočet PSD regulátoru podle (6.7) může vypadat následovně:

```
unsigned char PID(unsigned int W, unsigned int Y)
{
  static int preE = 0;
  static float integral = 0;
  float derivative;
  int E;
  unsigned char output;

  E = W - Y;                //regulační odchylka
  integral = integral+E*dt;//integrál násobený periodou vzorkování
  derivative = (E - preE)/dt;//derivace dělená periodou vzorkování
  output = Kp*E + Ki*integral + Kd*derivative;
  preE = E;                //přenesení odchylky do dalšího kroku
  return output;           //výstup
}
```

6.3 Simulace

V prostředí Matlab Simulink bylo pro simulaci chování hardwarového přípravku vytvořeno zapojení dle obr. 6.3. Žádaná hodnota *w* je námi nastavená hodnota výšky, regulátor představuje implementovaný algoritmus v mikroprocesoru, regulovanou soustavou (Hover Model) je hardwarový přípravek a výstupní hodnotou je jeho skutečná výška.

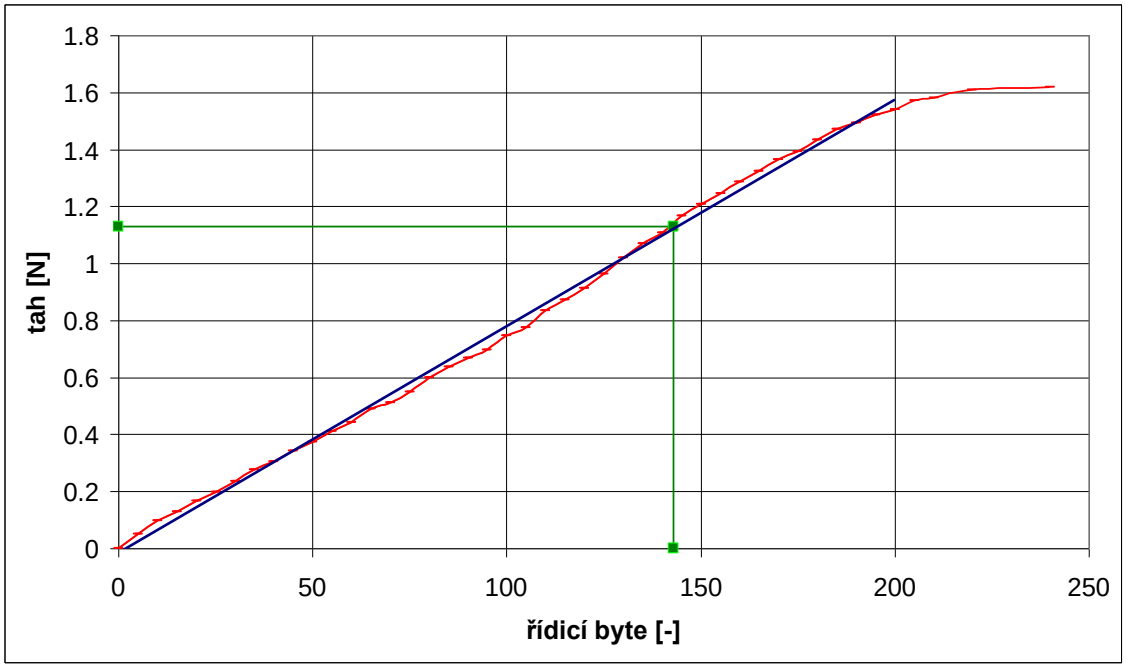


Obr. 6.3: Regulační smyčka s PSD regulátorem

Model regulované soustavy

Experimentálně bylo zjištěno, že při změně hodnoty nastavované regulátoru motoru, tedy změně otáček je přechodný děj velice rychlý a prakticky neměřitelný, proto byl při simulaci zanedbán a změna otáček je považována za okamžitou. Důležitou vlastností je to, jakou silou působí vztlak vrtule ve směru proti gravitační síle při daných otáčkách motoru, respektive při dané hodnotě řídicího bytu regulátoru motoru. Hodnota otáček v tomto případě není důležitá. Bylo tedy provedeno měření závislosti tahu vrtule (síly) na hodnotě řídicího bytu nastaveného regulátoru Dual BL-Ctrl. Přípravek byl mechanicky spojen se závažím položeným na přesné váze. Naměřený průběh je na

obr.8.4. vynesen červenou křivkou. Tabulka měření a přepočtu je obsahem přílohy C.



Obr. 6.4: Graf závislosti tahu motoru na řídicím slově

Z grafu je vidět, že závislost je přibližně lineární. Od hodnoty 200 výše se rapidně projevuje jakási saturace, kdy motor už není schopen vyšších otáček a proto tato oblast nebude využita. Výstupní hodnota z regulátoru je omezena na maximálně 200. Lineární oblast grafu byla aproximována přímkou se směrnicí

$$k_t = 0.007946 \qquad (6.11)$$

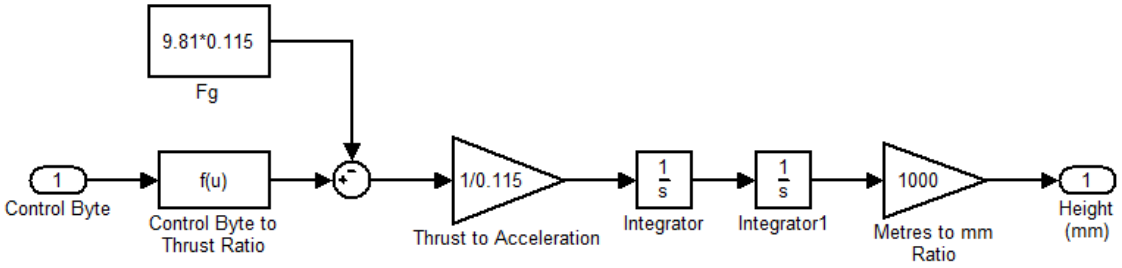
Dále je na obr. 8.4 zeleně vyznačen teoretický bod levitace, kdy gravitační síla F_g a vztlaková síla F_{vz} jsou vyrovnány, podle vztahu

$$F_{vz} = Fg = m \cdot g = 0{,}115 \cdot 9{,}81 = 1{,}12815N \qquad (6.12)$$

Této vztlakové síle odpovídá hodnota řídicího bytu

$$\check{r}.byte = \frac{F_{vz}}{k} = \frac{1{,}12815}{0{,}007946} \cong 142 \qquad (6.13)$$

S těmito poznatky byl vytvořen model regulované soustavy. Zobrazen je na obrázku 6.5.



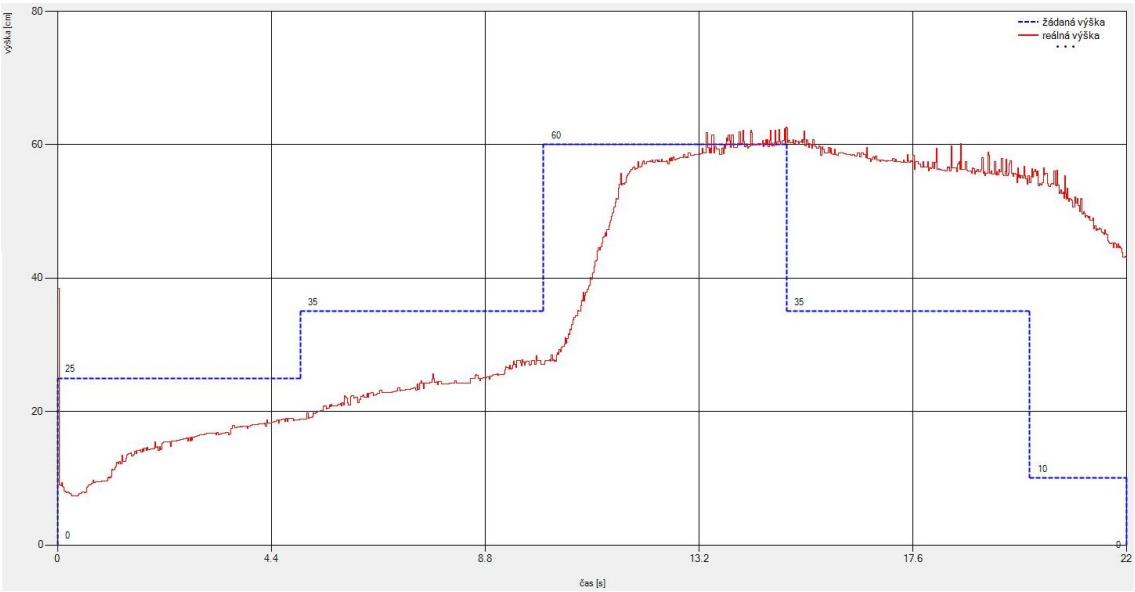
Obr. 6.5: Model regulované soustavy

Vstupem je řídicí byte nastavující otáčky motoru. Vynásobením konstantou k_t podle (6.11) dostaneme vztlakovou sílu v kladném směru. V sumátoru je od ní odečítána gravitační síla. Výsledná síla je vydělena hmotností a získáváme tak zrychlení. První integrací zrychlení je rychlost. Dalším integrováním dostaneme polohu, tedy výšku v metrech, po vynásobení tisícem v milimetrech.

Simulační schéma z obr. 6.3 bylo modifikováno tak, že k výstupu regulátoru je přičítána konstanta 142 zajišťující vyrovnání gravitační síly. Podle simulace tohoto systému byly upravovány konstanty použitého diskrétního PID regulátoru pro co možná nejlepší průběhy přechodových dějů.

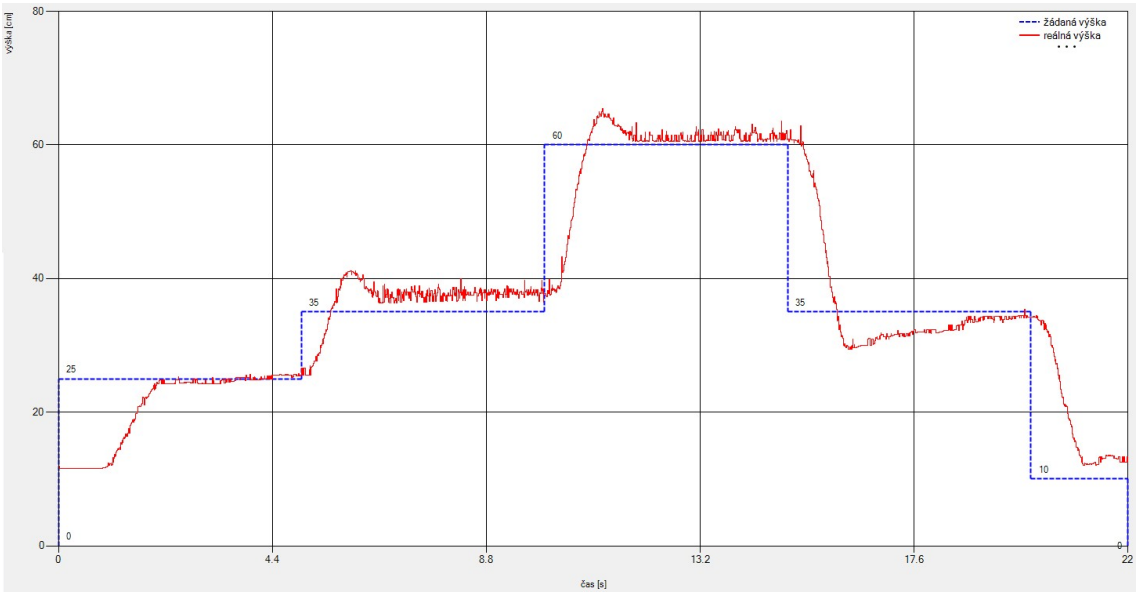
6.4 Ladění regulátoru

S výsledky simulace se přešlo do praktického testování regulace. Jak ukazuje graf na obr. 6.6, kvalita regulace s konstantami zesílení Kp,Ki a Kd jednotlivých složek podle simulace nebyla dobrá.



Obr. 6.6: Průběh regulace výšky 1

Částečně je to způsobenou nedokonalým popisem modelu soustavy, částečně i nežádoucími vlastnostmi přípravku jako je tření o vodicí tyče a do určité míry na výšce závislé vztlakové síle. V nižších polohách je totiž vztlak větší. Co můžeme také na tomto průběhu pozorovat je, že křivka měřené výšky je značně ovlivněna rušením. Toto zásadně ovlivňuje chování derivační složky. Algoritmus výpočtu PSD regulátoru v mikroprocesoru byl proto doplněn o filtraci derivační složky jednoduchým filtrem typu plovoucí průměr. Pro zlepšení vlastností bylo také doplněno jednoduché eliminování integrálního wind-up efektu (viz zdrojový kód v příloze 4 na DVD). Graf regulace výšky s experimentálně odladěným regulátorem je na obr. 6.7. Pro upřesnění, graf je kreslen ze všech měřených hodnot, tedy nefiltrovaných, i když je použita filtrace derivační složky.



Obr. 6.7: Průběh regulace výšky 2

Hodnoty zesílení jednotlivých složek:

Kp=0.083

Ki=0.00013

Kd=0.0022

7 ZÁVĚR

Byl navržen a realizován přípravek, na kterém je možné experimentálně zkoušet a měřit algoritmy regulace výšky. Řídicí mikrokontroler je možné programovat prostřednictvím ISP programátoru, což umožňuje měnit jeho software a implementovat další algoritmy regulace do jeho paměti. Díky vyvinuté aplikaci pro PC lze pohodlně tyto algoritmy testovat a vyhodnocovat kvalitu regulace. Tuto sestavu by bylo možné využít např. i jako učební pomůcku v laboratořích.

Pro zlepšení by bylo vhodné implementovat do řídicího programu mikroprocesoru i jiné algoritmy, např. stavový nebo některý z fuzzy regulátorů. Také by bylo užitečné moci měnit periodu vzorkování, která v některých případech výrazně ovlivňuje regulační děj.

Implementován byl funkční PSD regulátor. Simulací matematického modelu byly nalezeny konstanty zesílení jeho složek, které se však v praktickém testu neprojevily jako vhodné a bylo je třeba experimentálně na reálném přípravku doladit. I přes to ale dala simulace alespoň orientační použitelné hodnoty.

Obdobným postupem návrhu regulátoru je možno dospět k vhodnému regulátoru i například pro quadro-kopter. Po přepočtu hmotnosti a tahu vrtulí by se dal takovýto ekvivalent nalezeného regulátoru výšky implementovat do zdrojových kódů pro řízení kopteru.

LITERATURA

[1] *Robot Electronics* [online]. 2010 [cit. 2011-11-16]. SRF08 Ultra sonic range finder. Dostupné z WWW: <http://www.robot-electronics.co.uk/htm/srf08tech.html>.

[2] *MikroKopter* [online]. 2007 [cit. 2011-11-24]. BrushlessCtrl. Dostupné z WWW: <http://www.mikrokopter.de/ucwiki/en/BrushlessCtrl>.

[3] BABČANÍK, Jan. *Hw.cz : Teorie a praxe* [online]. 1.12.2006 [cit. 2011-12-02]. Začínáme s mikroprocesory Atmel AVR. Dostupné z WWW: <http://www.hw.cz/Teorie-a-praxe/Zacatecnici/ART1767-Zaciname-s-mikroprocesory-Atmel-AVR.html>.

[4] *Datasheet ATmega32A* [online]. San Jose : Atmel Corporation, 6/2008, 2/2011

[cit. 2011-12-02]. Dostupné z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc8155.pdf>.

[5] I²C. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16.6.2007, last modified on 23.1.2011 [cit. 2011-10-13]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/I²C>.

[6] *Hw.cz : Teorie a praxe* [online]. 19.5.2000 [cit. 2011-12-07]. Stručný popis sběrnice I²C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877. Dostupné z WWW: <http://www.hw.cz/design/i2c_pic/index.html>.

[7] USART. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 1.4.2008, 20.11.2011 [cit. 2011-12-11]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/USART>.

[8] SPI. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 14.5.2007, 18.8.2011 [cit. 2011-12-11]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/SPI>.

[9] VÁŇA, Vladimír. *Mikrokontroléry ATMEL AVR*: *programování v jazyce C : popis a práce ve vývojovém prostředí CodeVisionAVR C*. 1. vyd. Praha: BEN - technická literatura, 2003, 215 s. ISBN 80-730-0102-0.

[10] MANN, Burkhard. *C pro mikrokontroléry*: *ANSI-C, kompilátory C, spojovací programy - linkery, práce s ATMEL AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky*. Vyd. 1. Praha: BEN, 2003, 279 s. ISBN 80-730-0077-6.

[11] PETZOLD, Charles. *Programování Microsoft Windows Forms v jazyce C#*. Vyd. 1. Překlad Karel Voráček. Brno: Computer Press, 2006, 356 s. ISBN 80-251-1058-3.

[12] SHARP, John. *Microsoft Visual C# 2010*: *krok za krokem*. Vyd. 1. Brno: Computer Press, 2010, 696 s. ISBN 978-80-251-3147-3.

[13] HANÁK, Ján. *Praktické objektové programování v jazyce C# 4.0*. Vyd. 1. Brno: Artax, 2009, 180 s. Microsoft (Artax). ISBN 978-80-87017-07-4.

[14] PRATA, Stephen. *Mistrovství v C++*. 1. vyd. Praha: Computer Press, 2001, 966 s. ISBN 80-722-6339-0.

[15] BLAHA, Petr a Petr VAVŘÍN. *Řízení a regulace I*: *Základy regulace lineárních systémů - spojité a diskrétní*. Brno, [2009]. Skriptum. VUT v Brně.

[16] ATMEL CORPORATION. *AVR221: Discrete PID controller*: *Application Note* [online]. San Jose, 2006, 10 s. [cit. 2012-05-06]. Dostupné z: http://www.atmel.com/Images/doc2558.pdf

[17] PIVOŇKA, Petr. *Číslicová řídicí technika*. Brno, 2003. Skriptum. VUT v Brně.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

c	Rychlost zvuku ve vzduchu
ϑ	Teplota
w	řídicí veličina
e	regulační odchylka
u	akční veličina
y	regulovaná veličina
t	čas
k	pořadové číslo vzorku

TOF	Time Of Flight
RISC	Reduced Instruction Set Computer
ALU	Arithmetic Logic Unit
MIPS	Million Instructions Per Second
ISP	In System Programming
USART	Universal Synchronous / Asynchronous Receiver and Transmitter
PC	Personal Computer
I²C	Inter-Integrated Circuit
TWI	Two Wire Interface
SDA	Synchronous Data
SCL	Synchronous Clock
ACK	Acknowledge
SPI	Serial Peripheral Interface
SS	Slave Select
CS	Chip Select
MOSI	Master Output Slave Input
MISO	Master Input Slave Output
SCLK	Synchronous Clock
USB	Universal Serial Bus
PID	Proporcionální-Integrační-Derivační regulátor
PSD	Proporcionální-Sumační-Diferenční regulátor

SEZNAM PŘÍLOH

A Návrh HW zařízení **45**

A.1 Obvodové zapojení hlavní DPS 45
A.2 Deska plošného spoje – bottom (strana spojů) 46
A.3 Osazovací výkres DPS – top (strana součástek) 46
A.4 Osazovací výkres DPS – bottom (strana spojů) 47
A.5 Seznam součástek 48

B Snímek obrazovky vyvinuté aplikace pro Windows **49**

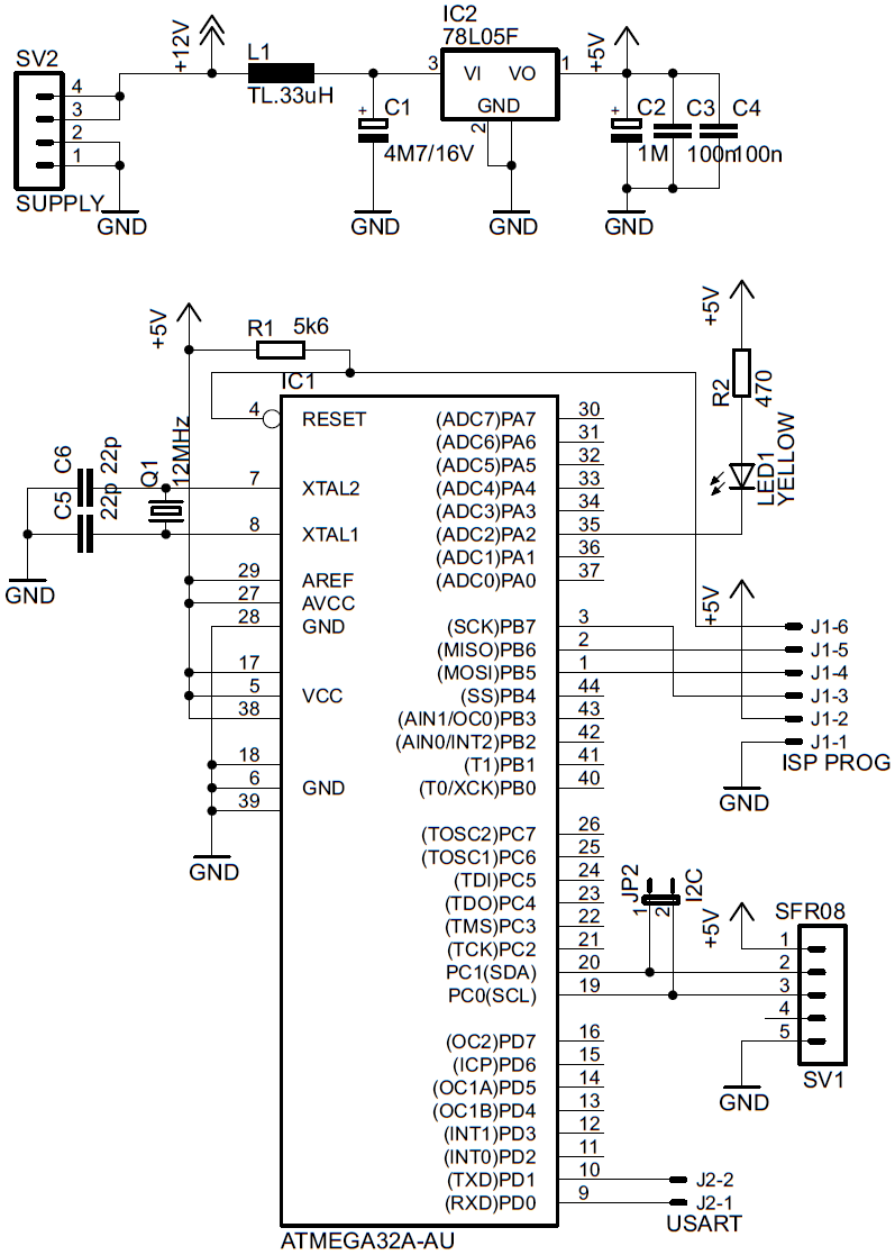
C Tabulka měření tahu vrtule **50**

Přílohy na DVD:

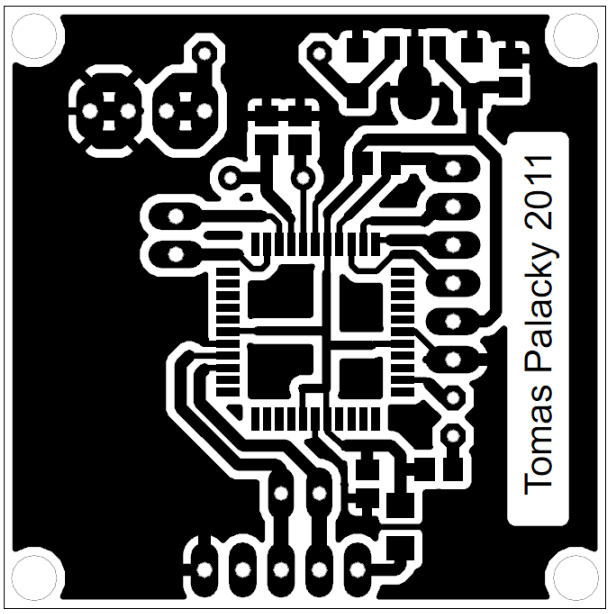
1. Elektronická verze této bakalářské práce
2. FTDI ovladače pro USB-USART převodník
3. Aplikace pro Windows – Testování regulace výšky
4. Zdrojové kódy SW mikrokontroleru
5. Zdrojové kódy SW pro Windows – Testování regulace výšky

A NÁVRH HW ZAŘÍZENÍ

A.1 Obvodové zapojení hlavní DPS

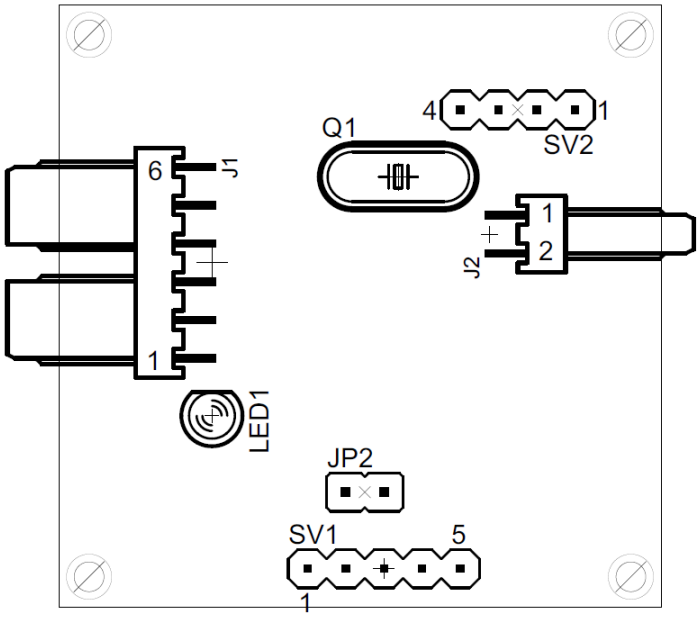


A.2 Deska plošného spoje – bottom (strana spojů)



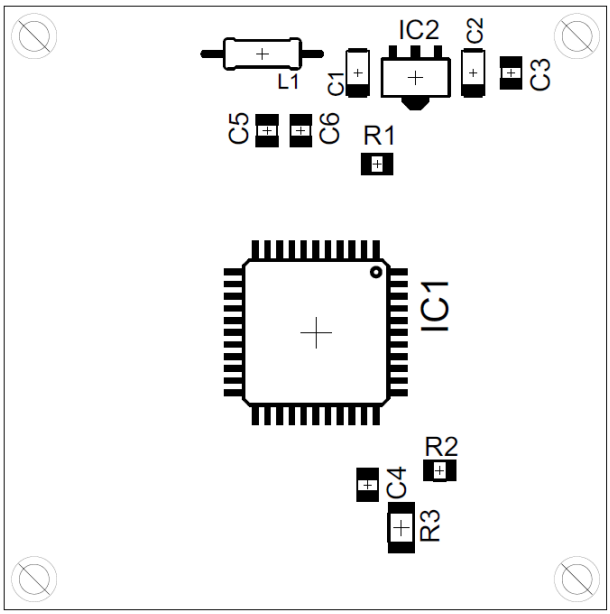
Rozměr desky 40 x 40 [mm], měřítko M2:1

A.3 Osazovací výkres DPS – top (strana součástek)



Rozměr desky 40 x 40 [mm], měřítko M2:1

A.4 Osazovací výkres DPS – bottom (strana spojů)

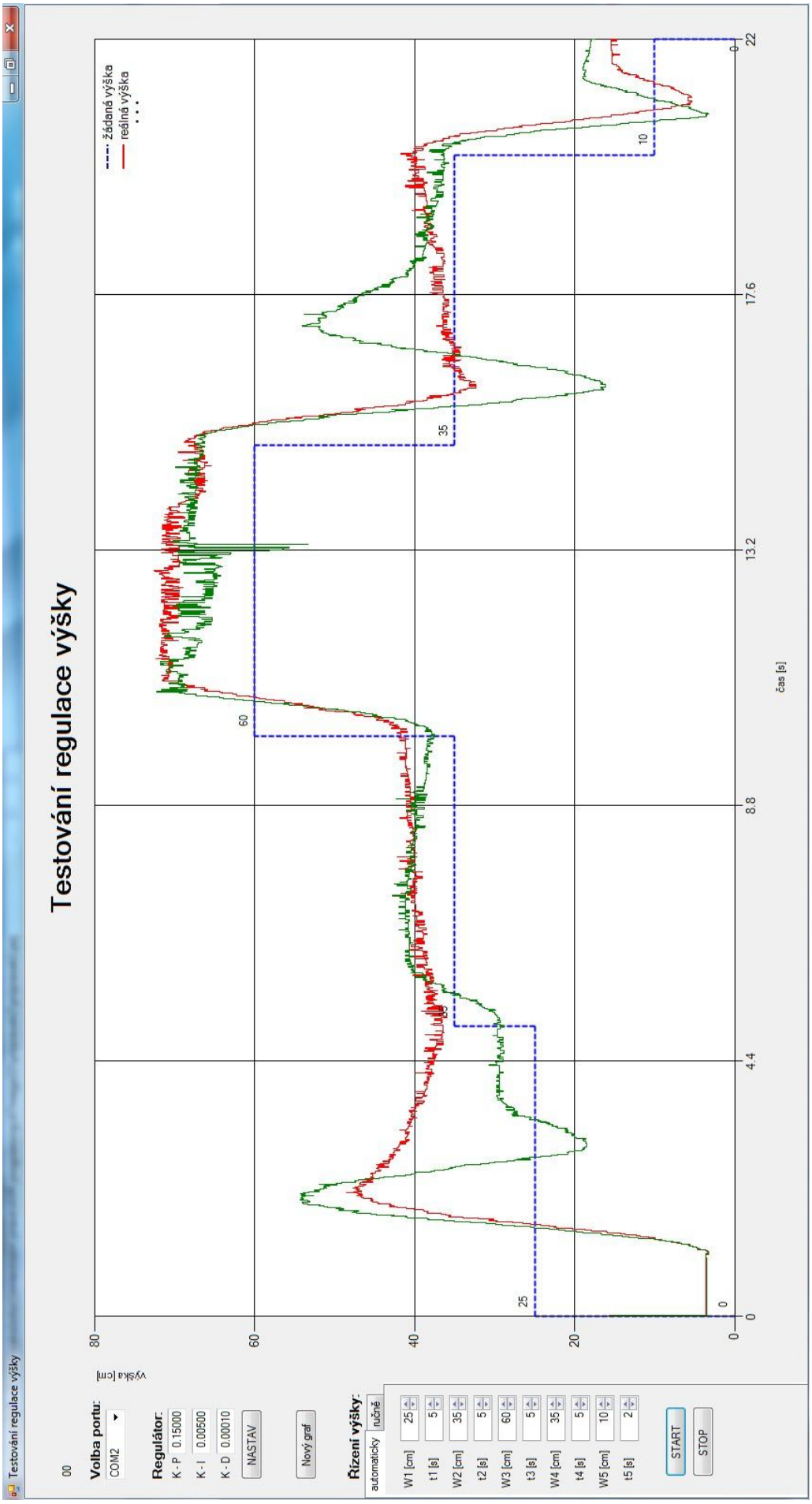


Rozměr desky 40 x 40 [mm], měřítko M2:1

A.5 Seznam součástek

Označení	Hodnota	Pouzdro	Popis
C1	4M7/16V	SMC_A	kondenzátor tantalový
C2	1M/10V	SMC_A	kondenzátor tantalový
C3	100n	C0805	kondenzátor keramický
C4	100n	C0805	kondenzátor keramický
C5	22p	C0805	kondenzátor keramický
C6	22p	C0805	kondenzátor keramický
IC1	ATMEGA32A	TQFP44	mikrokontroler
IC2	78L05F	SOT89	stabilizátor napětí
J1	PSH02-06W	7395-06	konektor se zámkem úhlový 6 pinů
J2	PSH02-02W	7395-02	konektor se zámkem úhlový 2 piny
JP2		JP1	kolíková řada oboustranná 2 piny
L1	33uH	0204/7	tlumivka axiální
LED1	YELLOW	LED3MM	LED dioda
Q1	12MHz	HC49/S	krystal
R1	47k	R0805	rezistor
R2	470R	R0805	rezistor
R3	0R	R1206	rezistor
SV1		MA05-1	kolíková řada úhlová 5 pinů
SV2		JP1	kolíková řada oboustranná 2 piny

B SNÍMEK OBRAZOVKY VYVINUTÉ APLIKACE PRO WINDOWS



C TABULKA MĚŘENÍ TAHU VRTULE

hmotnost přípravku	115	g
hmotnost závaží	958	g
hm. Přípravku se závažím	1073	g
tah levitace	1.12815	N

řídicí byte	měř.hm [g]	odlehčení [g]	tah [N]
0	1073	0	0
5	1068	5	0.04905
10	1063	10	0.0981
15	1060	13	0.12753
20	1056	17	0.16677
25	1053	20	0.1962
30	1049	24	0.23544
35	1045	28	0.27468
40	1042	31	0.30411
45	1038	35	0.34335
50	1035	38	0.37278
55	1031	42	0.41202
60	1028	45	0.44145
65	1023	50	0.4905
70	1021	52	0.51012
75	1017	56	0.54936
80	1012	61	0.59841
85	1008	65	0.63765
90	1005	68	0.66708
95	1002	71	0.69651
100	997	76	0.74556
105	994	79	0.77499
110	988	85	0.83385
115	984	89	0.87309
120	980	93	0.91233
125	975	98	0.96138
130	969	104	1.02024
135	964	109	1.06929
140	960	113	1.10853
145	954	119	1.16739
150	950	123	1.20663
155	946	127	1.24587
160	942	131	1.28511
165	938	135	1.32435
170	934	139	1.36359
175	931	142	1.39302
180	927	146	1.43226
185	923	150	1.4715
190	921	152	1.49112
195	918	155	1.52055
200	916	157	1.54017
205	913	160	1.5696
210	912	161	1.57941
220	909	164	1.60884
240	908	165	1.61865