České Vysoké Učení Technické v Praze

fakulta elektrotechnická, Katedra řídicí techniky

Konstrukce a realizace řídicí jednotky malého proudového motoru

DIPLOMOVÁ PRÁCE

2010

Vypracoval: Michal Vosecký

Vedoucí práce: Ing. Ondřej Špinka

Prohlášení

Prohlašuji, že jsem svou diplomovou (bakalářskou) práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Chtěl bych poděkovat všem lidem, bez nichž by má diplomová práce nemohla vzniknout. Děkuji především vedoucímu diplomové práce Ing. Ondřeji Špinkovi za jeho vedení, připomínky, trpělivost a cenné rady při řešení této práce. V neposlední řadě bych chtěl poděkovat rodině a všem přátelům za podporu. Děkuji!

Abstrakt

Diplomová práce se zabývá konstrukcí a realizací řídicí jednotky malého proudového motoru. Práce vychází z výsledků získaných v bakalářské práci autora a diplomové práce Miroslava Hájka.
V první části je uveden obecný princip funkce modelářské turbíny. Druhá část se zabývá návrhem řídicí jednotky (systém FADEC) s popisem jednotlivých HW částí a senzorů. Prostřední část se věnuje popisu návrhu jednotlivých stavových automatů řízení a komunikace, SW implementaci a popisu komunikačního protokolu. Jednotka je navrhována tak, aby byla kompatibilní s vizualizačním SW vytvořeným Miroslavem Hájkem. V poslední části je uveden popis použitého PID regulátoru (O. Špinka, O. Holub) a implementovaných pracovních režimů turbíny.

Abstract

This thesis deals with construction and design of a digital control unit for a modeler jet engine. The thesis references to results from bachelor thesis by autor and thesis by Miroslav Hájek.
A brief description of basic functional principles is given in the first part. Second part deals with design of structure control unit (FADEC) with descriptions of all HW parts and sensors. The middle part describes design and implementation of all state machines for control and communication. Unit is designed to be compatibile with visualization SW provided by Miroslav Hájek. In the last part is description of using PID controller from authors Ondřej Špinka and Ondřej Holub and all implemented modes of turbine.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Michal Vosecký**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Konstrukce a realizace řídicí jednotky malého proudového motoru**

Pokyny pro vypracování:

1. Navrhněte a naprogramujte základní stavový automat řízení proudového motoru.
2. Prostudujte komunikační protokol a implementujte komunikaci s nadřízeným systémem.
3. Navrhněte a implementujte řídicí algoritmy pro jednotlivé režimy chodu motoru.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Ondřej Špinka

Platnost zadání: do konce letního semestru 2010/2011

L.S.

prof. Ing Michael Šebek, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 22. 12. 2009

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Sen člověka „létat“ patří mezi ty nejstarší sny od počátku lidstva. Již okolo roku 400 před Kristem objevili Číňané draka, předmět, který mohl létat ve vzduchu. Mnoho let od vynalezení draků se lidé snažili napodobit létání ptáků. Konstruovali křídla vyrobená z peří, lehkého dřeva a připevňovali si je na ruce, bohužel výsledky byly katastrofální. Kolem roku 1480 udělal Leonardo da Vinci první reálnou studii letu. Navrhl Ornitopteru, předchůdce dnešní helikoptéry, která nebyla nikdy realizována a položil tak základy při pozdější konstrukci helikoptér a letadel. Pokud přeskočíme další pokusy člověka létat a zaměříme se na první řízený motorový let, dostáváme se automaticky na přelom 19. a 20. století, kdy bratři Orville a Wilbur Wrightové sestrojili v roce 1902 kluzák, následně motor o výkonu 12-ti koňských sil a v roce 1903 uskutečnili první let letadla s vlastním motorovým pohonem. Tato událost tak položila základy následnému motorovému létání. Beze sporu největší vývoj techniky se děje v období války. Během 1. a 2. světové války se letadla stala nepostradatelnou součástí boje a strategickou výhodou. Postupem času se zlepšují materiály, začíná se vyvíjet elektronika a v roce 1939 vzlétne v Německu Heinkel He 178, první letadlo poháněné proudovým motorem. Od té doby se proudový motor stane jedním z nejpoužívanějších typů pohonné jednotky v dopravní a vojenské letecké oblasti.

Při sledování této linie vývoje nesmíme přehlédnout modelářství jako koníčka, neboť hrál nepostradatelnou roli při vývoji funkčních a letu schopných vzorků. Vývojem nových materiálů, elektroniky a jejich dostupností běžnému uživateli, se ruku v ruce vyvíjí i letecké modelářství, které tak může sledovat moderní trend vývoje skutečných letadel. Od tvarů letadel, použitých profilů, pístových pohonných jednotek, až k myšlence sestrojit malý proudový motor. Proudový motor se tak stává i v této oblasti oblíbenou pohonnou jednotkou letadel či helikoptér, kde umocňuje realističtější charakter letu.

První modelářská turbína na světě byla zkonstruována v roce 1983 (Jerry Jackman), v Čechách se o první prototyp postaral Petr Stejskal, který začal v roce 1996 stavět první amatérskou turbínu a v roce 1998 byla funkční nasazena do modelu letadla.

Diplomová práce se zabývá konstrukcí a realizací řídicí jednotky malého proudového motoru, přičemž navazuje a vychází z výsledků získaných v bakalářské a diplomové práci Miroslava Hájka, která byla zaměřena na identifikaci proudového motoru a vytvoření vizualizačního SW pro zpracování telemetrických dat z turbíny. Dále pak využívá výsledky získané z bakalářské práce autora (Vosecký, M., 2008), která se zabývala vhodným výběrem řídicí jednotky a návrhem HW pro zpracování měřených signálů.

Druhá kapitola se zabývá obecným principem funkce, vstupy / výstupy a pracovními režimy turbíny. Řídicí jednotky se jak pro pístové, tak i pro proudové motory v letecké praxi souhrnně označují pod zkratkou FADEC[1]. Jedná se tedy o jednotky, na které jsou kladeny vysoké nároky z hlediska bezpečnosti, spolehlivost a správnosti chodu. Popis a návrh této jednotky je popsán ve třetí kapitole, kde jsou uvedeny a popsány všechny důležité termíny, dále je zde popsán výběr procesorové jednotky, senzorika a návrh vstupně / výstupní desky, která slouží pro úpravu měřených signálů a oddělení výkonových částí systému. Ve čtvrté kapitole je popsán komunikační protokol, který byl navržen v diplomové práci (Hájek, M., 2007) rovněž jako vizualizační SW (DataReader). Protože vizualizační SW a řídicí jednotka nebyly vyvíjeny souběžně, aby tak mohly být testovány proti sobě, vzniklo během vývoje vizualizačního SW řada chyb, které musely být odstraněny. Pátá kapitola se tedy zabývá stručným popisem nutných úprav vizualizačního SW, které musely být provedeny pro jeho bezproblémovou a správnou funkčnost. Šestá kapitola se zabývá rozborem a možnostmi implementace jednotlivých stavových automatů, které jsou programově implementovány v řídicí jednotce. Při návrhu těchto automatů a vhodné implementace byl jeden z hlavních požadavků snadná modifikace. V osmé kapitole je popsán používaný PID regulátor pro regulaci otáček turbíny (Špinka, O., 2009) a jednotlivé implementované režimy chodu řídicí jednotky.

[1] Full Authority Digital Engine Control

Kapitola 2

Modelářská turbína

Proudové modelářské motory nebyly do nedávna běžnou pohonnou jednotkou. Je to způsobeno tím, že ačkoli princip funkce je jednoduchý a neliší se příliš od skutečných turbín, jsou modelářské turbíny náročné na přesnost zpracování, použité materiály a řídicí elektroniku. Jednou z odlišností modelářské turbíny oproti skutečným je to, že na start se jako palivo používá propan-butan a že se zmenšující velikostí turbíny se zvyšuje potřebný počet otáček (Vosecký, M., 2008).

2.1 Vytváření tahu motoru

Proudový motor je ve své podstatě stále jen tepelný motor, jehož pracovní cyklus přibližně odpovídá klasickému cyklu čtyřdobého motoru. Fáze pracovního cyklu jsou stejné:

- sání
- komprese
- zapálení
- expanze

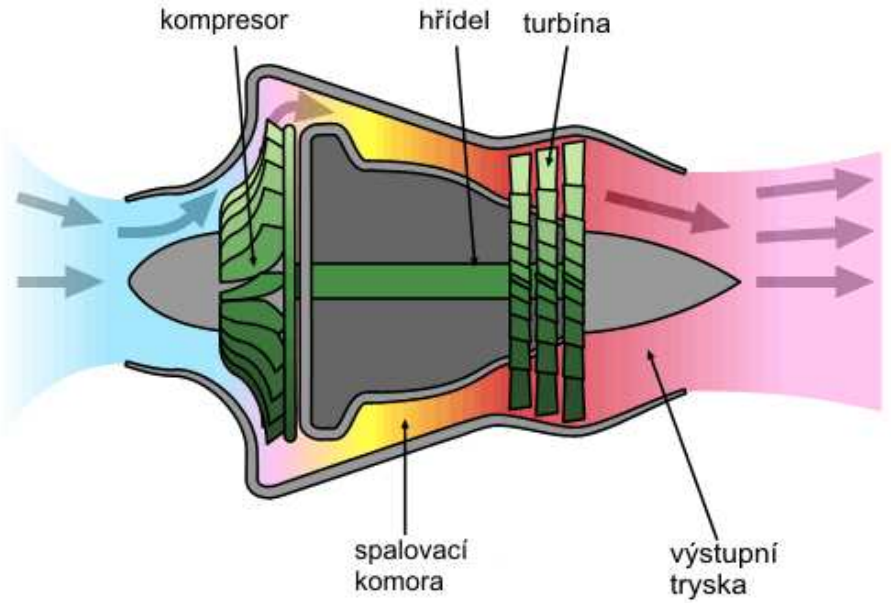
Proudové motory mají ale tu výhodu, že všechny fáze probíhají najednou, a tak motor dává při stejné velikosti více výkonu než motor pístový.
Vše se opírá o Newtonův zákon akce a reakce - na každou akci existuje protiakce (reakce). V případě pohonu letadel je akcí nasávaný vzduch do motoru a jeho rychlé vyfouknutí zadní částí. Reakcí je potom síla tlačící motor vpřed. Tah je vyvozen urychlením vzduchu,

který motorem prochází. Jde tedy o rozdíl hybnosti vzduchu z motoru vystupujícího a hybnosti vzduchu do motoru vstupujícího dle 2.1.

$$F = Q(v_{out} - v_{in}) \tag{2.1}$$

$F\,[N]$ - výsledná síla, $Q\,[kg/s]$ - průtočné množství za jednotku času, v_{out} a v_{in} - rychlost výstupních a vstupních plynů.

Schematicky je to znázorněno na obrázku 2.1

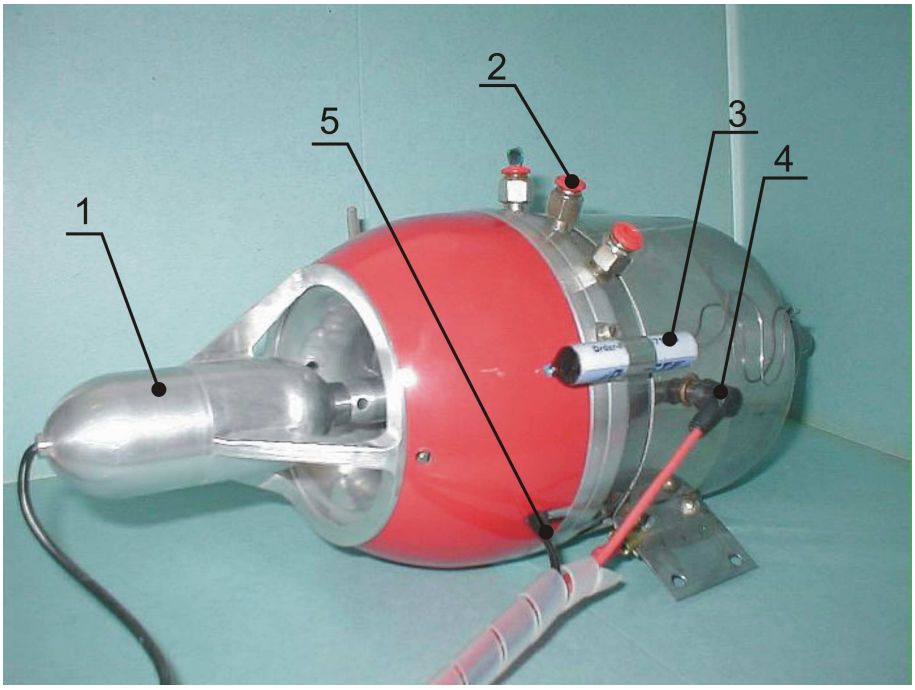


Obrázek 2.1: Principelní schéma turbíny

Vzduch je nasáván vstupním otvorem do odstředivého (radiálního) kompresoru, kde je stlačen až na tlak 100 kPa. Usměrněný a zpomalený proud stlačeného vzduchu vstupuje do spalovací komory. Ta je tvořena pláštěm motoru. Zde se míchá tzv. primární vzduch s rozprášeným palivem z palivových trysek a dochází k hoření při teplotě až 2000°C. Výstupní žhavé plyny dopadají pod správným úhlem na točící se lopatky axiální turbíny, která přes hřídel pohání kolo kompresoru. Pracovní otáčky se pohybují okolo 100 tisíc za minutu. Za turbínou mají plyny o teplotě 500 až 600°C ještě dostatek energie k vyvození reakčního tahu (Stejskal, P., 2003).

2.2 Vstupy / výstupy turbíny

Vstupem turbíny jsou palivové trysky sloužící pro přívod paliva (kerosin, propan-butan) do spalovací komory. Dále je to žhavící koncovka pro žhavící svíčku, která slouží k počátečnímu zapálení plamene uvnitř spalovací komory. Dalším vstupem je startovací motor, který slouží během startovacího procesu k počátečnímu roztočení rotoru turbíny a při nevydařeném startu k vyfoukání propan-butanu nahromaděném ve spalovací komoře. To je obzvláště důležité, jinak by mohlo dojít k explozi turbíny. Startovací motor se používá také po zastavení turbíny k chlazení. Výstupem turbíny je dopředný tah a měřené signály: teplota výstupních plynů a otáčky turbíny. Vše je vidět na následujícím obrázku 2.2



Obrázek 2.2: Vyznačení vstupních / výstupních částí turbíny

1. Startovací motor
2. Palivové trysky
3. Senzor teploty
4. Žhavící koncovka
5. Senzor otáček

2.3 Provozní režimy

V následující tabulce 2.1 jsou uvedeny hodnoty otáček turbíny, které je z hlediska bezpečnosti nutné dodržet.

Pracovní režim	**Otáčky [ot/min]**
Volnoběh	25 000 - 30 000
Norm. provozní režim	30 000 - 80 000
Max. výkon	120 000

Tabulka 2.1: Hodnoty otáček v jednotlivých pracovních režimech

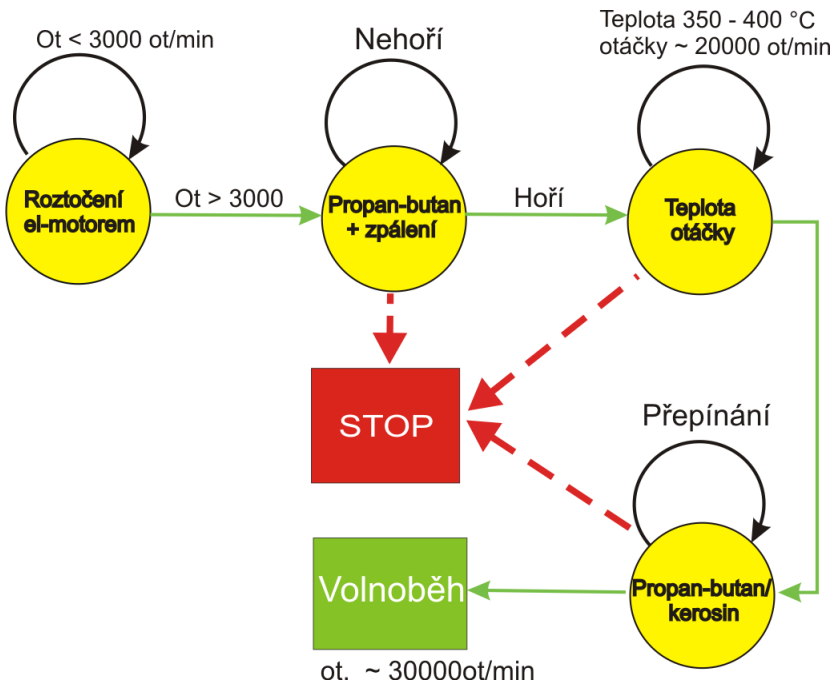
Hodnoty výstupních plynů turbíny se pohybují v rozmezí 450°C - 800°C, maximální hodnota je cca 1000°C.

2.3.1 Start turbíny

Blokově je průběh startu znázorněn na obrázku 2.3. Start se provádí po roztočení turbíny startovacím motorem přibližně na 3000 ot/min. Poté je otevřen ventil s propan-butanem a za pomoci žhavící svíčky se provede zapálení plynu ve spalovací komoře. Pokud se zapálení po určitém čase nepodaří, je proces zastaven, startovacím motorem se vyfouká přebytečný plyn ve spalovací komoře a začne se od začátku. Zapálení se detekuje nárůstem teploty výstupních plynů a otáček. Při podařeném zápalu běží turbína na propan-butan, dokud se teplota nepohybuje mezi 350 - 400°C a otáčky okolo 15 000 ot/min. Poté se může za stálého sledování teploty a otáček provést přepnutí z propan-butanu na kerosin. Přepnutí je pozvolné, kdy se uzavírá ventil na propan-butan a otevírá na kerosin, přičemž teplota by měla stoupat k 500°C a otáčky k 25 000 - 35 000 ot/min. Pokud je detekována jakákoliv chyba, je proces zastaven. Při úspěšném přepnutí běží turbína ve volnoběžných otáčkách.

2.3.2 Běh turbíny

Výkon turbíny se po úspěšném startovacím procesu ovládá přísunem paliva. Přísun paliva musí být ale pozvolný a musí odpovídat danému pracovnímu režimu. Například pokud bude přísun paliva moc veliký, palivo se nestačí spálit ve spalovací komoře a plameny začnou šlehat daleko z výstupní trysky nebo palivo plamen uhasí. Naopak pokud bude



Obrázek 2.3: Stavový diagram startu turbíny

množství paliva vzhledem k proudícímu vzduchu nedostatečné, proudící vzduch plamen "sfoukne".

2.3.3 Zastavení turbíny

Protože otáčky jsou řízeny přísunem paliva, zastavení turbíny se provádí postupným snižováním přísunu paliva.

Kapitola 3

Řídicí jednotka

Parametry turbíny : maximální otáčky okolo 100 000 ot/min, teplota výstupních plynů okolo 700°C a výkon 5 kW činí turbínu nebezpečným zařízením. Proto jsou kladeny vysoké nároky na řídicí jednotku. V leteckém průmyslu se elektronické řídicí jednotky označují pod zkratkou - FADEC. V následující kapitole bude popsána koncepce návrhu řídicí jednotky, která se opírá o výsledky získané z bakalářské práce (Vosecký, M., 2008).

3.1 FADEC

Řídicí jednotka (FADEC) se skládá z řídicího počítače (ECU[1]) a nutného rozhraní pro ovládací prvky motoru. Podstatou systému FADEC je zpracování pohybu plynové páky a jiných ovládacích prvků motoru v počítačové jednotce a až ta optimální způsobem řídí samotný motor. Požadavky na modularitu a snadnou rekonfigurovatelnost řídicích systémů vedly k návrhu tříúrovňové struktury, kterou tvoří blok řídicí elektroniky, senzorická část a výkonová část. Toto rozdělení do tří logických celků usnadňuje přestavitelnost podle aktuálních požadavků úlohy a také poskytuje rychlou možnost opravy některé z vadných částí (Švéda, M. a Hubík, Vl., 2008).

Řídicí elektronika (ECU) analyzuje signály ze senzorického bloku a na základě výsledků řídí motor podle požadovaného algoritmu.

Senzorická část řídicího systému má za úkol snímat signály, filtrovat, zesilovat a upravovat tak, aby mohly být zpracovány řídicím počítačem.

Výkonová část se používá k řízené dodávce elektrické energie do koncového zařízení

[1] Electronic Control Unit

(čerpadlo, elektrický motor, ...). Protože tato část ovládá výkonové prvky, prvky pracující na jiných úrovních napětí, musí být navržena tak, aby nerušila ostatní části řídicího systému.

Výhody jednotky jsou:

- ochrana motoru v situacích spadající mimo jeho bezpečný pracovní režim
- změna chování motoru přeprogramováním jednotky
- monitoring a diagnostika

Při návrhu architektury řídicí jednotky je třeba brát v potaz následující problematiku (Pačes, P., 2008):

- typ mikroprocesoru (rychlost, paměť, periferie)
- mechanismus detekce chyb
- náročnost programování
- vhodné rozhraní pro vývoj a testování
- měření, filtrování
- binární reprezentace čísel
- A / D konverze, vzorkování

Lze tedy vidět, že se jedná o komplexní problematiku zahrnující v sobě veškeré dostupné technologie a metody z měřící a řídicí techniky.

3.1.1 Popis vstupů a výstupů navrhované řídicí jednotky FADEC

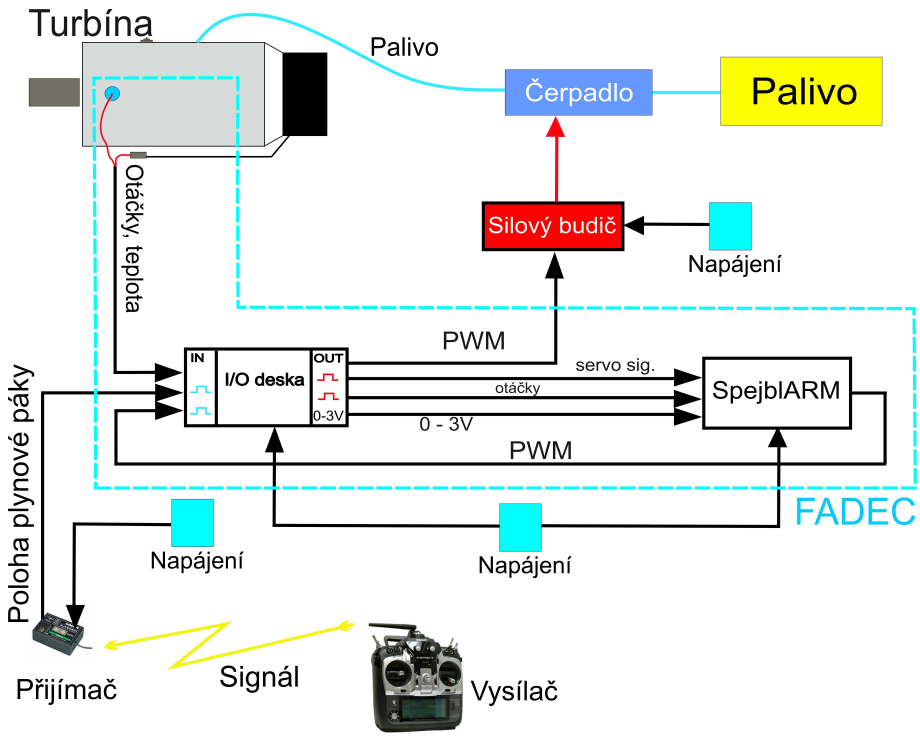
Na obrázku 3.1 je znázorněno blokově propojení řídicí jednotky se všemi vstupními a výstupními signály. Vstupními signály řídicí jednotky jsou:

- poloha plynové páky z modelářského přijímače (obrázek 3.5)
- teplota výstupních plynů turbíny
- otáčky turbíny

Výstupní signál tvoří:

- PWM signál generovaný řídicí jednotkou do silového budiče, který ovládá palivové čerpadlo opět PWM signálem

V současné době je návrh prováděn pro jeden výstupní signál - generování PWM pro silový budič JES006 (JETI MODEL, 2009).



Obrázek 3.1: Blokové schéma řídicí jednotky s připojenými periferiemi

3.1.2 Senzorika a připojené periferie řídicí jednotky

V následující části jsou popsány jednotlivé připojené senzory k řídicí jednotce:

- senzor pro měření otáček
- senzor teploty

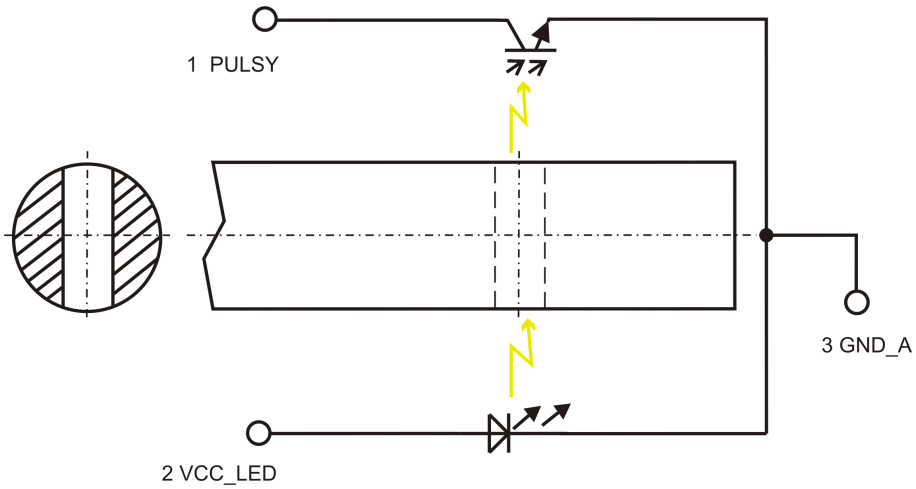
a jednotlivé připojené periferie:

- modelářský vysílač / přijímač
- silový budič JES006
- startovací elektromotor

3.1.2.1 Senzor otáček

Senzor otáček je tvořen infračervenou LED diodou a fototranzistorem. Na obrázku 3.2 je schematicky naznačeno měření otáček. V hřídeli turbíny je v podélné ose vyvrtán jeden otvor. V ose tohoto otvoru je umístěna na jedné straně infračervená dioda a na protější straně fototranzistor. Tranzistor je tedy osvícen každou otáčku dvakrát.

Při vysokých otáčkách turbíny (přibližně 100 000 ot/min) je úroveň signálu dopadající na fototranzistor malá, z tohoto důvodu se tranzistor nedokáže zcela plně otevřít a zároveň se nedokáže zcela zavřít. Tím pádem se objevuje v měřeném signálu stejnosměrná složka, která roste se vzrůstajícími otáčkami a zároveň snímaný signál modulovaný okolo stejnosměrné složky zdaleka není pravoúhlý. Je tedy požadavek na návrh obvodu, který odstraní stejnosměrnou složku a zároveň upraví signál na obdélníkový průběh o stejné frekvenci jako je frekvence spínání. Pokud by jinak takto neupravený signál byl připojen na vstup procesoru, nebyl by vůbec měřitelný.

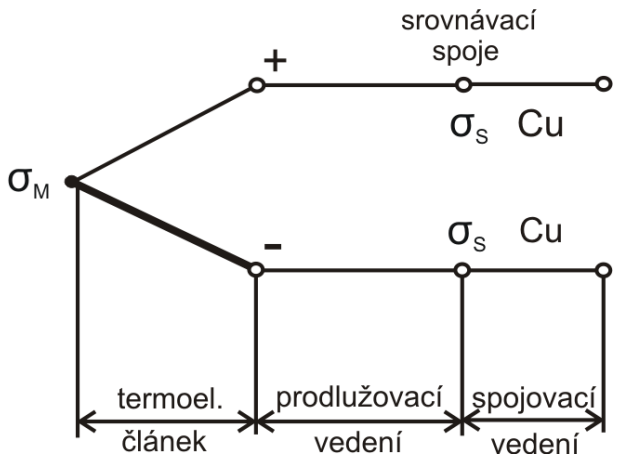


Obrázek 3.2: Princip měření otáček

3.1.2.2 Senzor teploty

Jako senzor teploty je použit termoelektrický senzor teploty (termočlánek) typu K, jehož měřící konec je umístěn na plášti výstupní trysky. To je dobře patrné na obrázku 2.2. Při měření teploty spojem σ_M je nutné zaručit stejnou teplotu na obou srovnávacích spojích σ_S. Senzor teploty je zapojen pomocí dvouvodičového zapojení (obrázek 3.3). Při použití tohoto připojení je chyba měření nejvíce ovlivněna odporem přívodů k senzoru a vlivem kolísání srovnávacích teplot. Vliv kolísání teploty srovnávacích spojů lze odstranit pomocí prodlužovacího vedení, kdy se posunou srovnávací spoje do míst, kde je teplota

konstantní, nebo pomocí kompenzační krabice (Ripka, P. et al., 2005). Protože ale není potřeba měřit teplotu s tak velikou přesností, odchylka například ±10°C je stále přijatelná, je dvouvodičové zapojení senzoru plně postačující.



Obrázek 3.3: Dvouvodičové připojení senzoru teploty

3.1.2.3 Modelářský vysílač, přijímač

Modelářsky vysílač je zařízení, pomocí kterého pilot vysílá signál na nosné frekvenci do modelářského přijímače, který ovládá konkrétní ovládací prvky na letadle, motoru a jiných částí. Podle modulace rozlišujeme dva základní typy:

PPM - (Pulzně Polohová Modulace) generuje signál ve tvaru obdélníků (PWM, obr. 3.5), sled obdélníků za sebou definuje kanál, pro který je daný obdélník určen (první obdélník první kanál, druhý obdélník druhý kanál . . .). Velikost (délka) obdélníku určuje, v jaké poloze je daný ovládací prvek. Pokud se během přenosu mezi vysílačem a přijímačem vlivem rušení nebo jiné chyby nějaký obdélník poškodí, nebo úplně ztratí, dojde k chybnému „přečtení“ pokynu v přijímači a tím i k chybnému nastavení výchylky serva. PPM vysílač vysílají na kmitočtu 35MHz nebo 40MHz.

PCM - (Pulzně Kódová Modulace) modulace přenáší v binární podobě přímo hodnoty, které definují kód kanálu a krok (polohu), ve kterém se nachází poloha ovládacího prvku. PCM vysílače vysílají na kmitočtu 2.4GHz, 35MHz nebo 40MHz.

Modelářsky přijímač funguje jako přijímací strana vysílače. Zpracovává tedy přijatý signál a ovládá tak koncová zařízení.

3.1.2.4 Startovací elektromotor

Za startovací elektromotor lze použít jakýkoliv modelářský elektromotor splňující požadavky:

- dostatečný výkon pro počáteční roztočení turbíny
- vhodné napájecí napětí

v aplikaci je použit stejnosměrný motor s feritovými magnety *Speed 280* (Graupner, 2009) se základními parametry dle tabulky 3.1

Max. proud	1.58A
Napájecí napětí	0 - 6V

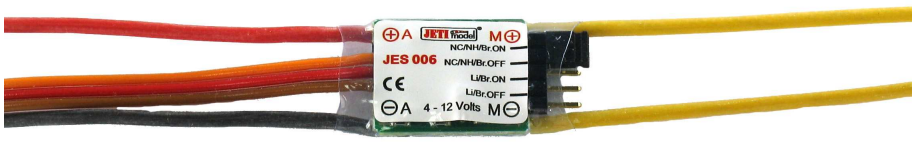
Tabulka 3.1: Základní technické údaje motoru Speed 280

3.1.2.5 Silový budič

Jako silový budič je použít modelářský regulátor JES006 (obrázek 3.4) . Jedná se o stejnosměrný regulátor s parametry dle následující tabulky:

Trvalý proud	6A
Špičkový proud (max 30s)	8A
Počet článků NiCd	4 - 8
Min. nap. napětí	4V
Max. nap. napětí	12V

Tabulka 3.2: Základní technické údaje regulátoru JES006

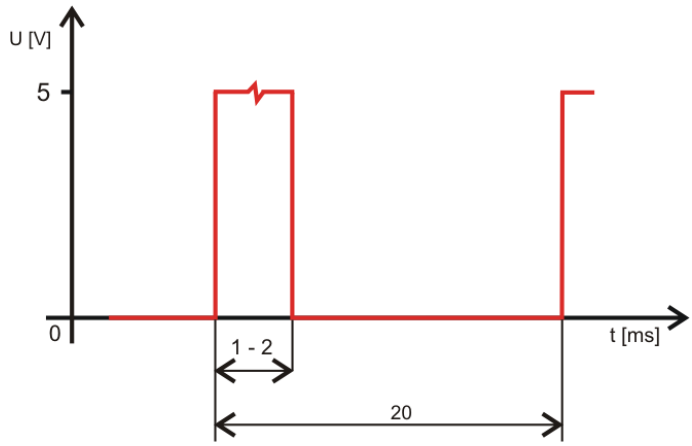


Obrázek 3.4: Modelářský regulátor JES006

3.1.3 Princip řízení modelářské turbíny

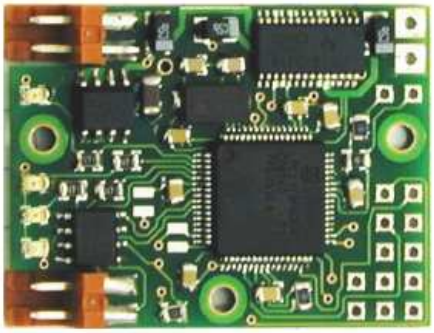
Pro řízení turbíny je třeba sledovat absolutní hodnotu otáček a dle tohoto údaje pak dále ovládat palivové čerpadlo. Palivové čerpadlo je řízené pomocí silového budiče (obrázek 3.1), který je reprezentován modelářským regulátorem JES006. Tento silový budič generuje výstupní PWM signál, který tak ovládá otáčky čerpadla.
Silový budič je řízen řídicí jednotkou, která generuje PWM signál na základě polohy plynové páky a počtu otáček. Řídicí signál silového budiče je stejný jako klasický servo signál modelářského přijímače (obrázek 3.5).

Servo signál je TTL signál o periodě *20 ms*, kde aktivní doba v logické „1“ se pohybuje v rozmezí *1 - 2 ms* a to dle polohy páky na vysílači.



Obrázek 3.5: Ukázka servo signálu

Pokud přijde požadavek na zvýšení / snížení výkonu (otáček) turbíny, je třeba k tomu úměrně generovat PWM signál pro silový budič. Řídicí jednotka generuje příslušný PWM signál za stálého sledování hodnoty otáček. Optimální regulace je taková, kdy je přísun palivové směsi nastaven tak, že změna k žádanému výkonu probíhá za konstantní teploty výstupních plynů (Vosecký, M., 2008).



Obrázek 3.6: Řídicí počítač SpejblARM

3.2 Procesorová jednotka

Jako procesorová jednotka byla v bakalářské práci (Vosecký, M., 2008) zvolena deska SpejblARM (obrázek 3.6) s procesorem lpc2119 a jádrem ARM7TDMI (Philips, 2004). Jednotka byla vybrána z důvodu existujícího ověřeného zapojení a z důvodu dobré podpory vývojových nástrojů.

3.2.1 Charakteristika procesorové jednotky

Základem desky je jednočipový mikroprocesor lpc2119 s jádrem ARM. Jádro ARM7TDMI provádí výpočty v 32-bitové aritmetice. To dává desce dostatečný výpočetní výkon pro implementaci potřebných stavových automatů řízení, komunikace a měření. Zavádění a ladění programu je prováděno po vyvedené asynchronní sériové lince s použitím vnitřního ISP[2] zavaděče. K vynucení spouštění ISP zavaděče je vyveden signál ISPSEL.

Na destičce je stabilizováno napětí 5V pro budič CAN PCA82C250, dále pak 3.3V a 1.8V pro LPC2119. Stabilizaci 3.3V a 1.8V provádí spolu s vytvářením signálu RESET kombinovaný obvod TPS73HD318. Obě napájení jsou větvena mezi analogovou a číslicovou část (Peca, M., 2008).

Parametry jednotky:

- mikroprocesor s 32b jádrem ARM7TDMI
- maximální frekvence CPU 60MHz

[2]ISP - In-System Programming

- 16kB on-chip Static RAM a 128kB on-chip Flash
- In-System Programming (ISP)
- 2xUART rozhraní, 2xSPI a rychlé I^2C rozhraní
- 2x32b timery a 6 PWM výstupů
- 8-kanálový 10b AD převodník
- napájení CPU od 1.65V do 1.95V, tj.1.8V ±8.3%
- vstupy / výstupy v 3.3 V logice 5V tolerantních

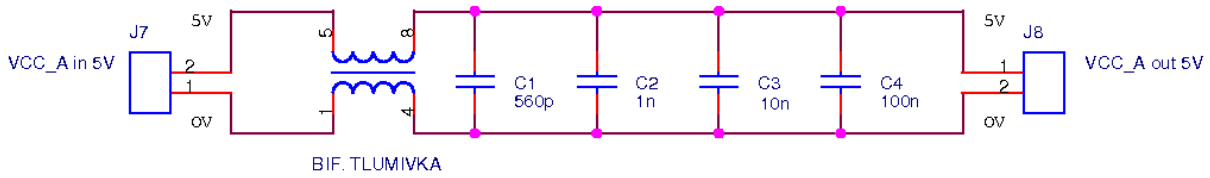
3.3 Deska vstupů / výstupů

Kvalita signálů ze snímačů není příliš dobrá, proto je nutné každý měřený signál upravit a až po té přivést ke zpracování řídicím počítačem. Dále je nutné galvanicky oddělit jednotlivé prvky, aby nedocházelo k potencionální vzájemné interferenci mezi digitálními a analogovými částmi. Deska vstupů / výstupů byla navržena v bakalářské práci (Vosecký, M., 2008) a v následující části bude uveden její popis s výsledky laboratorních testů.

3.3.1 Napájecí obvod desky vstupů / výstupů

Z obrázku 3.1 je patrné, že zdroj napájení je pro řídicí desku (SpejblARM) a I / O desku stejný. V řídicí desce je pak stabilizovaným napájením 5V (IO LE50CD[3]) napájen integrovaný obvod TPS73HD318, který vytváří 3.3V a 1.8V pro mikrokontrolér LPC2119 (Peca, M., 2008). Pro I / O desku je napájecí část řešena tak, že do napájení je zařazen napájecí filtr (obrázek 3.7). Filtr se skládá z bifilární tlumivky a sady kapacitorů. Princip spočívá v tom, že bifilární tlumivka se pro střídavý signál chová jako zkrat (propustí stejnosměrný) a sada kondenzátorů slouží jako filtr na odstranění špiček v napájení. Jádro tlumivky je železo-prachové, na kterém je navinuto 17 závitů.

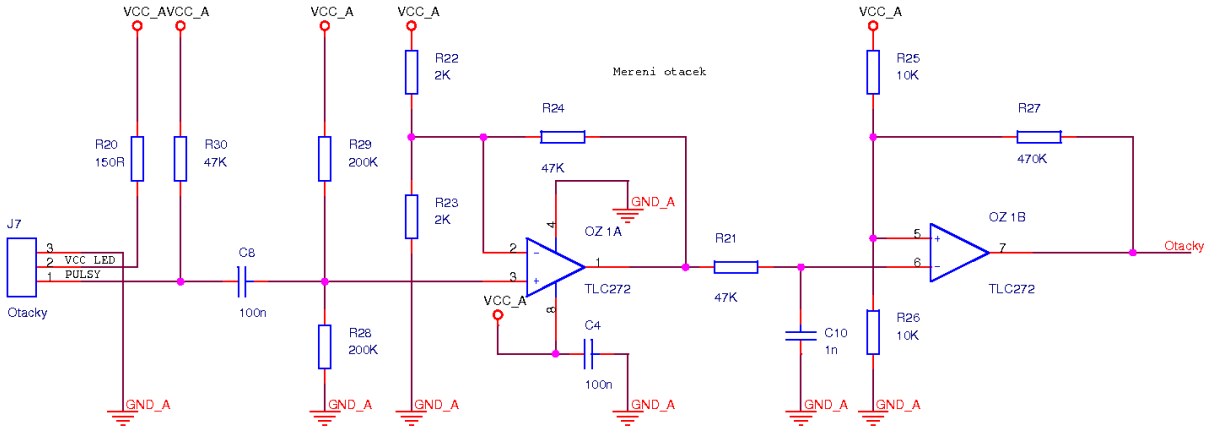
[3]stabilizátor napětí, výstupní napětí je 5V



Obrázek 3.7: Napájecí obvod I/O desky

3.3.2 Měřící obvod senzoru otáček

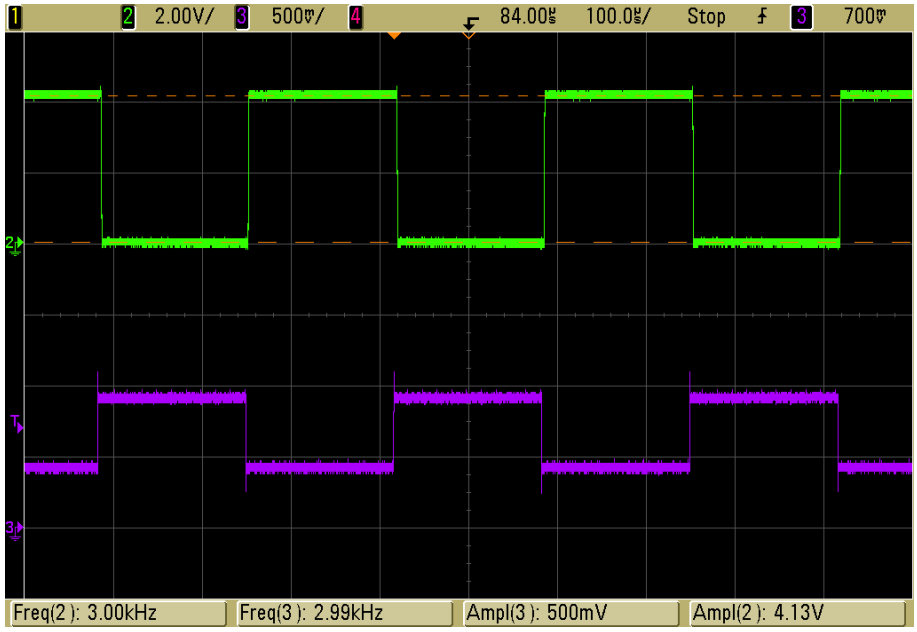
Senzor otáček je připojen k měřícímu obvodu (obrázek 3.8) tak, že kolektor tranzistoru je připojen na PIN 1 konektoru J7. Tento pin je připojen také přes R30 (pull-up rezistor) k VCC. Operační zesilovač OZ1A je zapojen jako neinvertující zesilovač s napěťovým zesílením A = 49, za zesilovacím členem se nachází operační zesilovač OZ1B zapojený jako komparátor s hysterezí. Na výstupu tohoto komparátoru je obdélníkový signál o stejné frekvenci, jako je frekvence spínání fototranzistoru a amplitudě 5V. Tento výstup je pak následně připojen přímo na vstup registru CAP0.3[4].



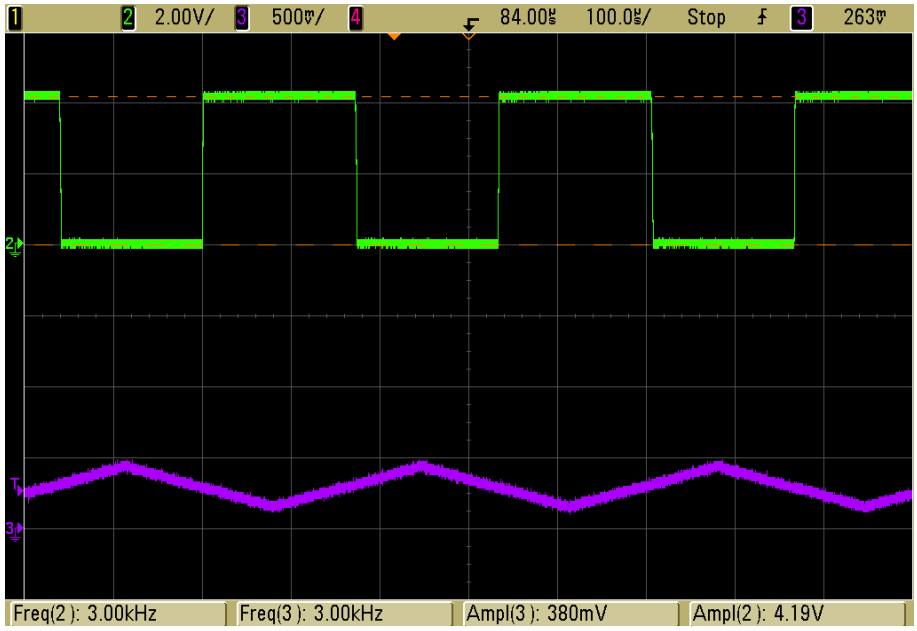
Obrázek 3.8: Měřící obvod pro senzor otáček

Správnost návrhu obvodu a výsledky jsou na obrázcích 3.9, 3.10 a 3.11, kde jako vstupní emulovaný signál z kolektoru tranzistoru byl volen postupně obdélníkový (ampl = 500mV, f = 2.99KHz), pilový (ampl = 380mV, f = 3.00KHz) a sinusový (ampl = 1.25V, f = 3.00KHz) signál. Z průběhů lze vidět že obvod správně zesiluje a tvaruje signál i při malé amplitudě vstupního signál, vyšší frekvenci (řádově KHz) a stejnosměrné složce.

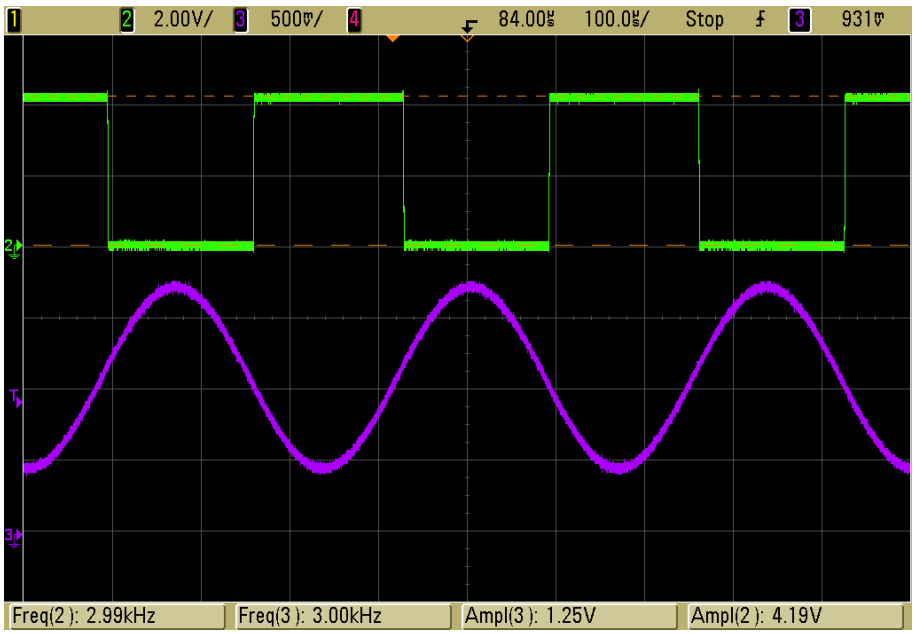
[4]CAP0.3 - záchytný vstupní registr mikrokontroléru LPC2119



Obrázek 3.9: Vstupní obdélníkový signál a výstupní (upravený) signál obvodu otáček



Obrázek 3.10: Vstupní pilový signál a výstupní (upravený) signál obvodu otáček



Obrázek 3.11: Vstupní sinusový signál a výstupní (upravený) signál obvodu otáček

3.3.3 Měřící obvod senzoru teploty

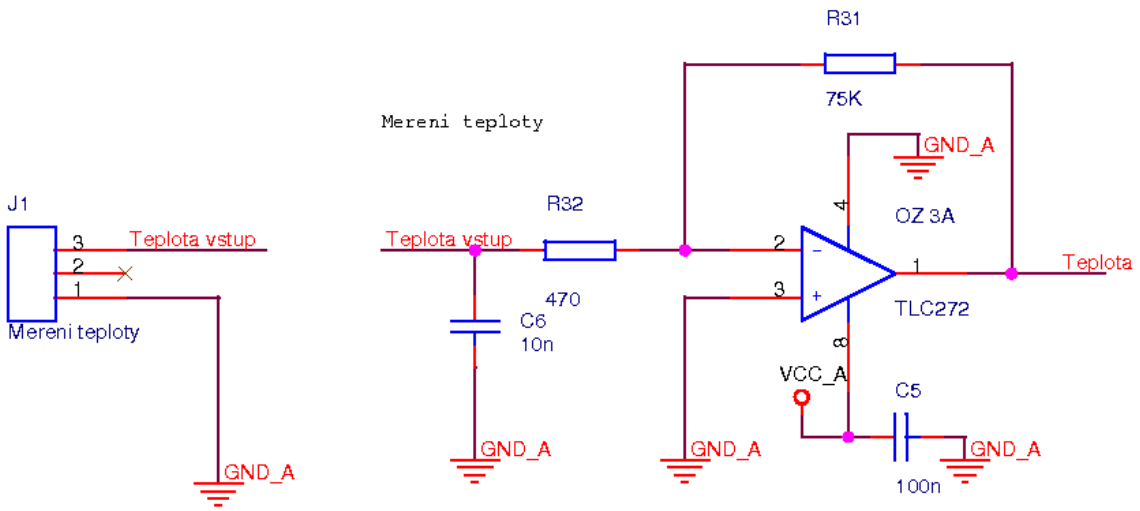
Měřící obvod pro měření teploty z termočlánku je na obrázku 3.12. Obvod se skládá z operačního zesilovače OZ3A, který je zapojen jako invertující zesilovač s napěťovým zesílením A = 96. Zesílení je voleno tak veliké, protože ze senzoru je zesilováno napětí o hodnotách řádově desítky mV. Senzor se připojí na konektor J1 na piny 3 a 1. Výstup zesilovače je připojen na vstupní pin mikroprocesoru LPC2119 AIN0[5] .

3.3.4 Galvanické oddělení řídicích signálů

Galvanickým oddělením RC přijímače od řídicí jednotky a silového budiče od zbytku elektroniky eliminujeme rušivé vlivy jednotlivých částí v napájení. Galvanicky jsou odděleny signály:

- signál z modelářského přijímače (obr.3.5) - poloha plynové páky (signál přiveden na vstup řídicí jednotky)
- signál PWM generovaný řídicí jednotkou (obr.3.5) - PWM signál (signál přiveden na vstup silového budiče)

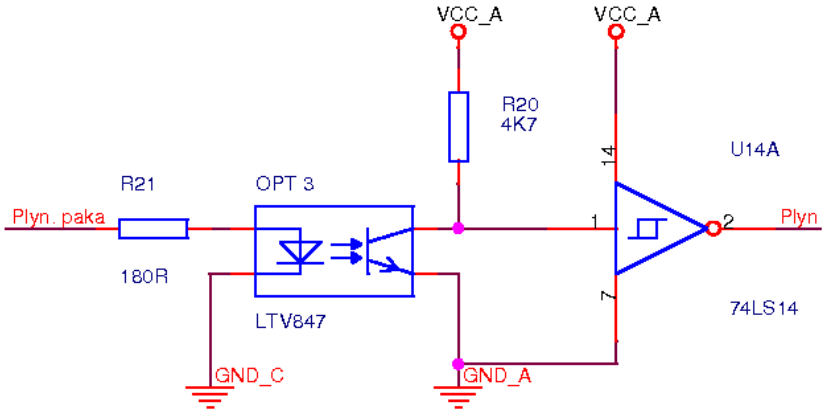
[5] AIN0 - vstupní pin registru PINSEL1



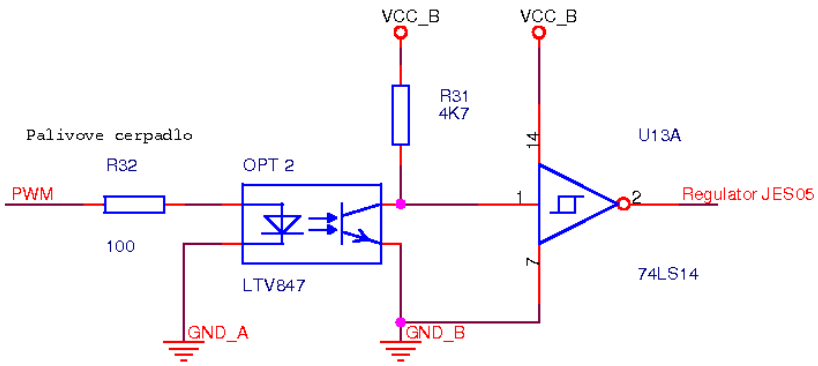
Obrázek 3.12: Měřící obvod pro senzor teploty

3.3.4.1 Zapojení galvanického oddělení

Na obrázcích 3.13 a 3.14 jsou jednotlivá schémata zapojení pro oddělení signálů. Pro tvarování výstupního signálu je pouľit Schmittův klopný obvod.



Obrázek 3.13: Galvanické oddělení polohy plynové páky



Obrázek 3.14: Galvanické oddělení PWM

Kapitola 4

Komunikační protokol

Mezi komunikací řídicí jednotky a nadřazeným vizualizačním SW byl navržen v diplomové práci (Hájek, M., 2007) komunikační protokol, který jako přenosové medium využívá sériovou linku. Tím se řídicí jednotka stává plně diagnostikovatelnou a konfigurovatelnou. Během vývoje bylo zjištěno, že je potřeba doplnit protokol o nové funkčnosti. Všechny tyto změny jsou popsány níže.

4.1 Seriová linka

Ke komunikace mezi vizualizačním SW a řídicí jednotkou se používá asynchronní sériový přenos. V následující tabulce 4.1 jsou uvedeny jednotlivé parametry pro nastavení sériové linky.

Počet dat. bitů	8
Rychlost komunikace	38 400 bps
Parita	None
Stop bity	1
Řízení toku	None

Tabulka 4.1: Základní nastavení sériového portu

4.2 Navržený komunikační protokol

Posílané zprávy jsou děleny do dvou základních typů:

- příkazy - nenesou žádá data, příklad viz. tabulka příkazů 4.2
- zprávy - nesou data, příklad viz. tabulka zpráv 4.3

Zprávy jsou posílány jako posloupnost znaků, kde čísla jsou reprezentována jako řetězec hexadecimálních číslic, ve formátu *Big Endian*. Obecný formát dat je:

```
<start byte><délka zprávy><tělo zprávy><XOR>
```

Jako start byte se používá znak @, čímž se jednoznačně odliší příchod další zprávy. Dále následuje *délka zprávy* o velikosti 2 byte. Pro ověření správnosti příchozích dat se používá příčná parita přes celou zprávu. XOR je umístěn jako poslední byte zprávy. Prostřední část zprávy, *tělo zprávy*, se liší pro příkazy a pro zprávy.

Příkazy jsou zprávy, které nenesou žádná data. Jejich formát je:

@	0	3	**C**	**Id**	XOR

za délkou zprávy následuje řídící znak *C*, který jednoznačně identifikuje zprávu jako příkaz. Dále pak následuje *Id* příkazu dle tabulky příkazů 4.2.

Zprávy na rozdíl od příkazů neobsahují za délkou zprávy identifikátor *C*, ale konkrétní *Id* zprávy (viz tabulka 4.3). Formát má tedy tvar:

@	0	X	**Id**	**DATA**	XOR

Zprávy tedy obsahují část DATA. Význam a popis jednotlivých typů zpráv je popsán v příloze.

4.3 Průběh komunikace

Na obrázku 4.1 je znázorněn sequence diagram[1] komunikace mezi řídicí jednotkou a nadřazeným SW. Řídicí jednotka posílá neustále telemetrická data (viz tabulka 4.3 a příloha) přičemž vizualizační SW čeká na jejich první příchod. Po prvním příchodu vizualizační SW odešle požadavek na nastavení vzorkovací periody[2]. Po odeslání čeká PC

[1] znázorňuje, jak jednotlivé objekty / systémy komunikují mezi s sebou v čase

[2] vzorkovací perioda se nastavuje v menu vizualizačního SW

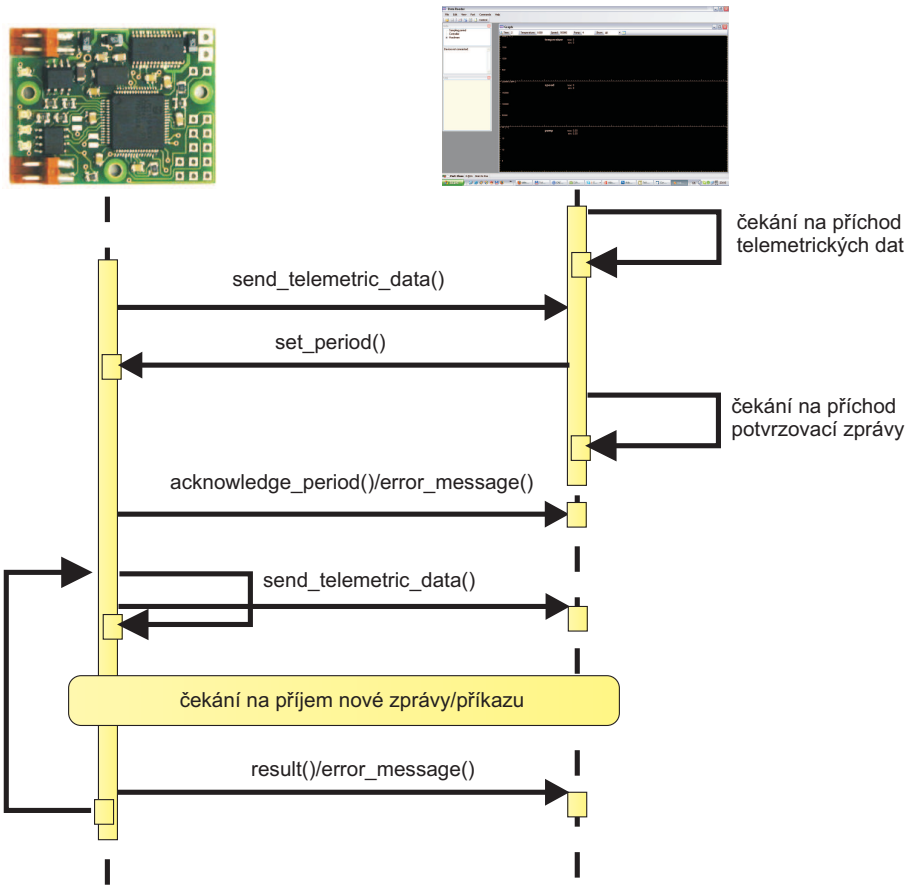
Příkaz	**Id**	**Popis**
regulátor	R	Řídicí deska pošle aktuální nastavení regulátoru
program	A	Předá řízení programu
manual	B	Přepne řízení na manuální
control	F	Přepne na řízení pomocí PID regulátoru
chyba	E	Chybový kód přijaté zprávy/příkazu
potvrzení	O	Potvrzení přijaté zprávy/příkazu
stop	C	Zastaví turbínu
vz. perioda	S	Odešle do PC nastavenou vz. periodu

Tabulka 4.2: Tabulka příkazů

Zpráva	**Id**	**Popis**
data	V	Řídicí jednotka pošle telemetrická data do PC
regálátor	R	PC pošle nastavení regulátor
skok	J	Řídicí jednotka provede skok o definované výšce
rampa nahoru	U	Rampa nahoru
rampa dolu	D	Rampa dolu
vz. perioda	P	Nastaví vzorkovací periodu
otáčky	S	Otáčky
mezní hodnoty	L	Mezní hodnoty otáček reference

Tabulka 4.3: Tabulka zpráv

na příchod potvrzení o správnosti přijetí, nebo o přijetí chybové zprávy. Pokud řídicí jednotka pošle do PC chybovou zprávu, opakuje se opět požadavek na nastavení vzorkovací periody. Když PC přijme potvrzení o přijetí vzorkovací periody, řídicí jednotka dále posílá telemetrická data, dokud nepřijde požadavek ze strany PC. Po každém odeslání požadavku ze strany PC, čeká vizualizační SW na příchod potvrzovací / chybové zprávy stanovený časový limit. Pokud do té doby nepřijde odezva, opakuje se odeslání požadavku, což se děje maximálně dvakrát. Pokud ani do té doby nepřijde odezva ze strany řídicí jednotky, je spojení ukončeno.



Obrázek 4.1: Sequence diagram komunikace mezi řídicí jednotkou a PC

4.4 Popis vizualizačního SW

Na obrázku 4.2 je ukázka hlavního okna vizualizačního SW (Hájek, M., 2007), který slouží pro:

- zobrazování telemetrických dat turbíny, diagnostiku
- ukládání měřených dat
- komunikaci s řídicí jednotkou : nastavování PID regulátoru, zapínání jednotlivých režimů turbíny

Vizualizační SW (*DataReader*) byl programován v prostředí .NET a je tedy určen výhradně pro běh pod operačním systémem Windows. Hlavní okno obsahuje tyto prvky:

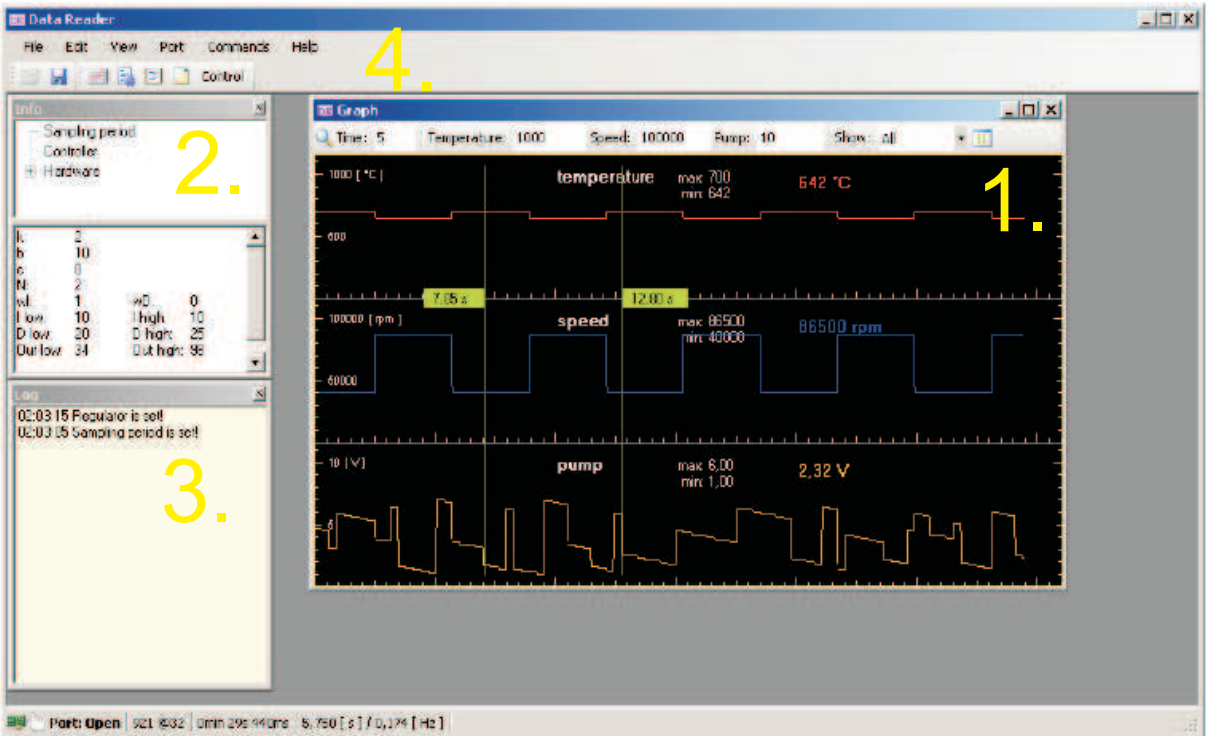
1. Graph
2. Info panel
3. Log panel
4. Ovládací prvky

Graph slouží pro zobrazování telemetrických dat. Okno je rozděleno dle obrázku 4.2 do tří částí, kde každá část zobrazuje konkrétní měřenou veličinu (otáčky, teplota, napětí na palivovém čerpadle). Pro každou měřenou hodnotu se dá individuálně v horní části okna nastavit zvolený rozsah měřených hodnot.

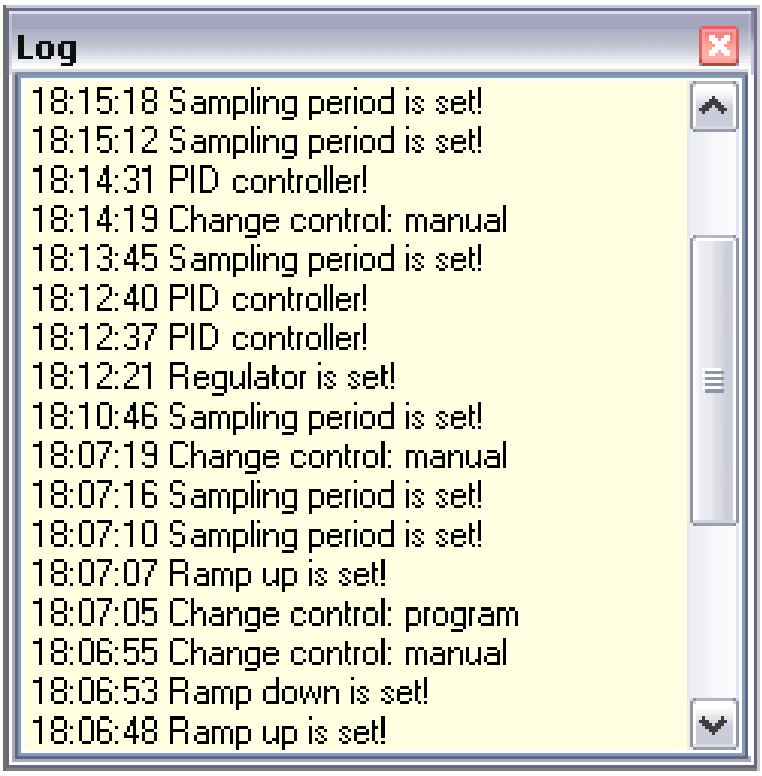
Info panel se používá pro zobrazování nastavení aktuální hodnoty vzorkování a pro aktuální nastavení jednotlivých konstant PID regulátoru (obrázek 5.2).

Log panel slouží jako kontrolní výpis odeslaných příkazů z vizualizačního SW a k výpisu příchodu odpovědí (obrázek 4.3).

Ovládací prvky jsou umístěny v toolbaru hlavního okna. Mezi tyto prvky patří: okno nastavení regulátoru, okno ovládání jednotlivých režimů řídicí jednotky, zobrazování jednotlivých oken. Dále pak tato část obsahuje standardní menu nabídku pro nastavení komunikačního portu, ukládání měřených dat, navázání komunikace a vymazání log panelu.



Obrázek 4.2: Ukázka programu DataReader



Obrázek 4.3: Ukázka výpisu Log panelu

Kapitola 5

Úpravy vizualizačního SW

Vizualizační SW byl vyvíjen dříve než řídicí jednotka. Proto mohlo během jeho vývoje dojít k chybám, které nemusely být odhaleny, protože oba systémy (DataReader a řídicí jednotka) nebyly testovány proti sobě. To se ukázalo jak pravda a muselo tedy dojít k opravám kódu vizualizačního SW. Jednalo se o chyby:

- špatné sestavování konstant do zprávy o nastavení regulátoru
- chybný výpis nastavení konstant regulátoru
- zacyklení při ukládání vzorků do „temp file“

Kromě oprav byly během vývoje řídicí jednotky přidány i nové funkčnosti do vizualizačního SW. Jsou to:

- možnost nastavení rozsahu maximálních a minimálních hodnot reference (otáček) podle polohy plynové páky na vysílači
- nový režim „Control“

5.1 Popis oprav DataReader

Špatné sestavování konstant regulátoru do zprávy

Konstanty regulátoru se nastavují v okně Regulátor (obrázek 5.1). Po kliknutí na tlačítko „Set“ se zavolá metoda umístěná ve třídě `FormController.cs`:

```
private void buttonSet_Click(object sender, EventArgs e)
```

Tato metoda provede uložení aktuálních hodnot do instance třídy `ControllerSettings` (viz příklad).

```
ControllerSettings controllerSettings;
this.controllerSettings.k = float.Parse(this.textBoxK.Text);
this.controllerSettings.wI = float.Parse(this.textBoxWI.Text);
this.controllerSettings.wD = float.Parse(this.textBoxWD.Text);
this.controllerSettings.b = float.Parse(this.textBoxB.Text);
```

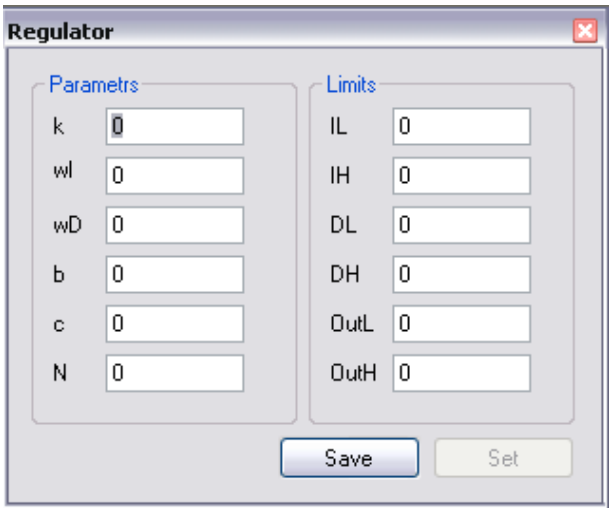
⋮

Po té se v cyklu volá metoda `public void WriteFloat(float number)`, která ukládá data do instance třídy `Datagram`. Na závěr se zavolá metoda pro poslání datagramu po sériové lince.

```
Datagram datagram = new Datagram();
datagram.WriteFloat(this.controllerSettings.k);
datagram.WriteFloat(this.controllerSettings.wI);
```

⋮

```
this.port.WriteWithWatchDog(datagram, "Set regulator");
```

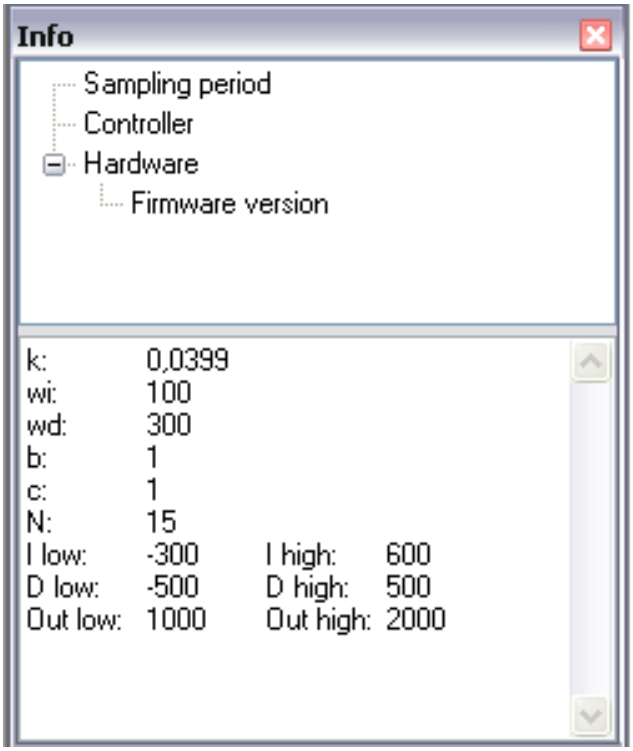


Obrázek 5.1: Okno nastavení konstant regulátoru

Problém byl v tom, že načítaná data z okna „Regulator“ nebyla správně ukládána do jednotlivých „properties“ třídy `ControllerSettings`, čímž docházelo ke ztrátě informace v posílané zprávě.

Chybný výpis nastavení konstant regulátoru

Na obrázku 5.2 je vidět aktuální informační výpis o nastavení jednotlivých konstant regulátoru. Tento výpis se zobrazí, pokud vizualizační SW zažádá o zaslání aktuálního nastavení regulátoru z řídicí jednotky. V souvislosti s předcházející chybou byl výpis chybný, protože přijatá data nepřicházela ve správném pořadí dle komunikačního protokolu (viz příloha) a zároveň data nebyla kompletní.



Obrázek 5.2: Ukázkový výpis Info panelu o nastavení regulátoru

Zacyklení při ukládání vzorků do „temp file“

Tato chyba byla nejzávažnější, neboť způsobovala „zaseknutí“ programu. Pokud je řídicí jednotka připojena a je navázána komunikace s PC, posílá řídicí jednotka telemetrická data. V hlavním nastavení vizualizačního programu (File → Settings, obrázek 5.3) se nastavuje, po kolika vzorcích se ukládají telemetrická data s kompletním nastavení programu do souboru typu temp, z důvodu záchrany dat při pádu programu. Při příchodu nových dat ze strany řídicí jednotky se zavolá metoda:

`private void Port_DataAvailable(object sender, DataAvailableEventArgs e)`,

ve které je pro pársování dat (dle typu zprávy) použita programová konstrukce `switch` (viz příklad).

```
switch (e.Data.Id)
{
// (D) Data
case Datagram.DATA:
Values values = new Values();
values.Id = e.Data.ReadInt();
values.Pump = e.Data.ReadInt();
values.Speed = e.Data.ReadInt();
values.Temperature = e.Data.ReadInt();
this.data.Add(values);
```

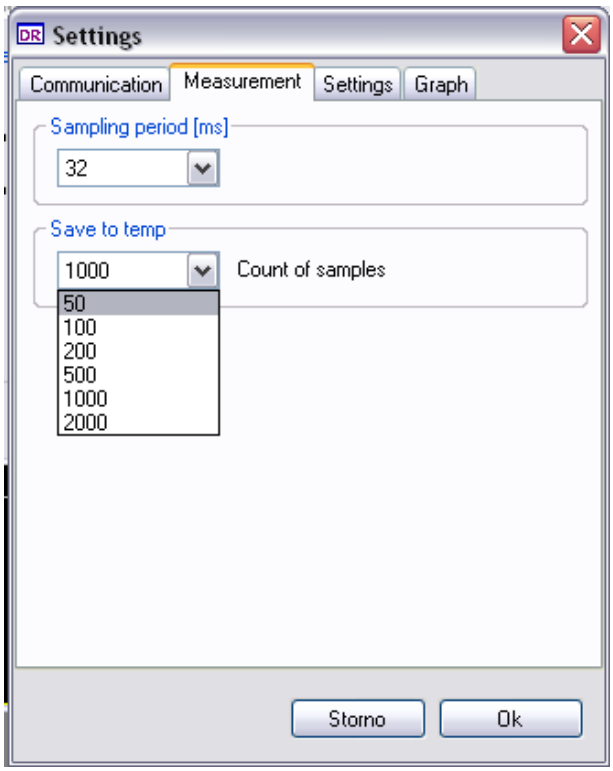
Při příchodu telemetrických dat se provede aktualizace dat instance třídy `Data data`. Pokud přijde „x-tý“ vzorek, po kterém následuje ukládání do temp souboru, zavolá se metoda:
`this.bckgWorkerSaveTempData.RunWorkerAsync(args);`
(`bckgWorkerSaveTempData` je instancí třídy `BackgroundWorker`[1]), která vytvoří vlákno pro ukládání. Protože byla ale podmínka pro ukládání špatně formulována, bylo způsobeno zacyklení programu a tím vytváření vláken pro ukládání. Důsledek byl takový, že program nestíhal vykreslovat aktuální telemetrická data, vypršel časovač pro příchod dalších dat a komunikace byla přerušena.

5.2 Nové funkčnosti DataReader

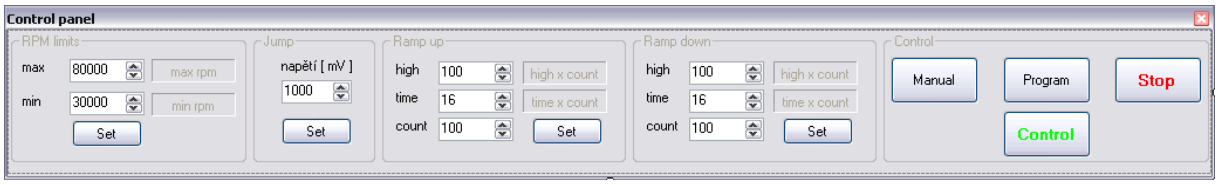
Možnost nastavení rozsahu maximálních a minimálních hodnot reference
Na obrázku 5.4 je znázorněn hlavní ovládací panel, který ovládá jednotlivé režimy turbíny. Pro možnost nastavení minimálních a maximálních hodnot otáček byl přidán do původního panelu (třída `FormControl.cs`) nový prvek `GroupBox` *RPM limits*. Zároveň bylo nutné doplnit komunikační protokol o tuto novou funkčnost (viz tabulka 4.3). Řídicí jednotka tedy při měření polohy plynové páky v dolní poloze vyhodnotí minimální otáčky a naopak v horní poloze maximální.

Režim „Control“
Jednotlivé režimy turbíny před přidáním režimu *Control* byly:

[1] pomocná třída pro správu pracovních vláken



Obrázek 5.3: Hlavní nastavení komunikace a programu



Obrázek 5.4: Hlavní ovládací panel režimů turbíny

- režim *Manual* - napětí na palivovém čerpadle je řízeno přímo polohou plynové páky. Tento režim se používá hlavně při startování turbíny.
- režim *Program* - při tomto režimu jsou aktivní všechny funkčnosti na ovládacím panelu (obrázek 5.4)

Bylo tedy nutné přidat do funkčnosti režim, při kterém turbína přejde do automatického režimu, kde reference otáček bude poloha plynové páky na vysílači. Tlačítko bylo přidáno opět do třídy `FormControl.cs` a zároveň byl komunikační protokol doplněn o nový příkaz (viz 4.2)

Kapitola 6

Návrh stavových automatů řídicí jednotky

V následující kapitole bude popsán návrh jednotlivých stavových automatů, které jsou implementovány v řídicí jednotce.

Při návrhu stavového automatu aplikace je nutné uvažovat následující požadavky:

- s ohledem na funkčnost správné dekomponování jednotlivých stavů a automatů
- uvažování a zpracování chybových stavů, stavů spadající mimo normální pracovní režim

Protože může během běhu řídicí jednotky dojít k chybě, musí být tyto stavy detekovány a zároveň musí dojít k jejich bezpečnému odstranění, popřípadě bezpečnému odstavení jednotky.

6.1 Možné programové implementace stavového automatu

K programové implementaci stavového automatu a přechody mezi jednotlivými stavy, se dá použít hned několik programových konstrukcí. Nejjednodušší implementace je pomocí podmíněného výrazu:

```
if (i == 1) {
  /* Stav 1 */
```

```
} else if (i == 2) {
    /* Stav 2 */
  } else if (i == 3) {
      /* Stav 3 */
    } else {
       /* Error stav*/
      }
```

Z příkladu jde vidět, že kód se stává nepřehledný, těžkopádný a přidání nového stavu naruší celou strukturu. Další možnost je pomocí programové konstrukce `switch`

```
int state;

switch (state) {
  case 1:
   /* Stav 1 */
    break;
  case 2:
    /* Stav 2 */
    break;
  default:
    /* Error stav */
    break;
}
```

Konstrukce je přehlednější, ale přidání nového stavu je značně komplikované. Je tedy nutné volit programovou konstrukci vzhledem k rozsáhlosti implementace a použitého programovacího jazyka. Nejvhodnější způsob je za využití nízkoúrovňových vlastností jazyka C (viz příklad).

```
int stav_1(int *state, args...)
{
int result;
switch (*state) {
  case 0:
   /* inicializace promennych */
   *state = 1;
```

```
    break;
  case 1:
    /* vykonavani stavu */
    *state = 2;
    break;
  case 2:
   /* prechod do dalsiho stavu */
    *state = 0;
    fp_state = stav_2;
    break;
  default:
    /* Error stav */
    fp_state = error_state;
    break;

return result
}
```

Každý stav je reprezentován jednou funkcí, uvnitř které je použita pro vnitřní běh stavu programová konstrukce `switch`. V části `case 0` se provádí inicializace všech potřebných proměnných stavu. V další části `case 1` je implementována vlastní logika stavu. V poslední části `case 2` se provádí rozhodnutí o přechodu do dalšího stavu. Pokud dojde k volání stavu, přičemž ani jedna podmínka `case` nebude splněna, došlo k vnitřní chybě běhu programu a automaticky se stavový automat přepne do chybového stavu, kde je provedeno patřičné ošetření. Volání jednotlivých stavů je prováděno pomocí volání pointeru na funkci s konkrétními potřebnými parametry. Přechod do dalšího stavu se provádí pomocí přesměrování ukazatele pointeru na funkci na jinou funkci. Voláním pointeru na funkci ve smyčce získáváme tedy sekvenční běh stavového automatu a zároveň tím oddělujeme logiku vlastního programu od jednotlivých stavových automatů, čímž se kód stává přehlednější a snazší na odhalování chyb.

6.2 Hlavní stavový automat

Na obrázku 6.1 je znázorněna nekonečná smyčka hlavního stavového automatu, která slouží pro odběr měřených dat z jednotlivých periferií (poloha plynové páky, měření otáček, měření teploty) a zároveň jako smyčka pro volání jednotlivých stavových automatů:

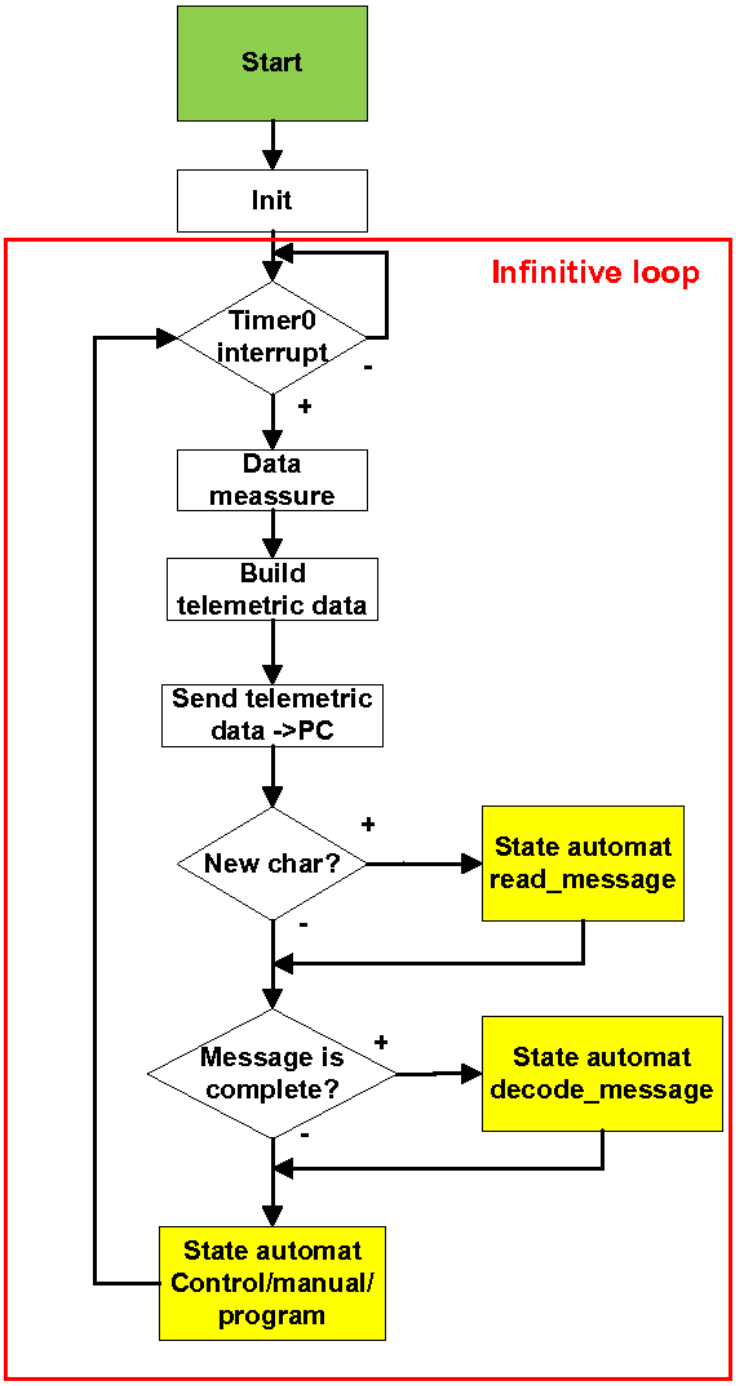
- načítání zprávy
- dekódování zprávy
- koncové stavy (dle přijaté zprávy)

Na začátku se provádí inicializace všech potřebných periferií (čítač timer0, timer1, A / D převodník, vstupy / výstupy), poté se čeká na vypršení časovače, který řídí časově synchronizaci hlavní programové smyčky. Hodnota vzorkování se nastavuje ve vizualizačním SW, zasláním zprávy (tabulka 4.3) s hodnotou násobku základní vzorkovací periody (základní vzorkovací perioda je *16 ms*). Dále následuje sběr aktuálních měřených telemetrických dat, sestavení zprávy telemetrických dat (tabulka 4.3) a odeslání do vizualizačního SW. Následně se kontroluje periferie *UART0* zda nepřišla po sériové lince nějaká data. Pokud ano, spustí se volání stavového automatu pro načítání zprávy. Pokud se tak nestalo, pokračuje se dále v kontrolování zda není nějaká zpráva kompletně načtena. Pokud se tak stane, spustí se volání stavového automatu pro dekódování načtené zprávy. Následně se volá stavový automat pro vykonávání koncového stavu řízení, dle zvoleného aktivního režimu ve vizualizačním SW.

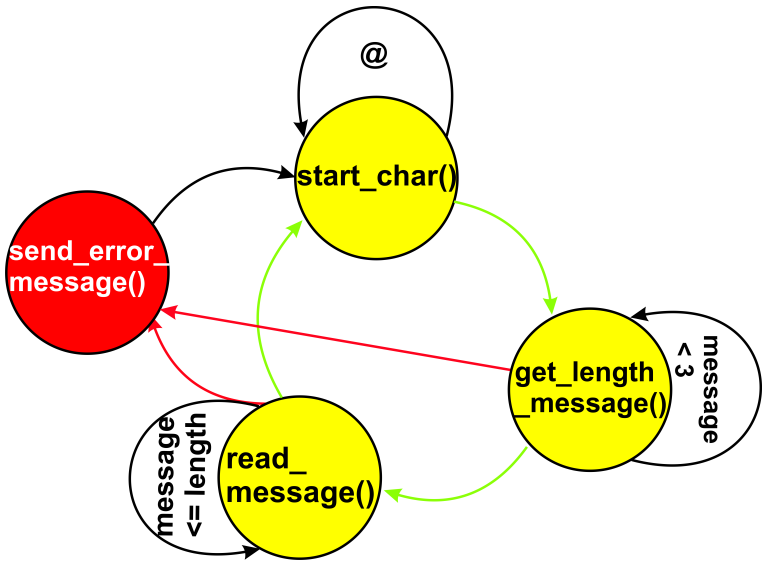
6.3 Stavový automat pro příjem zprávy

Na obrázku 6.2 je znázorněn stavový automat pro načítání zprávy. Automat je navrhován tak, aby byl nezávislý na příchodu počtu jednotlivých znaků. Pokud tedy dojde během komunikace k nějakému uváznutí, stavový automat zůstává ve stavu, dokud není načtena daná posloupnost a počet znaků.

Příchod nového nepřečteného znaku na periferii UART0 se kontroluje v hlavní programové smyčce. Poté se spustí automat pro načítání zprávy. Počáteční stav, funkce **read_char()**, čeká dokud nepřijde znak „@“, který je na začátku každé nové zprávy.



Obrázek 6.1: Hlavní smyčka stavového automatu



Obrázek 6.2: Stavový automat načítání zprávy

Po příchodu úvodního znaku se přejde do dalšího stavu - **get_length_message()**. V tomto stavu se načítají následující další dva znaky, které určují celkovou délku zprávy. Po příchodu těchto dvou znaků se provede konverze na celočíselný datový typ **int** a zjištění tím délky zprávy. Pokud při této konverzi dojde k chybě, to znamená že přijatá posloupnost neodpovídala „hexadecimální soustavě“, přejde stavový automat do chybového stavu, ve kterém se provede inicializace stavového automatu. Pokud konverze proběhne v pořádku, přejde automat do dalšího stavu **read_message()**, kde je postupně načítán zbytek zprávy. Po načtení celé zprávy se provede kontrolní XOR napříč celou zprávou. Pokud hodnota XORu odpovídá hodnotě posledního přijatého znaku, načtená zpráva je přijata bez chyb. Pokud tomu tak není, přejde se do chybového stavu, kde se provede inicializace stavového automatu a pošle se chybová zpráva do vizualizačního SW.

6.4 Stavový automat dekódování zprávy

Stavové automaty pro načítání a dekódování zprávy jsou striktně odděleny a jsou na sobě nezávislé, čímž se lépe strukturalizuje program a funkčnost.

Pokud je přijata celá zpráva bez chyb, nastaví se příznak o tomto přijetí, který se následně kontroluje v hlavní programové smyčce (obrázek 6.1). Na obrázku 6.3 je znázorněn stavový automat dekódování zprávy. V prvním stavu **decode_message()** se provede zjištění, zda se jedná o příkaz, nebo zprávu (rozdíl viz kapitola 4). Tedy jestli

další stav bude **statement_decode()** (zpracování příkazu), nebo **message_decode()** (zpracování zprávy). V těchto stavech se pak následně zjistí, jaký bude koncový stav. V programu jsou deklarovány dvě pole pointerů na funkce, zvlášť pro zprávy a příkazy:

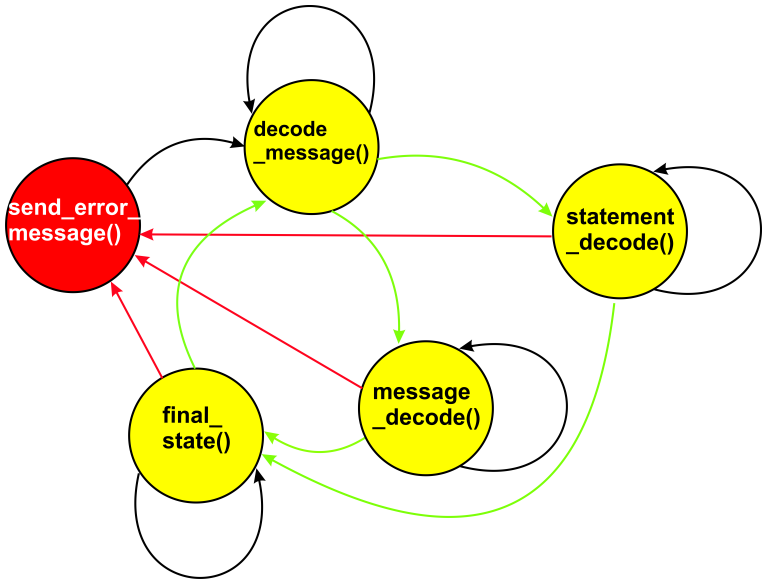
```
//pole pro prikazy
void (*statement_array[])(struct message_data *m_data) = {
statement_forward_control, statement_switch_man_control,
statement_stop_turbine, error_message, ....};

//pole pro zpravy
void (*message_array[])(struct message_data *m_data) = {
error_message, ..., message_ramp_down, error_message,
message_jump, error_message, message_rpm_limits, ...};
```

Každá zpráva / příkaz má jednoznačný identifikátor (viz tabulka 4.2 a 4.3). Protože jsou jednotlivé přijaté znaky zprávy ukládány jako datový typ `char`, lze hodnotu znaku brát také jako číslo. Alfa znaky začínají v ASCII tabulce od hodnoty *65 dec* (znak „A“). Od jednoznačného identifikátoru zprávy / příkazu je odečtena tato bázová hodnota znaku „A“, tím se získá jednoznačný index do pole pointerů zpráv / příkazů. Následně se dané pole pomocí získaného indexu zaindexuje a zavolá konkrétní konečná funkce. Ve stavu **final_state()** se pak získají data z přijaté zprávy a pošle se potvrzovací nebo datová zpráva (v závislosti na dekódované zprávě) do vizualizačního SW.

6.5 Stavový automat řídicí části

Po správném dekódování a přijetí zprávy, se řídicí jednotka uvede do příslušného režimu. Tento režim se nastaví v konečném stavu dekódovacího automatu (**final_state()**). Jako poslední část hlavního stavového automatu je pak volání právě tohoto konečného stavu.



Obrázek 6.3: Stavový automat dekódování přijaté zprávy

Kapitola 7

Softwarové vybavení řídicí jednotky

V následující kapitole bude popsáno softwarové vybavení řídicí jednotky, které běží na procesoru s jádrem ARM (lpc2119). Kód je vzhledem ke své komplexnosti rozdělen do několika základních bloků:

- hlavní programová smyčka
- implementovaný regulátor
- stavové automaty pro zpracování přijatých zpráv
- řídicí stavové automaty
- část pro práci s jednotlivými periferiemi procesoru

Tím se kód stává přehlednější a lépe strukturalizovaný, což značně usnadňuje práci při jednotlivých změnách kódu.

7.1 Vývojové nástroje

Software pro řídicí jednotku je vyvíjen v jazyce ANSI C pod operačním systémem Linux, překládán kompilátorem GCC 3.4.3 s podporou sestavovacího prostředí OMK.

7.1.1 Kompilační systém OMK

OMK OCERA Make System je systém, vyvinutý na katedře Řídicí techniky ČVUT v Praze. Hlavním účelem systému OMK je zjednodušit kompilaci komponent na hosti-

telském systému a crooss kompilaci pro cílový systém. Make systém umožňuje provádět kompilaci ze stromů zdrojových souborů a ukládat výsledky kompilace do oddělené adresářové struktury, tím se zjednodušuje testování a instalace programů. OMK systém dává možnost vyvíjet řídicí program na stolním počítači / notebooku a následně provést cross kompilaci finálního projektu pro cílovou platformu.

Cross-kompilace je proces, kdy se překládá zdrojový kód programu na jednom typu architektury pro jiný typ architektury. Výsledkem tohoto překladu je pak binární soubor spustitelný na cílové architektuře. V případě této aplikace se provádí crooss-kompilace z architektury Intelx86 na architekturu ARM7 (Vosecký, M., 2008).

7.2 Struktura programu

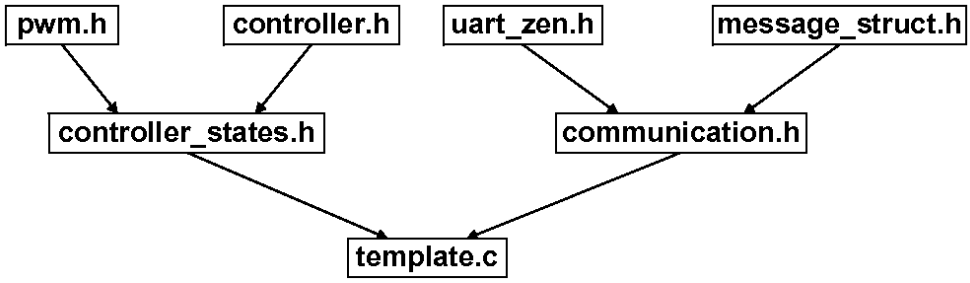
Vzhledem k odlišnostem práce jednotlivých částí programu, je program rozdělen do několika knihoven a do hlavního zdrojového souboru. Na obrázku 7.1 je diagram propojení jednotlivých knihoven. Jsou to:

- pwm.h
- controller.h
- message_struct.h
- controller_states.h
- communication.h
- template.c
- uart_zen.h

Jasně se tím odděluje program na synchronizační, komunikační, řídicí a část pro práci s jednotlivými periferiemi.

7.2.1 Knihovna regulátoru (controller.h)

Knihovna `controller.h` (Ondřej Špinka) obsahuje dvě funkce. Funkce `PID_init(args)` inicializuje strukturu (`PID_params_structure`) parametrů PID regulátoru. Tato struktu-



Obrázek 7.1: Diagram zdrojových souborů řídicího programu

ra pak udržuje vnitřní stavy a konstanty regulátoru. Druhá funkce `PID_control(args)` obsahuje a vykonává samotný algoritmus PID regulátoru.

7.2.2 Knihovna pro práci s PWM modulem (pwm.h)

Procesor lpc2119 disponuje PWM modulem, přičemž může být jako PWM výstup použito až 6 výstupů procesoru. V knihovně pwm.h (Marek Peca) jsou funkce pro inicializaci a nastavení vlastnosti PWM modulu. V aplikaci je jako PWM výstup použit kanál PWM2 (Philips, 2004). Frekvence PWM signálu je nastavena na 50Hz ($T = 20ms$) a doba v logické „1“ se pohybuje v rozmezí $1 - 2\,ms$, jedná se tedy o generování stejného PWM signálu, jako je servo signál (viz obrázek 3.5).

7.2.3 Knihovna pro práci se sériovou linkou (uart_zen.h)

Pro komunikace po sériové lince procesor disponuje dvěma zařízeními UART0 / UART1 (asynchronní sériové porty (Philips, 2004)). V procesoru se pro komunikaci a nahrávání programu používá zařízení UART0 s rychlostí komunikace 38400 bps. V knihovně `uart_zen.h` (Ondřej Špinka) se nacházejí všechny potřebné funkce pro kompletní ovládání tohoto zařízení, tedy funkce na nastavení, odeslání/čtení a testování nových přijatých dat.

7.2.4 Knihovna datových struktur (message_struct.h)

Knihovna obsahuje všechny datové struktury jednotlivých přijatých zpráv (viz tabulka 4.3). Při přijetí každé zprávy obsahující data, se tyto data překonvertují na jednotlivé datové typy a uloží do dané struktury. Jedná se například o struktury:

- `struct control_parameters` - struktura jednotlivých parametrů PID regulátoru

- `struct ramp_up_parameters` - struktura parametrů rampy nahoru
- `struct meassure_data` - struktura, do které se ukládají aktuální měřená data

7.2.5 Knihovna stavových automatů pro zprávy (communication.h)

Dle obrázku 7.1 jde vidět, že knihovna `communication.h` „inklůduje“ knihovnu pro práci se sériovou linkou (uart_zen.h) a knihovnu datových struktur (message_struct.h). Knihovna obsahuje jednotlivé stavy (funkce) pro příjem, dekódování a odeslání zpráv / příkazů. Dále obsahuje definice délky jednotlivých zpráv, definice ID znaků zpráv / příkazů a pole pointerů na funkci pro koncové stavy zpráv / příkazů (kapitola 6.4).

7.2.6 Knihovna stavových automatů řídicích stavů (controller_states.h)

Tato knihovna „inklůduje“ knihovnu pro práci s PWM modulem (pwm.h) a knihovnu s PID regulátorem (controller.h). Obsahuje jednotlivé řídicí koncové stavy. Podle tabulky zpráv 4.3 a příkazů 4.2 to jsou:

- rampa nahoru
- rampa dolu
- skok
- přepnutí na manuální/automatický režim
- zastavení turbíny

7.2.7 Hlavní synchronizační knihovna (template.c)

Jedná se o hlavní zdrojový soubor obsahující metodu `main`. Jsou zde implementovány jednotlivé inicializační funkce pro nastavení vstupů/výstupů procesoru, časovačů TIMER0 / TIMER1, A/D převodníku a nízkoúrovňové funkce pro obsluhu přerušení. Jednotlivé vstupy řídicí jednotky jsou:

- P0.10 nastaven jako záchytný registr CAP1.0 - poloha plynové páky

- P0.18 nastaven jako záchytný registr CAP1.3 - měření doby periody
- P0.30 nastaven jako AIN3 - A / D převodník

Výstup:

- P0.7 nastaven jako PWM2 - vvýstup PWM modulu

Procesor lpc2119 obsahuje dva 32 bitové nezávislé čítače / časovače - TIMER0 a TIMER1. Tyto čítače / časovače se používají v aplikaci pro synchronizaci hlavní programové smyčky a pro měření délky pulzu a délky periody vstupních signálů. TIMER0 je nastaven na čítání frekvencí 1KHz (frekvence PLL[1] procesoru je 10MHz). Tato frekvence se mění příchodem zprávy o změně vzorkovací periody (viz tabulka 4.3). TIMER0 tedy slouží jako časovač pro synchronizaci jednotlivých periferií (sběr dat) a spouštění hlavní programové smyčky (kapitola 6.2). TIMER1 je nastaven na čítání frekvencí 1MHz. Zároveň je na tento časovač zaregistrováno přerušení od registru CAP1.0 (náběžná a sestupná hrana), který měří délku pulzu v log „1“ a CAP1.3 (náběžná hrana), který měří délku periody. Vzhledem k tomu, že frekvence měřené délky pulzu je 50Hz a maximální frekvence pro měření doby periody je 3.5kHz, dává čítač dostatečně velké rozlišení pro přesné měření.

Procesor lpc2119 disponuje jedním 10 bit A/D převodníkem a 4 kanály připojenými k tomuto převodníku, mezi kterými se provádí multiplex. V aplikaci je vzorkovací frekvence A/D převodníku nastavena na 1MHz a kanál se používá AIN3. Vzhledem k tomu, že se jedná o měření statického signálu, je vzorkovací frekvence 1MHz plně postačující. Spouštění A/D převodníku je prováděno softwarově s frekvencí spouštění rovné synchronizační frekvenci (TIMER0).

[1] Phase Locked Loop

Kapitola 8

Řízení chodu motoru

V následující kapitole budou uvedeny jednotlivé režimy chodu motoru ovládané z vizualizačního SW. Jsou to:

- rampa nahoru
- rampa dolu
- skok o definované výšce
- zastavení turbíny
- regulace otáček PID regulátorem

Dále zde bude popsán identifikovaný model turbíny (HÁJEK, M., 2006) a z něj plynoucí omezení pro řízení turbíny. Pro řízení otáček je použit PID regulátor, jeho struktura a implementace bude popsána níže.

8.1 Identifikovaný model

V bakalářské práci (HÁJEK, M., 2006) byla turbína identifikována jako SISO systém 2. řádu a byl vytvořen přenosový model mezi napětí na palivovém čerpadle (vstup) a otáčkami turbíny (výstup). Výsledný přenos je:

$$P(s) = \frac{72298}{s^2 + 2.08s + 1.513} \tag{8.1}$$

Dále pak bylo z identifikace určeno omezení rychlostí náběhu (*rate limit*[1]) napětí na palivovém čerpadle. Hodnota omezení byla určena na:

$$rateLimit = 0.85V/s \tag{8.2}$$

Více o identifikovaném modelu viz (Hájek, M., 2006).

8.2 PID regulátor

Na obrázku 8.1 je zobrazeno blokové schéma PID regulátoru které bylo navrženo Ondřejem Holubem a implementováno Ondřejem Špinkou. Oproti klasickému PID regulátoru využívá některé přidané techniky jako jsou:

- vážení a filtrování reference
- dopředná větev
- separace D složky (reference a měření)
- blok pro beznárazové přepínání z manuálního řízení

Parametry regulátoru jsou:

- k, i, d, m - proporcionální, integrační, derivační a dopředné zesílení
- b, c - koeficienty vážení reference pro proporcionální a derivační složku
- w - filtr reference
- n - filtr derivační složky
- h - vzorkovací perioda

Výpočet rozdílu reference r a měření y je rozděleno do dvou separátních větví. To umožňuje vážení reference. Z tohoto důvodu má regulátor odděleny vstupy pro r a y, místo jednoho vstupu pro regulační odchylku. Nízkofrekvenční filtr je začleněný do obou derivačních větví (měření, reference), aby se zbavilo možného vysokofrekvenčního šumu

[1] omezení rychlosti náběhu akční veličiny

superponovaného na vstupních signálech. Nakonec je pak příspěvek derivační větve staticky omezen saturací.

Integrační větev je navržena klasicky, přičemž výstup integrační větve je staticky omezen pomocí *anti-windup* filtru.

V manuálním režimu (MCM - manual control mode) je sumátor integrační odchylky nastaven do polohy tak, že odpovídá přesně aktuálnímu výstupu. Proto když dojde k přepnutí do automatického režimu (ACM - automatic control mode), nedojde k prudkým rázům do akčního zásahu.

Proporcionální větev je také klasická, pouze reference r je vážena pomocí parametru b před tím, než se počítá regulační odchylka e.

Výstup regulátoru je staticky omezen pomocí saturace, tak aby nemohlo dojít k překročení akčních zásahů. Přičemž k této situaci by nemělo dojít pokud jednotlivé parametry regulátoru jsou nastaveny přesně (Špinka, O., 2009).

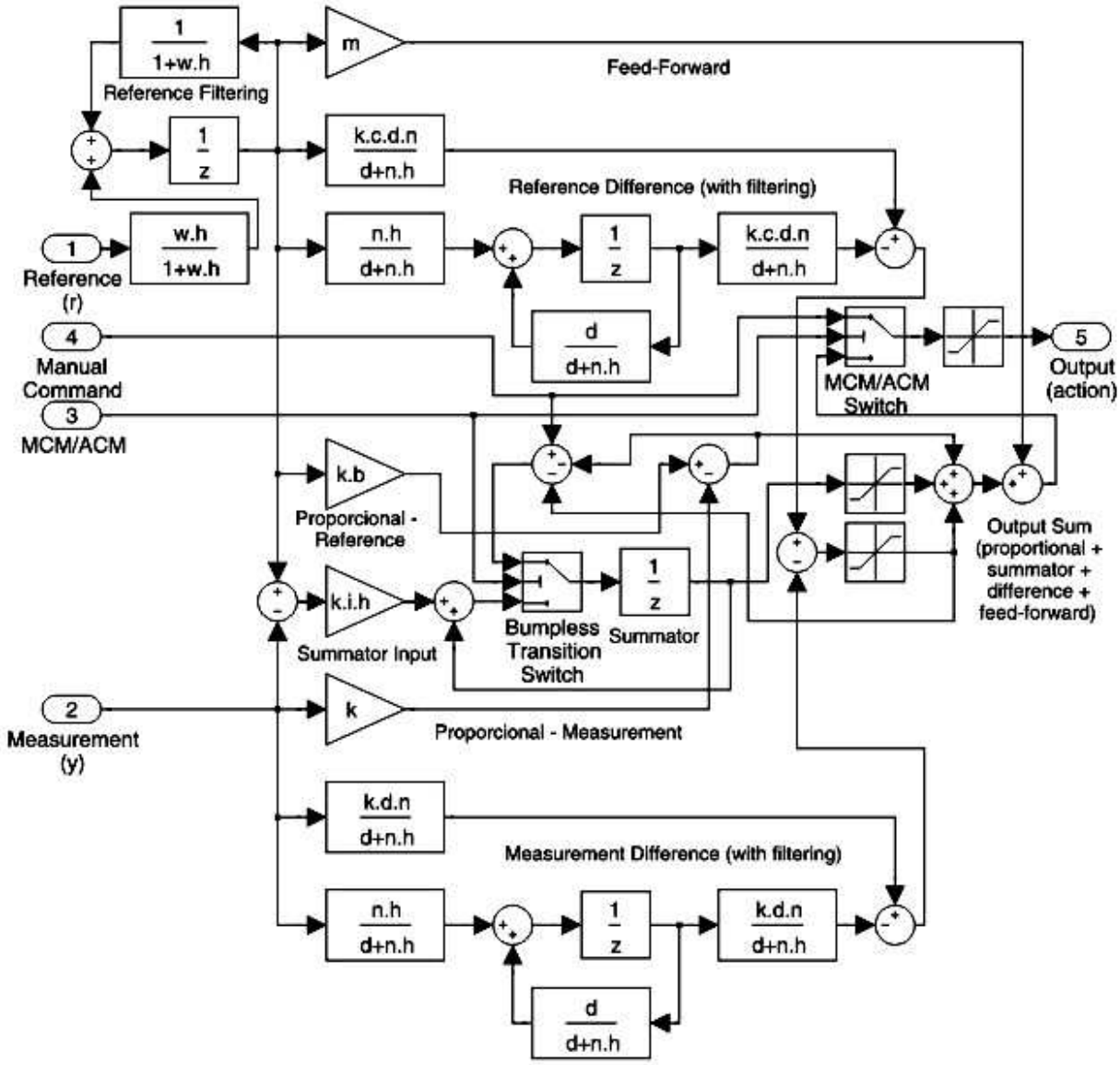
8.3 Simulace

V této části budou uvedeny jednotlivé simulace režimů řídicí jednotky ovládané z vizualizačního SW. Protože zatím nedošlo k ostrým testům na reálném zařízení, jsou výsledky pouze demonstrativní pro ověření správné funkčnosti systému. Na obrázku 8.2 je diagram vývoje řídicího systému. Z diagramu lze vidět že každý návrh řídicího systému obsahuje několik komplexních klíčových bodů a zároveň, že simulace (experimenty) jsou nedílnou součástí každého návrhu. Obousměrnost šipek naznačuje, že pokud systém neobstojí v experimentech, neznamená to pouze chybu implementace (Rantzer, A., 2007).

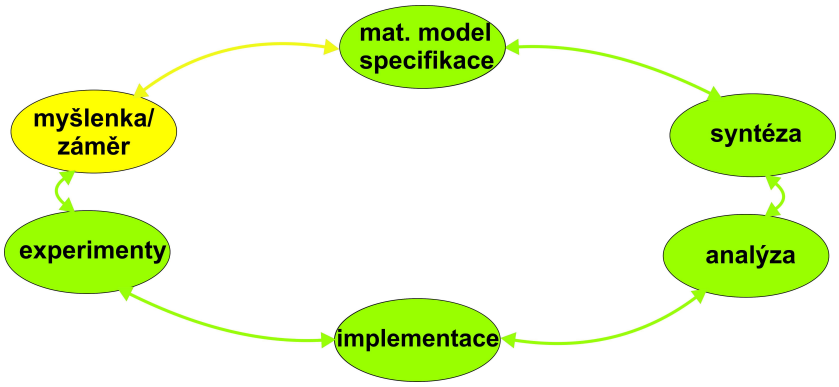
8.3.1 Rampa nahoru a rampa dolu

Rampa nahoru

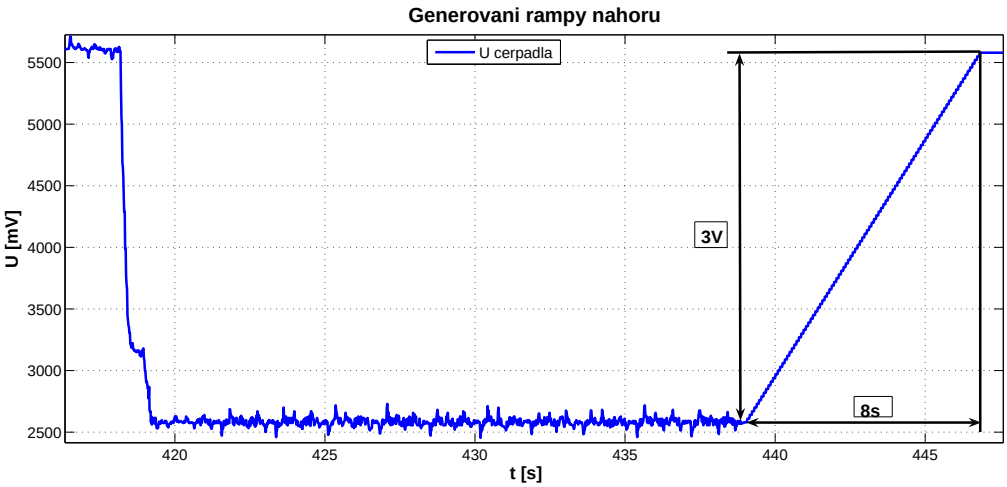
Na obrázku 8.3 je znázorněno napětí na palivovém čerpadle. Do času $t = 439s$ byla jednotka v automatickém režimu a výstupní průběh odpovídá akčnímu zásahu. Po té byla jednotka přepnuta vizualizačním SW do programového režimu a zaslán požadavek na generování rampy směrem nahoru s parametry: výška = 3V, délka = 8s. Z průběhu lze vidět, že jednotka opravdu přešla z automatického režim a vygenerovala rampu s požadovanými parametry.



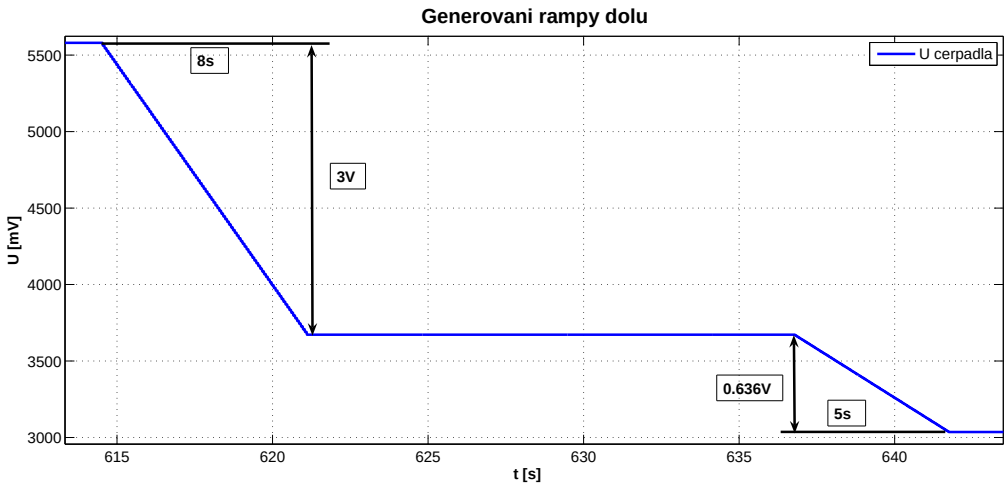
Obrázek 8.1: Blokové schéma implementovaného regulátoru



Obrázek 8.2: Diagram vývoje řídicího systému



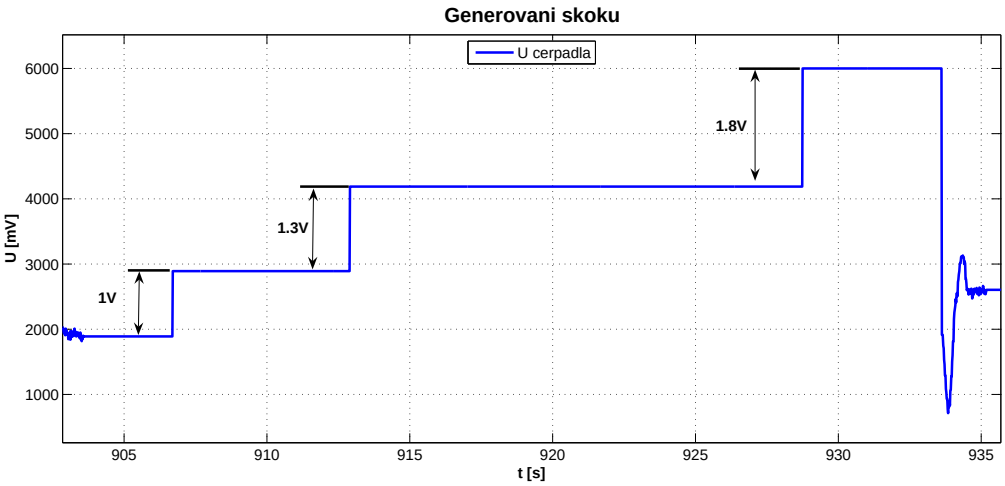
Obrázek 8.3: Rampa nahoru



Obrázek 8.4: Rampa dolu

Rampa dolu

Na obrázku 8.4 je generování rampy dolu. Jednotka je v programovém režimu (přijímá povely z ovládacího panelu 5.4), přičemž v časech $t = 614s$ a $t = 637s$ přijde požadavek z vizualizačního SW na generování rampy dolu s parametry: první rampa: $délka = 8s$, $výška = 3V$, druhá rampa: $délka = 5s$, výška $= 0.636V$. Z průběhu lze vidět, že jednotka opět vygenerovala rampu s požadovanými parametry a zároveň, že jednotka setrvá v programovém režimu a udržuje napětí na čerpadle, dokud nepřijde z vizualizačního SW jiný požadavek.



Obrázek 8.5: Skok o definované výšce

8.3.2 Skok o definované výšce

Na obrázku 8.5 jsou generované skoky o výšce: *1V, 1.3V, 1.8V*. V čase $t = 934s$ přejde jednotka do manuálního režimu.

8.3.3 Zastavení turbíny

Při stisknutí tlačítka STOP na hlavním ovládacím panelu (obrázek 5.4) řídicí jednotka začne automaticky snižovat napětí na palivovém čerpadle rychlostí danou *rate limit* omezením.

8.3.4 Automatický režim

V automatickém režimu jsou otáčky turbíny řízeny pomoci PID regulátoru (viz 8.2). Do tohoto režimu se řídicí jednotka dostane buď přepnutím z manuálního režimu, nebo přepnutím z programového režimu. V obou případech se využívá beznárazového přepnutí popsané v kapitole 8.2. Po přepnutí do automatického režimu je nutné sjet plynovou pákou na minimum s tím, že jako minimální otáčky se nastaví otáčky turbíny v době přepnutí a jako minimální hodnota napětí na palivovém čerpadle se nastaví hodnota v době přepnutí.

8.3.5 Manuální režim

V tomto režimu je napětí na čerpadle ovládáno přímo polohou plynové páky. Tento režim se používá zejména při startování turbíny.

Kapitola 9

Závěr

Diplomová práce se zabývala návrhem a konstrukcí řídicí jednotky malého proudového motoru. Kromě samotného návrhu řídicí jednotky byla do návrhu systému začleněna komunikace řídicí jednotky s vizualizačním SW, který byl vytvořen v diplomové práci (Hájek, M., 2007). Tímto se řídicí jednotka stává plně konfigurovatelnou a diagnostikovatelnou a zároveň velmi univerzální vůči použitému typu modelářské turbíny.

Pro implementaci řídicí jednotky byla zvolena tříúrovňová struktura (kapitola 3), kde jednotlivé části jsou: řídicí elektronika, senzorická část a výkonová část. Toto rozdělení zaručuje snazší možnost opravy a změnu návrhu konkrétní části. Rovněž tím pak eliminuje možnost rušení mezi jednotlivými částmi.

Pro řízení turbíny, odběr měřených dat a spouštění jednotlivých stavových automatů (načítání a dekódování zprávy, řídicí automat) byl navržen základní stavový automat řízení (kapitola 6), jehož vývojový diagram je znázorněn na obrázku 6.1. Automat běží v nekonečné smyčce, která je synchronizována časovačem *TIMER0*. Hodnota vzorkovací periody (*základní = 16ms*) se nastavuje přijmutím zprávy o nastavení vzorkovací periody.

Protože nebylo nikde uvedeno v diplomové práci Miroslava Hájka jak probíhá komunikace mezi řídicí jednotkou a vizualizačním SW, musel být pro návrh dalších částí systému prostudován komunikační protokol (kapitola 4). Na základě toho byl vytvořen sequence diagram (obrázek 4.1), který popisuje průběh komunikace v čase. Dále bylo nutné upravit vizualizační SW, protože docházelo k chybovému běhu: SW se zasekával, posílaná data o nastavení regulátoru nebyla v konzistentním stavu. Následně se ukázalo nezbytně nutné přidat nové funkčnosti: nastavení mezních hodnot otáček (RPM limits), přepínač automatického režimu (tlačítko Control).

Pro načítání nové přijaté zprávy (kapitola 6) byl navržen stavový automat. Stavový automat byl rozdělen na dva nezávislé automaty: automat pro načítání zprávy a automat

pro dekódování zprávy. Tím se jasně odděluje logika načítání a vykonávání. Zároveň byla do implementace začleněna možnost vyskytnutí chybových stavů a jejich ošetření.

Jako koncové stavy byly navrženy a implementovány jednotlivé řídicí algoritmy režimů chodu motoru. V automatickém režimu se využívá PID regulátor, který byl navržen Ondřejem Holubem a implementován Ondřejem Špinkou. Jedná se o PID regulátor, který využívá některých přidaných technik: vážení a filtrování reference, dopředná větev, separace derivační složky (reference a měření) a blok pro beznárazové přepínání z manuálního režimu. Zároveň byl do implementace řízení začleněn omezující parametr - omezení *rate limit*, které bylo stanoveno v bakalářské práci (Hájek, M., 2006).

Protože zatím neproběhly ostré testy na reálném zařízení, byly jednotlivé vstupní signály emulovány v laboratoři (otáčky: funkční generátor, teplota: zdroj napětí). Ukázky jsou uvedeny v kapitole 8.

Literatura

Philips (2004), *User manual - LPC2119/LPC2129*. Online zdroj `http://www.datasheetcatalog.org/datasheet/philips/LPC2119.pdf`.

Graupner (2009), *SPEED Motor Specifications*. Online zdroj `http://www.modelflight.com.au/graupner_speed_specifications.htm`.

Hájek, M. (2006), *Bakalářská práce - Identifikace proudového motoru pro model letadla*.

Hájek, M. (2007), *Diplomová práce - Návrh a implementace SW pro vizualizaci telemetrických dat pro malý proudový motor*.

JETI model (2009), *Modelářský DC regulátor JES006*. Online zdroj `http://www.jetimodel.cz/index.php?page=product&id=79`.

Pačes, P. (2008), *Přednášky k předmětu X38PRS*. Online zdroj `http://www.pacespavel.net/PRS/`.

Peca, M. (2008), *Diplomová práce - Kráčející robot*.

Špinka, O. (2009), *Online verze - RAMA UAV Control System*. Online zdroj `http://rtime.felk.cvut.cz/helicopter/`.

Rantzer, A. (2007), *Přednášky k předmětu - Computer Controlled systems*. Online zdroj `http://www.control.lth.se/~kursdr/lectures.html`.

Ripka, P., Ďado, S., Kreidel, M. a Novák, J. (2005), *Senzory a převodníky*, Praha: Vydavatelství ČVUT. ISBN 80-01-03128-3.

Stejskal, P. (2003), *Amatérské turbíny, Stránky pro příznivce modelářských turbín*. Online zdroj `http://www.mklipence.wz.cz/turbiny.htm`.

Švéda, M. a Hubík, Vl. (2008), *Elektrické pohony pro kritické aplikace v letectví*, AUTOMA 7/2008. Online zdroj `http://www.odbornecasopisy.cz/res/pdf/37537.pdf`.

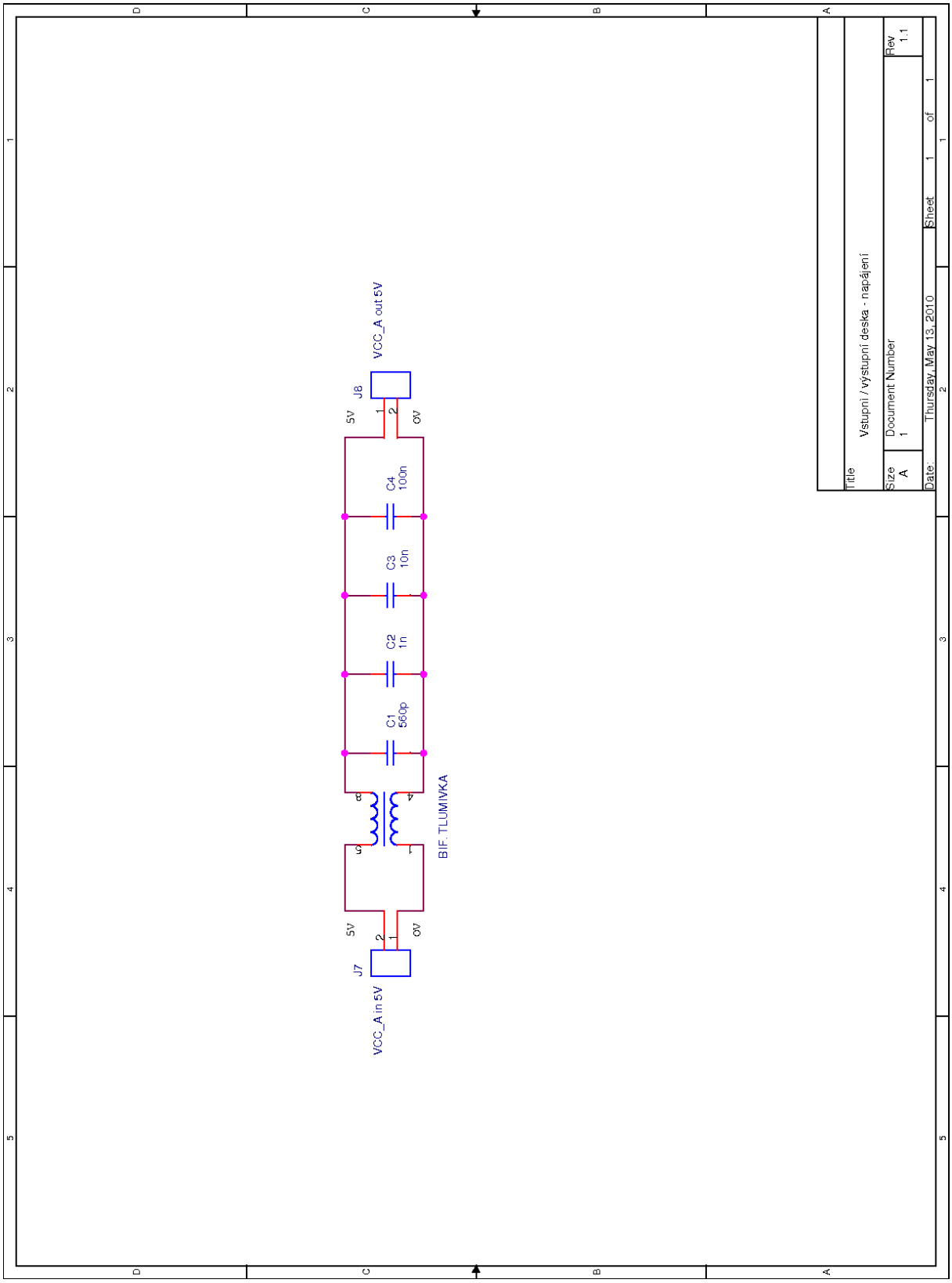
Vosecký, M. (2008), *Bakalářksá práce - Návrh a realizace řídicí jednotky malého proudového motoru.*

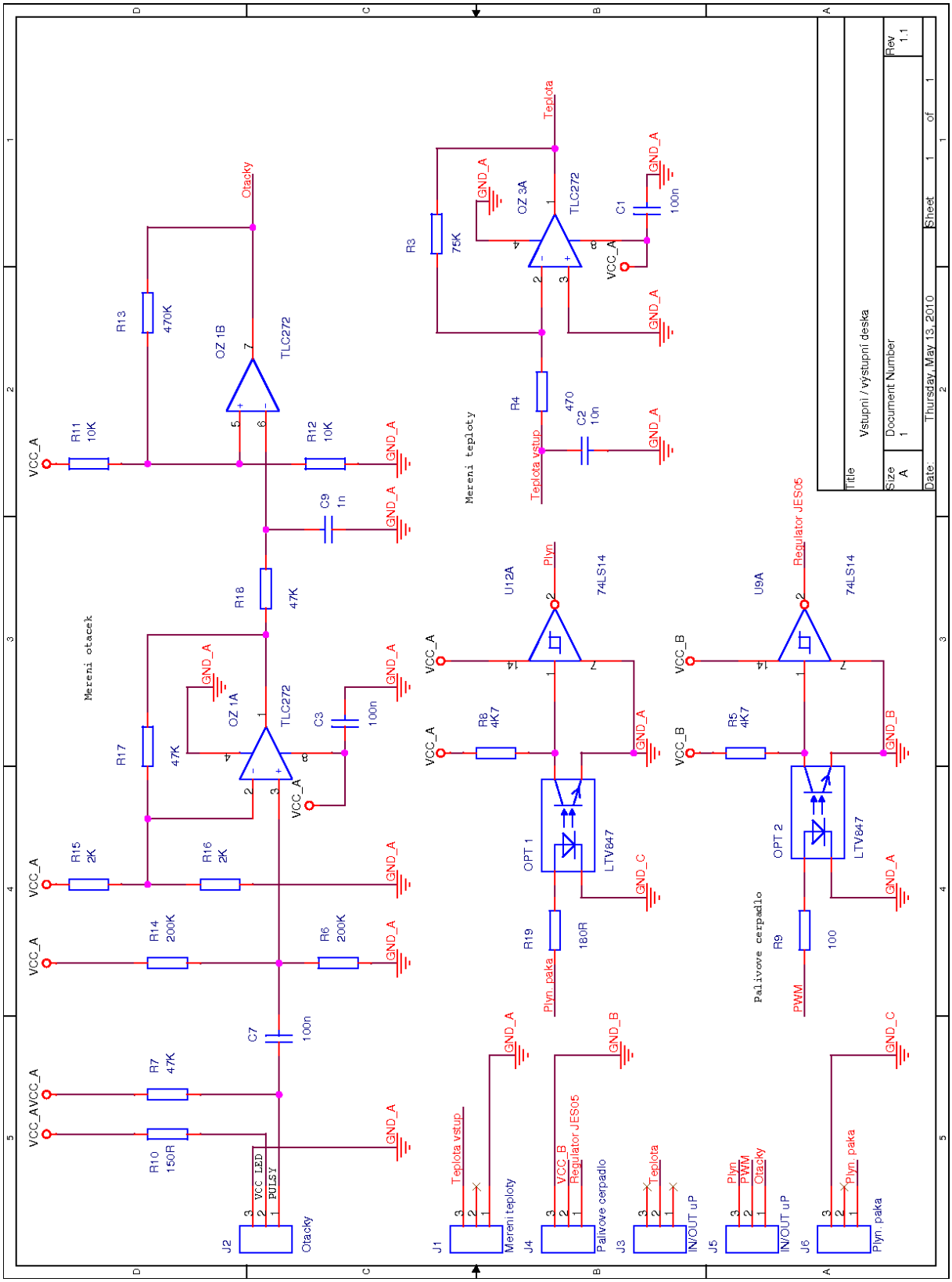
Příloha A

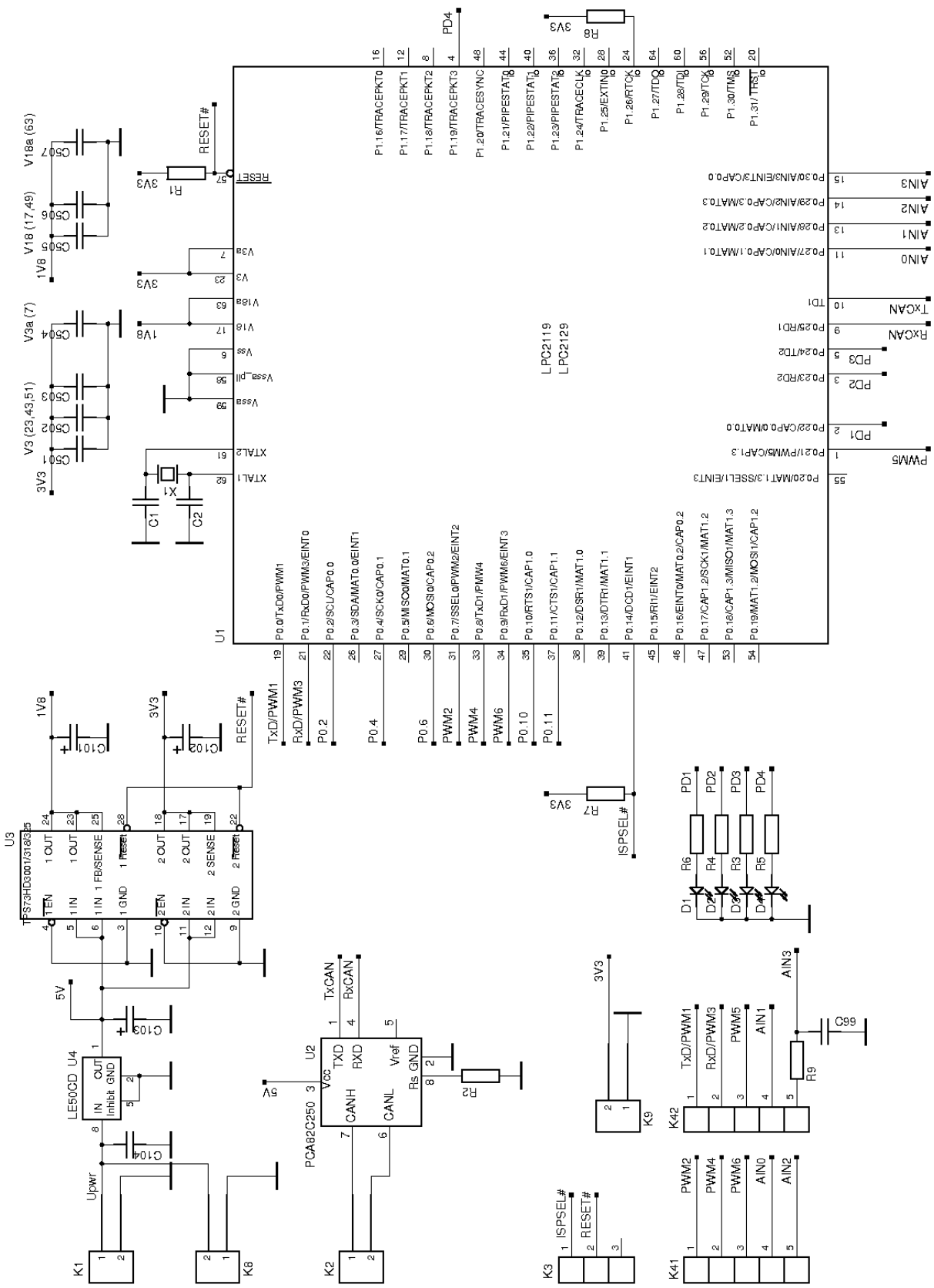
Schémata

V této části přílohy jsou:

- napájecí obvod desky vstupů/výstupů
- schéma desky vstupů/výstupů
- SpejblARM (Peca, M., 2008)


VCC_A in 5V
J7
5V
0V
BIF. TLUMIVKA
C1
560p
C2
1n
C3
10n
C4
100n
J8
VCC_A out 5V
Title
Vstupní / výstupní deska - napájení
Size
A
Document Number
1
Rev
1.1
Date:
Thursday, May 13, 2010
Sheet
1
of
1


Mereni otacek
Otacky
Mereni teploty
Teplota
Palivove cerpadlo
Regulator JES05
Plyn
Plyn paka
Teplota vstup
VCC_A
VCC_B
GND_A
GND_B
GND_C
TLC272
OZ 1A
OZ 1B
OZ 3A
74LS14
LTV847
OPT 1
OPT 2
J1
J2
J3
J4
J5
J6
Title
Vstupni / výstupni deska
Size
A
Document Number
1
Rev
1.1
Date:
Thursday, May 13, 2010
Sheet
1
of
1


U1
LPC2119
LPC2129
U3
TPS73HD3001/3180/35
U4
LE50CD
U2
PCA82C250
K1
K2
K3
K6
K41
K42
K9
RESET#
ISPSEL#
TxD/PWM1
RxD/PWM3
PWM5
AIN0
AIN1
AIN2
AIN3
TxCAN
RxCAN
Upwr
5V
3V3
1V8
V3 (23,43,51)
V3a (7)
V18 (17,49)
V18a (63)
PD1
PD2
PD3
PD4
PWM2
PWM4
PWM6
P0.6
P0.10
P0.11
P0.2
P0.4
C99
R9
R2
R7
R8
R1
X1
C1
C2

Příloha B

Komunikační protokol

Tento protokol byl navržen Miroslavem Hájkem v diplomové práci (HÁJEK, M., 2007) jako komunikační protokol mezi softwarovým vybavením osobního počítače, PDA a řídicí jednotkou pro řízení proudového motoru. Komunikace probíhá po sériové lince RS232, která je popřípadě emulována na PC rozhraním USB. Data jsou posílána ve formátu *big-endian*. Zpráva je posílána jako posloupnost znaků, kde čísla jsou reprezentována řetězcem hexadecimálních čísel o různé délce.

B.1 Protokol

Obecný formát zpráv:
Samotný řetězec znaků se skládá ze zprávy začínající kontrolním start bytem, následují znaky reprezentující délku zprávy, id zprávy a samotná data. Jako poslední se přenáší kontrolní XOR byte, který se počítá napříč celou zprávou.

```
<start byte><délka zprávy><id zprávy><data><XOR>
```

Použité znaky:
start byte

Velikost [Byte]	Znak	Popis
1	@	Úvodní znak

Tabulka B.1: Start byte

délka zprávy - určuje délku samotných dat. Hodnota se tedy nevztahuje na start byte a na délku zprávy.

Velikost [Byte]	**Znak**	**Popis**
2		Počet dat v datablokuvčetně hodnoty XOR

Tabulka B.2: Délka zprávy

id zprávy - samotné textové řetězce jsou identifikovány podle znaku *id*. Tím se dělí na **zprávy**, **příkazy** a **potvrzující/chybové zprávy**.

Posílané zprávy se dělí do dvou základní typů - zprávy a příkazy. Příkazy neobsahují část *data* a za řídicím znakem **C** následuje *id* konkrétního povelu. Zprávy obsahují *id* povelu a následně datovou část odpovídající konkrétnímu typu zprávy (viz kapitola B.2.2).

V následující tabulce B.3 je seznam všech podporovaných povelů.

Velikost	**Znak**	**Popis**
1	V	Data
1	C	Příkaz viz tabulka příkazů
1	R	Regulátor
1	E	Error
1	O	Ok
1	J	Provede skok o definované výšce
1	U	Rampa nahoru
1	D	Rampa dolů
1	P	Nastaví vzorkovací periodu
1	S	Otáčky
1	L	Nastavení dolní a horní hranice otáček
1	F	Automatický režim

Tabulka B.3: Tabulka všech podporovaných povelů

XOR - kontrolní znak vypočítaný přes celou zprávu a přidaný jako poslední.

Velikost	Znak	Popis
1		XOR

Tabulka B.4: XOR

B.2 Směr zpráv/příkazů

V tabulce B.5 jsou uvedeny směry vysílání všech povelů a následných odpovědí.

PC $\Longrightarrow$		$\Longleftarrow$ **Řídicí deska**	
Zpráva	**Odpověď**	**Odpověď**	**Zpráva**
			Z Data
Z Regulátor		Ok / Error	
Z Otáčky		Ok / Error	
Z Skok		Ok / Error	
Z Rampa		Ok / Error	
Z Vz. perioda		Ok / Error	
Z Limity pro otáčky		Ok / Error	
P Regulátor		**Z** Regulátor	
P Program		Ok / Error	
P Manuál		Ok / Error	
P Automat		Ok / Error	
P Stop		Ok / Error	
Z Vz. perioda		**Z** Vz. perioda	

Tabulka B.5: Směr zpráv/příkazů

* **Z** - zpráva, **P** - příkaz

B.2.1 Příkazy

`<kód>`

§	0	3	**C**	**Id**	XOR

Délka: 3

Příkazy jsou zprávy, které obsahují pouze *id* zprávy (znak C), po kterém následuje jeden

znak reprezentující daný příkaz. Příkaz tak neobsahuje blok data. V následující tabulce B.6 je seznam všech použitých příkazů.

Tabulka příkazů

Příkaz	**Kód**	**Popis**
regulátor	R	Řídicí deska pošle aktuální nastavení regulátoru
program	A	Předá řízení programu
manual	B	Přepne řízení na manuální
stop	C	Zastaví turbínu
vz. perioda	S	Odešle do PC nastaveno vz. periodu

Tabulka B.6: Příkazy

B.2.2 Zprávy

Data

```
<id dat><napětí na čerpadle><otáčky><teplota>
```

§	2	2	**V**	**Id**	**P**	**S**	**T**	XOR

Délka: 34

Znak	**Formát**	**Jednotky**	**Popis**
Id	int32	-	Číslo vzorku
P	int32	[mV]	Napětí na čerpadle
S	int32	[ot/min]	Otáčky
T	int32	[°C]	Teplota

Na prvním místě je úvodní znak, následuje znak, který identifikuje datablok jako data. Délka zprávy je 34 (*22h*). *Id* odpovídá číslu vzorku a je inkrementováno s každým následujícím vzorkem. Pokud poslaná hodnota XOR neodpovídá vypočtené, informuje se uživatel.

Nastavení regulátoru

`<parametry><limity>`

§	C	2	**R**	**k**	w_I	w_D	**b**	**c**	**N**	I_L	I_H	D_L	D_H	O_L	O_H	XOR

Délka: 192

Znak	**Formát**	**Jednotky**	**Popis**
k	float	-	Zesílení
w_I	floate	-	Zlomová frekvence integrační složky
w_D	float	-	Zlomová frekvence derivační složky
b	float	-	Váhový koeficient nelin. členu P složky
c	float	-	Váhový koeficient nelin. členu D složky
N	float	-	Dolní propust derivační složky
I_L	float	-	Spodní mez integrátoru
I_H	float	-	Horní mez integrátoru
D_L	float	-	Spodní mez derivační složky
D_H	float	-	Horní mez derivační složky
O_L	float	-	Spodní mez výstupu
O_H	float	-	Horní mez výstupu

Tabulka B.7: Jednotlivé konstanty PID regulátoru

Vzorkovací perioda

`<násobitel>`

§	0	A	**P**	**M**	XOR

Délka: 10

V tabulce B.9 jsou uvedeny hodnoty jednotlivých vzorkovacích period. Základní vzorko-

Znak	**Formát**	**Popis**
M	int32	Násobitel základní vz. periody

Tabulka B.8: Vzorkovací perioda

vací perioda je *16ms*.

Násobitel	**Vzorkovací perioda [ms]**
1	16
2	32
⋮	⋮

Tabulka B.9: Násobky vzorkovací periody

Otáčky

`<otáčky>`

§	0	A	**S**	**X**	XOR

Délka: 10

Znak	**Formát**	**Popis**
X	int32	Otáčky turbíny [ot/min]

Tabulka B.10: Otáčky turbíny

Skok

`<napětí>`

§	0	A	**J**	**U**	XOR

Délka: 10

Po přijetí tohoto příkazu řídicí jednotka vygeneruje napěťový skok na palivovém čerpadle

Znak	**Formát**	**Popis**
U	int32	Napětí [mV]

Tabulka B.11: Napěťový skok

o výšce dané parametrem v tabulce B.11.

Rampa nahoru a dolu

`<počet kroků><Výška jednoho kroku><čas jednoho kroku>`

§	1	A	**U**	**C**	**H**	**T**	XOR

Znak	**Formát**	**Popis**
C	int32	Počet kroků
H	int32	Výška jednoho kroku [mV]
T	int32	Délka jednoho kroku [ms]

Tabulka B.12: Parametry rampy nahoru a dolu

Délka: 26

Po přijetí tohoto příkazu řídicí jednotka začne generovat rampu směrem nahoru (U), popřípadě dolu (D) definované výšky (CxH) a délky (CxT).

B.2.3 Potvrzující a chybové zprávy

Potvrzující zprávy se posílají:

- pouze z řídicí jednotky

Potvrzují se:

- příkazy
- nastavení regulátoru

Chybové zprávy se posílají:

- pokud nastala chyba při zpracovávání přijaté zprávy

Potvrzující zprávy

`<id potvrzované zprávy>`

§	0	3	**O**	**Id**	XOR

Délka: 3

Pokud je vyžadována potvrzující zpráva, měla by být do určitého časového limitu doručena zpět straně, která ji vyžádala. Pokud se tak nestane:

Znak	**Formát**	**Popis**
id	char	Id zprávy, která má být potvrzena. Např. pro regulátor je *Id* „R“

- informuje se uživatel
- opakuje se automaticky pokus (např. max 3x → informuje se uživatel)

Chybové zprávy

`<id>`

§	0	A	**E**	**Id**	XOR

Délka: 10
Id odpovídá identifikátoru přijaté zprávy.

Zabezpečení zpráv - XOR

`<zpráva><XOR>`

Kontrolní příčná sudá parita se počítá jako exkluzivní OR. Vždy přes celou zprávu.
Příklad

§	**2**	**2**	**V**	**Id**	**P**	**S**	**T**	XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

B.2.4 Formát dat

Int32

Velikost čísla typu *int32* je *4B* ⇒ 8 znaků

Float

Velikost čísla typu *float* má délku 4B ⇒ 8 znaků

Char

Představuje jeden libovolný znak

B.2.5 Příklad zprávy

Data

§22V000000020000157C000151E400000282[XOR]

Id zprávy:	V = data
Id (číslo) vzorku:	2
Napětí na čerpadle:	5500 mV
Otáčky:	86500 ot/min
Teplota:	642 °C

Příloha C

Obsah přiloženého CD

K této práci je přiloženo CD s tímto obsahem:

- Adresář 1: **datasheets** (dir) Datasheety nejdůležitějších součástek použitých v zapojení
- Adresář 2: **schémata** (dir) Schémata elektronických zapojení
- Adresář 3: **turbina_codes** (dir) Zdrojové kódy programu
- Adresář 4: **latex** (dir) Zdrojové kódy zprávy
- file 5: **dp_2010_michal_vosecky.pdf**