České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídicí techniky

Diplomová práce

Demonstrační prostředí pro vestavěné systémy

Únor 2004 Ondřej Špinka

Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ……. …….

Podpis

Abstrakt

Tato práce se zabývá návrhem a konstrukcí vestavěných systémů, tj. elektronických řídicích jednotek, které jsou integrální součástí řízeného stroje. Skládá se ze dvou částí; v první je popsán návrh, konstrukce a realizace řídicí jednotky malokapacitní automatické krmičky ryb (včetně konstrukce samotné krmičky, tj. řízeného stroje). Ve druhé části je popsán koncepční návrh řídicího systému modelu vrtulníku a podrobněji pak jeden jeho modul, jednotka řízení servomotorů. Obě popisovaná zařízení mají mnoho společného; jsou řešeny jako autonomní inteligentní moduly, připojené spolu s dalšími jednotkami na průmyslovou sběrnici (RS-485 a CANBUS) a obě používají jako akční členy servomechanismy. Zatímco konstrukce automatické krmičky ryb je dovedena až do stavu praktické použitelnosti a prošla testováním ve skutečné chovné stanici, popis jednotky řízení servomotorů je pouze koncepční; k jeho praktické realizaci dosud nedošlo.

Abstract

This thesis is dealing with design and construction of embedded systems. It consists of two separate parts; in the first one, a design, construction and testing of a control unit for an automatic fish feeder is presented (including a design of the feeder itself). In the second one, a concept of a control unit for a model helicopter is described. Both of those devices are designed in a similar manner and have a lot of common properties. Both units are autonomous intelligent modules, connected to an industrial-standard bus (i.e. RS-485 and CANBUS respectively). Both devices also use servomechanisms as their actuators. While the fish feeder design is a completed project, including testing results, the model helicopter control unit is presented only as a conceptual idea for future development work.

Poděkování

Na tomto místě bych rád poděkoval všem, kdož mi pomáhali a podporovali mě v práci zde popisované. Jmenovitě panu Dr. Ing. Zdeňku Hanzálkovi, svému vedoucímu, za rady a vedení této práce, jenž přestože směroval mou činnost ke konkrétnímu cíli, zachoval mi dostatečnou tvůrčí svobodu k uplatnění vlastních myšlenek a postupů. Dále panu ing. Liboru Waszniovskému za rady při práci se systémem DisCO a panu ing. Liboru Švehlovi z firmy DataPartner za spolupráci při začleňování řízení automatické krmičky do téhož systému. Rád bych též jmenoval pana ing. Pavla Kozáka, Ph.D. a paní Dr. ing. Jitku Hamáčkovou z Výzkumného ústavu rybářského a poděkoval jim za spolupráci a pomoc při vývoji uživatelského rozhraní pro automatickou krmičku a při praktických testech tohoto zařízení. Nakonec děkuji panu ing. Pavlu Burgetovi za pomoc při práci se školním modelem rybochovné stanice.

Obsah

1. Úvod 3
2. Automatické krmení ryb 4
2.1 Úvod 4
2.2 Popis řízené technologie 4
2.2.1 Intenzivní chov ryb 4
2.2.2 Popis modelu chovné stanice 6
2.3 Použitý hardware 8
2.3.1 Sběrnice RS-485 8
2.3.2 Hlavní řídicí počítač 9
2.3.3 Reléový výstupní modul ADAM-4060 9
2.3.4 Chemické senzory 11
2.4 Použitý software 13
2.4.1 Operační systém PharLap ETS 13
2.4.2 Řídicí systém DisCO 18
2.5 Automatická krmička ryb 19
2.5.1 Úvod 19
2.5.2 Mechanická konstrukce 20
2.5.3 Řídicí jednotka 26
2.5.4 Firmware řídicí jednotky 32
2.5.5 Testování automatického krmícího systému 38
2.5.6 Integrace ovládání automatických krmiček do řídicího systému DisCO 40
2.5.7 Možná zlepšení systému 40
2.6 Shrnutí 41
3. Automatické řízení vrtulníku 42
3.1 Úvod 42
3.2 Popis řízeného systému 42
3.3 Popis řídicího systému 44
3.3.1 Úvod 44
3.3.2 Bloková struktura řídicího systému 46
3.4 Použitý hardware 47
3.4.1 Hlavní řídicí počítač 47
3.4.2 Jednotky bezdrátového spojení se zemí 47
3.4.3 Elektronický kompas 48
3.4.4 Jednotka GPS 48
3.4.5 Výškoměr 48
3.4.6 Jednotka inerciální navigace 49
3.4.7 Jednotka řízení servomotorů 49
3.4.8 Pozemní stanice 53
3.4.9 Použité modelářské vybavení 53
3.5 Matematické modely a systémový popis vrtulníku 54
3.5.1 Úvod 54
3.5.2 Systémový popis vrtulníku 54
3.6 Způsoby automatického řízení vrtulníku 57
3.6.1 Úvod 57
3.6.2 Popis jednotlivých typů regulátorů 57
3.7 Shrnutí 59
4. Závěr 60
5. Seznam použité literatury 61

6. Přílohy 63
6.1 Základy aerodynamiky a mechaniky letu vrtulníku 63
6.1.1 Úvod 63
6.1.2 Základní aerodynamika 63
6.1.3 Dynamika vrtulníku 66
6.1.4 Aerodynamika vrtulníku 69
6.1.5 Mechanika letu vrtulníku 73
6.1.6 Závěr 81
6.2 Výpis zdrojového kódu firmwaru automatické krmičky ryb 82
6.3 Uživatelský manuál k automatické krmičce ryb 105
6.4 Výsledky experimentů automatického krmení ryb 111
6.5 Hodnocení automatické krmičky od odborníka na chov ryb 123
6.6 Obsah přiloženého CD-ROM 127

1. Úvod

Jak z názvu této práce vyplývá, budeme se v ní zabývat převážně tzv. vestavěnými systémy (z angl. embedded systems). Pod tímto na první pohled poněkud obskurním pojmem rozumíme takové řídicí systémy, které jsou trvale „vestavěny“ přímo v řízeném zařízení a tvoří tak jeho nedílnou součást. Vestavěným systémem je například řídicí jednotka motoru automobilu či řídicí systém inteligentního senzoru.

Tato práce je složena ze dvou zcela nezávislých částí, jejichž společným jmenovatelem je fakt, že se obě zabývají nějakým vestavěným systémem. V první z nich je popsáno automatické krmící zařízení pro ryby, určené k použití v intenzivním velkochovu. Zařízení je řešeno jako inteligentní autonomní modul, připojený na sběrnici RS-485. Vestavěným systémem je jeho elektronická řídicí jednotka. V části druhé je rámcově popsán řídicí systém modelu vrtulníku, který by měl umožnit jeho automatické řízení. Jedná se vlastně o hardwarovou platformu, na níž by mohl běžet libovolný zatím blíže nespecifikovaný řídicí algoritmus.

Tato dvě témata byla zvolena především proto, že jejich řešení bylo potřebné v rámci dvou projektů, v nichž se katedra řídicí techniky v současné době angažuje. Katedra je zapojena do projektu IFIBO, jenž se zabývá vývojem komplexního řídicího systému pro plně automatizovanou rybochovnou stanici. Pro tento projekt bylo potřeba navrhnout a postavit prakticky použitelnou automatickou krmičku ryb a integrovat ji do budovaného řídicího systému. Protože se v nedávné době také začal rozjíždět projekt zabývající se automatickým řízením modelu vrtulníku, bylo třeba provést rešerši na toto téma a začít se zabývat návrhem struktury použitelného řídicího systému. Proto bylo toto téma též začleněno do obsahu předkládané práce, ačkoli ta se tím stává na první pohled možná poněkud nekonzistentní.

Část první, zabývající se automatickým krmením ryb, je hlavním tématem této práce. V jejím úvodu je popsán princip intenzivního chovu ryb, model automatické rybochovné stanice a struktura použitého řídicího systému. Jeho hardwarové i softwarové komponenty jsou později popsány podrobněji, aby bylo vidět, do jakého prostředí je třeba automatickou krmičku začlenit a aby tak bylo možno lépe pochopit požadavky na ni kladené. Následuje kapitola věnovaná automatické krmičce ryb. Zabývá se všemi aspekty její konstrukce a popisuje praktické testy a zkušenosti s automatickým krmením. V závěru této části jsou shrnuty dosažené výsledky.

V části druhé jsou nejprve velice stručně popsány hlavní aspekty vrtulníkového letu. Dále jsou rámcově popsány požadavky kladené na řídicí systém vrtulníku, společně s blokovým schématem navrhovaného řešení a popisem jeho předpokládaných komponent. Jednou z nich je i jednotka řízení servomotorů, uvedená v zadání této práce. Nakonec jsou stručně nastíněny způsoby systémového popisování vrtulníku a možnosti použití různých typů regulátorů k jeho automatickému řízení.

K této práci je také připojeno značné množství příloh. Jejich začlenění jsem považoval za nezbytné, protože obsahují mnoho informací velmi podstatných pro plné pochopení řešené problematiky a není možno uvést je pouze odkazem, jelikož nejsou nikde dostupné. Většinu z nich (části 6.1, 6.2 a 6.3) jsem sepsal sám, části 6.3 a 6.5 obsahující zhodnocení a výsledky praktických testů automatických krmiček byly sepsány pracovníky Výzkumného ústavu rybářského ve Vodňanech, kde testování probíhalo.

2. Automatické krmení ryb

2.1 Úvod

V této části podrobně popíšeme první řešený problém, kterým je návrh a realizace malokapacitního automatického krmícího zařízení pro ryby, včetně jeho integrace do již existujícího řídicího systému pro automatizovaný chov ryb. Takovéto zařízení je schopno ušetřit značnou část lidské práce v chovné stanici a také snížit spotřebu krmiva, díky přesnějšímu dávkování a minimálním ztrátám oproti ručnímu krmení. Zajímavé je také posouzení vlivu automatického krmení na rychlost růstu ryb a jeho srovnání s ručním krmením.

V první kapitole popíšeme princip intenzivního chovu ryb a model chovné stanice, postavený pro demonstrační účely na naší škole. Dále jsou popsány komponenty použité na stavbu tohoto modelu a software použitý pro jeho řízení. Nakonec následuje podrobný popis a technická dokumentace k automatické krmičce ryb a výsledky zkušebního provozu tohoto zařízení a experimentů s ním prováděných.

2.2 Popis řízené technologie

2.2.1 Intenzivní chov ryb

2.2.1.1 Úvod

Intenzivní chov ryb je poměrně moderní zemědělská technologie, která usiluje o produkci maximálního množství tržních ryb na co nejmenším prostoru při minimálních nákladech na výrobu. Význam pojmu „intenzivní“ si nejlépe vysvětlíme na příkladu z klasického zemědělství. Když chce klasický (extenzivní) zemědělec zvýšit svoji produkci, přikoupí nová pole a oseje větší plochu, kterou obdělává stejně jako předtím. Chce-li svou produkci zvýšit zemědělec intenzivní, snaží se zlepšit své pracovní postupy – začne lépe hnojit, zaseje lepší odrůdy pěstovaných rostlin apod. Přitom pracuje na stále stejné ploše.

Podobně také klasický (extenzivní) chov ryb se vyznačuje tím, že ryby jsou chovány v rybnících podle staleté empirie chovatelů, za minimálního použití moderní techniky a s velikým podílem lidské práce. Extenzivní rybochovná stanice svou produkci zvýší jedině tak, že rozšíří plochu svých rybníků.

Naproti tomu pro chov intenzivní je typické, že ryby nejsou chovány v rybnících, ale v chovných nádržích, ve kterých se snažíme udržovat životní podmínky optimální pro daný druh ryb. V chovných nádržích proto udržujeme optimální teplotu vody, optimální množství kyslíku ve vodě rozpuštěného, optimální pH a další parametry. Ryby krmíme vhodným krmivem v přesně odměřených dávkách a ve vhodných časových intervalech, udržujeme optimální poměr dne a noci (pomocí umělého osvětlení) atd. Protože se jedná o klasický regulační úkol, otevírá se zde široké pole pro nasazení moderní řídicí techniky. Jejím použitím se snažíme odbourávat lidskou práci, a tím zlevnit provoz stanice a také minimalizovat možnost lidské chyby.

Velmi významnou roli při nasazování automatizace v zemědělství ovšem hraje její cena – finanční možnosti rybářů jsou omezené a je třeba brát v úvahu návratnost této investice. Proto se snažíme navrhnout systém robustní a spolehlivý, ale především cenově přijatelný, aby jeho nasazení v praxi nebylo předem odsouzeno k nezdaru.

2.2.1.2 Hydrochemické parametry vody

Jak je uvedeno výše, snažíme se o dosažení co nejlepších životních podmínek v chovných nádržích. Kvalita životního prostředí ryb je závislá především na hydrochemických parametrech okolní vody. Nedodržení optimálních parametrů má za následek snížení produkce (rozumí se přírůstek biomasy, tj. živé hmotnosti ryb za jednotku času), a v extrémním případě může vést až k úhynu chovaných ryb. Nejdůležitější hydrochemické parametry jsou tyto:

- Teplota – optimální se zpravidla pohybuje v intervalu 20-30 °C, úhyn nastává typicky při teplotách menších než 10 °C nebo větších než 40 °C.
- pH – optimální zpravidla 6,5-8,5 , úhyn pro pH menší než 5 a větší než 10.
- O_2 – (kyslík rozpuštěný ve vodě), optimální mezi 5-10 mg/l, úhyn zpravidla nastává pro koncentrace menší než 3 mg/l a větší než 25 mg/l.
- NH_3 – (amoniak), jeho množství se snažíme minimalizovat.

Výše uvedené optimální hodnoty u jednotlivých parametrů jsou pouze orientační, přesnější hodnoty závisí samozřejmě na druhu chovaných ryb a druh od druhu se mohou lišit.

Amoniak se ve vodě vyskytuje buď ve formě rozpuštěného plynu NH_3, který je pro ryby toxický, nebo disociovaný ve formě kationtů NH_4^+. Je to produkt rybího metabolismu a proto jeho obsah ve vodě stoupá vždy po krmení. Jeho množství ve vodě se snažíme minimalizovat, abychom zabránili otravě.

Podobně také časový průběh množství kyslíku rozpuštěného ve vodě je korelován s časy krmení a množstvím krmiva; spotřeba kyslíku roste při a po krmení, vlivem zvýšené aktivity ryb při krmení a trávení po něm. Pokud není ve vodě rozpuštěno dostatečné množství kyslíku, ryby nemohou přijímat potravu a krmení je tak neefektivní a v extrémním případě dokonce nebezpečné, protože nespotřebované krmivo rozkládající se ve vodě představuje riziko znečištění. Při poklesu úrovně kyslíku je proto nutné krmení okamžitě přerušit.

2.2.1.3 Popis chovného systému

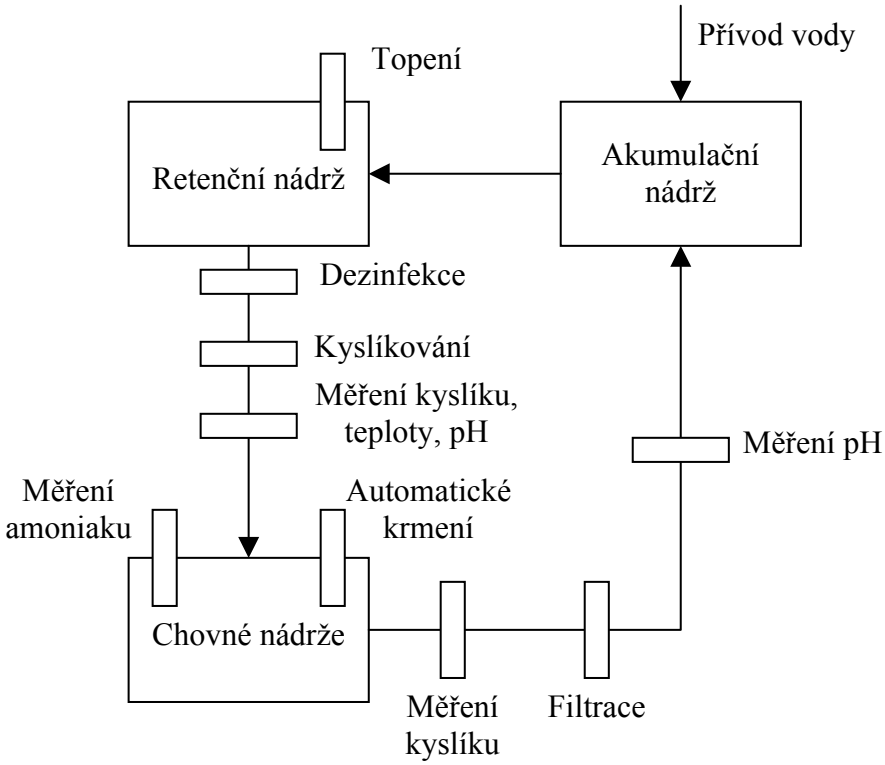
Funkce chovného systému určeného pro intenzivní chov ryb je založena na cirkulaci vody v uzavřeném okruhu. Voda je médiem pro přenos surovin (surovinami jsou zde myšleny především krmivo a kyslík) a odvod zplodin (zplodiny představuje především amoniak) a samozřejmě také prostředím pro život ryb.

Hlavní částí systému jsou **chovné nádrže**, ve kterých žijí chované ryby. V nich se snažíme udržovat optimální životní podmínky, a ostatní části systému slouží tomuto účelu.

Do chovných nádrží je přiváděna čerstvá a vhodně upravená voda (tj. voda o vhodné teplotě, pH, vhodně prokysličená a zbavená amoniaku) z takzvané **retenční nádrže**, ve které probíhá ohřev vody. Z chovných nádrží je voda odváděna přes čistící filtr do takzvané **akumulační nádrže**, kde jsou doplňovány ztráty vody (způsobené především výparem v chovných nádržích) z vnějšího zdroje. Mezi retenční nádrží a chovnými nádržemi se v potrubí nachází zařízení pro okysličování vody a její dezinfekci (pomocí UV záření). V chovných nádržích je umístěno zařízení pro automatické krmení.

Hydrochemické parametry vody jsou měřeny v následujících bodech: Teplota na vstupu chovných nádrží, množství kyslíku ve vodě na vstupu a výstupu chovných nádrží, pH na vstupu akumulační nádrže a na vstupu chovných nádrží a množství amoniaku uvnitř chovných nádrží.

Schematický nákres chovného systému je přehledně znázorněn na obrázku 2.1.



Obr. 2.1 – Schéma chovného systému

2.2.2 Popis modelu chovné stanice

2.2.2.1 Popis technologické části

Jako názorná ukázka funkce celého systému intenzivního chovu ryb byl v naší laboratoři postaven model chovné stanice. Používá se především pro demonstrační účely na různých prezentacích a výstavách. Ukazuje všechny hlavní části a funkce chovné stanice a jeho chovná nádrž může být pro maximální názornost dokonce obsazena živými akvarijními rybami.

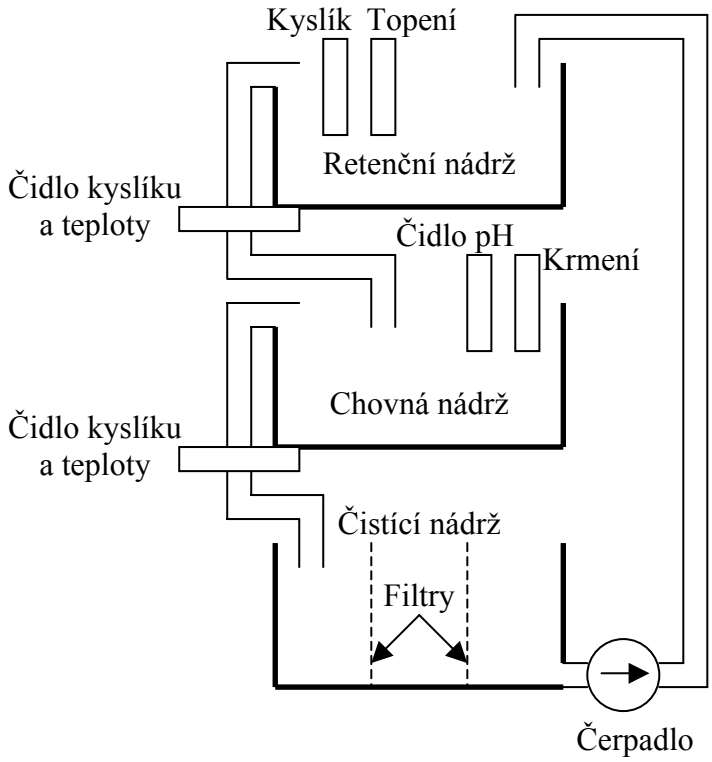
Model se skládá ze tří nádrží umístěných nad sebou, s podobnými funkcemi jako u skutečné technologie. Nejvýše je umístěna retenční nádrž, ve které je voda ohřívána a okysličována. Retenční nádrž je přepadovým odtokem spojena s prostřední chovnou nádrží, ve které žijí chované ryby. Poslední, spodní nádrž slouží jako čistící a s chovnou nádrží je spojena taktéž přepadem. Obsahuje mechanický a biologický filtr. Mechanický filtr zbavuje vodu makroskopických nečistot a je tvořen porézní látkou, např. molitanem, přes kterou je voda čerpána. Biologický filtr slouží k dezinfekci vody. Z této nádrže je voda čerpána zpět do nádrže retenční.

Cirkulace vody v systému je zajišťována akvarijním čerpadlem, které je schopno přečerpat přibližně 300 l/h. Voda je ohřívána dvěma 100W topnými tělesy, umístěnými v retenční nádrži. Toto řešení umožňuje třípolohovou regulaci vytápění. K okysličování vody je použito běžné akvarijní vzduchovadlo, umístěné rovněž v retenční nádrži. Ke krmení ryb je použita automatická krmička (viz. 2.5), umístěná nad chovnou nádrží.

Ve výtoku retenční nádrže je umístěno čidlo pro měření množství kyslíku rozpuštěného ve vodě (viz. 2.3.4.3). V chovné nádrži je čidlo pH (viz. 2.3.4.4) a ve výtoku z ní druhé čidlo kyslíku. Všechna zmiňovaná čidla jsou zároveň vybavena teploměry, protože jejich údaje jsou tepelně závislé a je nutno provádět korekci chyby měření. Proto je současně používáme k měření teploty. Obě čidla kyslíku jsou záměrně umístěna ve výtokových hadicích, protože k jejich správné funkci je nutný alespoň minimální pohyb okolní vody. Model není bohužel vybaven žádným senzorem pro měření množství ve vodě rozpuštěného amoniaku.

Celý model je vestavěn do rozebíratelné lehké konstrukce z pozinkovaného hliníku. Nádrže jsou umístěny nad sebou a propojeny hadicemi. Veškerá elektronika vyjma řídicího počítače je vestavěna do dvou vodotěsných rozvodných krabic přišroubovaných nad chovnou nádrží. Hlavní řídicí počítač je umístěn ve zvláštní nerezové krabici, která má na přední straně displej s touch panelem, na kterém běží vizualizace a pomocí kterého je možné celý systém ovládat. Tato krabice je umístěna ve vrchní části konstrukce. Automatická krmička je přišroubována k hliníkové konstrukci nad chovnou nádrží.

Nákres modelu chovné stanice je na obrázku 2.2.



Obr. 2.2 – Nákres modelu chovné stanice

Na závěr podotkněme, že některé akční členy nejsou kvůli svému malému výkonu schopny významně ovlivnit životní prostředí v chovné nádrži. Týká se to především topení. Ukázalo se, že dvě 100W tělesa jsou na zhruba 230l vody příliš slabá a teplotu vody v systému jimi téměř nelze ovlivnit. Taktéž akvarijní vzduchovadlo se ukázalo být příliš málo výkonným. Z tohoto důvodu budou tyto prvky v budoucnu nahrazeny výkonnějšími.

2.2.2.2 Popis řídicího systému

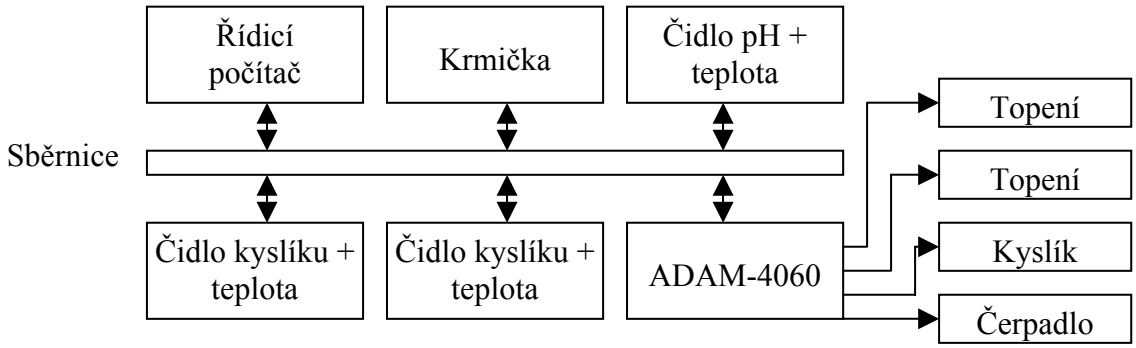
Řídicí systém modelu má sběrnicovou topologii s řízením přístupu metodou Master-slave. K propojení všech komponent je použita průmyslová sběrnice RS-485. Přenosová rychlost po sběrnici je pouze 9600 b/s, což ovšem není na závadu, vezmeme-li v úvahu velmi

malý datový tok a dlouhé časové konstanty senzorů (například jen měření obsahu kyslíku trvá asi 0,5s).

Master celé sběrnice je hlavní řídicí počítač (viz. 2.3.2), na kterém běží vlastní řídicí algoritmus a vizualizace. Master řídí veškeré dění na sběrnici a je jediný, kdo může iniciovat přenos. Podřízené stanice smí pouze odpovídat na dotazy zařízení master a vykonávat jeho povely, samy od sebe nikdy nesmí začít vysílat. Tak je zajištěno, že na sběrnici nedochází ke konfliktům. Každá stanice na sběrnici je jednoznačně identifikována svojí adresou.

Na hlavním řídicím počítači běží SCADA systém DisCO (viz. 2.4.2) firmy DataPartner pod operačním systémem PharLap ETS (viz. 2.4.1) firmy VenturCom. Systém DisCO se stará o řízení celého systému včetně vizualizace a interakce s uživatelem.

Kromě hlavního řídicího počítače je na sběrnici připojen reléový výstupní modul ADAM-4060 (viz. 2.3.3), který řídí obě topná tělesa, čerpadlo vody a vzduchovadlo. Všechna tato zařízení jsou řízena pouze připojením/odpojením napájecího napětí. Dalšími stanicemi na sběrnici jsou čidla kyslíku a pH a poslední stanicí je automatická krmička. Blokové schéma systému je na obrázku 2.3.



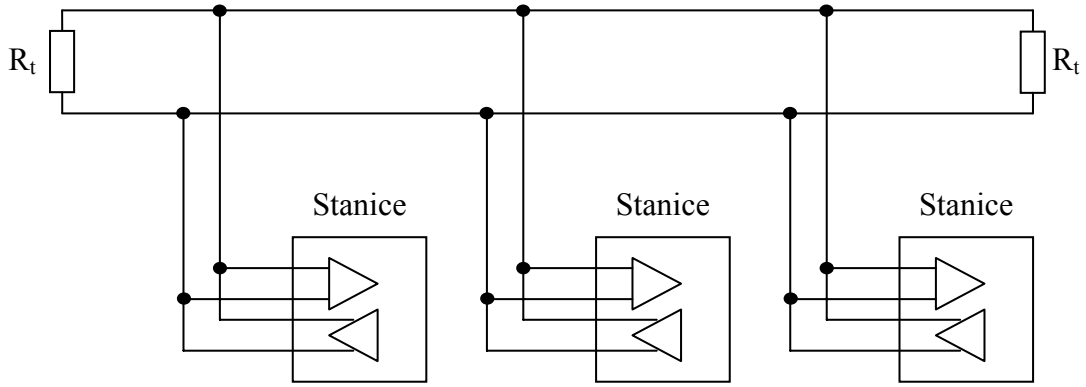
Obr. 2.3 – Blokové schéma zapojení modelu

2.3 Použitý hardware

2.3.1 Sběrnice RS-485

Sběrnice RS-485 patří mezi nejrozšířenější průmyslové standardy. Jako fyzickou vrstvu ji využívá mnoho používaných přenosových systémů, mj. například PROFIBUS. Její přesné specifikace jsou obsaženy v normě EIA RS-485.

Přenosovým médiem bývá nejčastěji kroucená dvoulinka UTP či stíněná kroucená dvoulinka STP. Ta je na obou koncích impedančně přizpůsobena rezistory o odporu 120Ω. K této lince jsou paralelně připojeny jednotlivé stanice, které zpravidla mají každá svůj vysílač a přijímač. Blokové schéma je na obrázku 2.4.



Obr. 2.4 – Blokové schéma sběrnice RS-485

Jak vysílač, tak přijímač jsou vždy vybaveny třístavovým vstupem/výstupem, který umožňuje nastavit stav vysoké impedance a stanici tak od sběrnice odpojit. Maximální počet přijímačů připojených k sběrnici je 32. Z tohoto důvodu musí být každý vysílač schopen budit alespoň 32 přijímačů.

Každá stanice mívá zpravidla svůj přijímač trvale připojený ke sběrnici a vysílač ve stavu vysoké impedance. Vysílač připojuje pouze tehdy, chce-li vysílat a má ho připojen pouze po dobu relace. Po jejím ukončení se okamžitě odpojí, aby sběrnici neblokovala. V jednom okamžiku smí vysílat vždy maximálně jedna stanice, ostatní poslouchají. Z toho plyne, že přenos po RS-485 je vždy typu half-duplex – stanice může v jednu chvíli buď jen vysílat nebo jen přijímat, nikdy oboje současně.

Vysílač sběrnice RS-485 musí být schopen odolávat trvalému zkratu na výstupu a musí být odolný vůči stejnosměrné složce. Z tohoto důvodu se často používá optické oddělení stanic od linky.

Maximální přenosová rychlost po sběrnici RS-485 je 10Mb/s, ovšem dosažení tak vysoké rychlosti v průmyslových podmínkách bývá často nemožné a mnohdy nebývá ani potřebné, proto jsou obvyklé v praxi používané přenosové rychlosti mnohdy až o několik řádu nižší. Například našem případě je to 9600b/s.

Více informací o sběrnici RS-485 lze najít například v [12].

2.3.2 Hlavní řídicí počítač

Jako hlavní řídicí počítač systému je použito průmyslové PC firmy ADVANTECH PCA-6145L. Je postaveno na bázi procesoru 486, nicméně pro danou aplikaci se jeho výkon jeví jako postačující.

Základní deska je vybavena integrovanou grafickou kartou, řadičem pevných i pružných disků, sériovým a paralelním portem a rozhraním pro myš a klávesnici (případně touchpad). Dále obsahuje tzv. DiscOnChip, což je vlastně Flash disc, použitelný jako harddisk. Má 32MB operační paměti a 128kB cache. Deska je vybavena programovatelným watchdogem.

V našem případě je tento počítač dále rozšířen o ethernetovou síťovou kartu a dále převodník RS-485/RS232, připojený na sériovém portu. Přes ten je řídicí počítač připojen na vnější sběrnici.

Více informací lze získat na stránkách výrobce [8].

2.3.3 Reléový výstupní modul ADAM-4060

2.3.3.1 Úvod

Modul ADAM-4060 slouží k ovládání čerpadla, vzduchovadla a obou topných těles. Jedná se o inteligentní reléový modul, který je připojen na sběrnici RS-485 a podle povelů z ní přijímaných spíná jednotlivá výstupní relé, čímž ovládá připojené prvky. Typ 4060 disponuje celkem čtyřmi výstupními relé. Tato relé mohou spínat střídavá napětí do 250V/0,3A či stejnosměrná do 110V/1,6A. Moduly ADAM-4060 se napájejí stejnosměrným napětím 10-30V.

Více informací o modulu ADAM-4060 lze získat na webových stránkách výrobce, firmy ADVANTECH [8].

2.3.3.2 Komunikační protokol přístroje ADAM-4060

Modul ADAM-4060 používá ke komunikaci po RS-485 jednoduchý znakově orientovaný komunikační protokol (tj. komunikace probíhá ve znacích ASCII). Komunikace je typu master-slave.

Z parametrů komunikace je softwarově nastavitelná pouze přenosová rychlost (baud-rate). Moduly typu ADAM podporují přenosové rychlosti od 1200 do 38400bps (bites per second). Datový rámec je pevný, s délkou 10 bitů na jeden bajt – 1 startbit, 8 datových bitů a 1 stopbit. Parita se nepoužívá.

Každé zařízení na sběrnici je jednoznačně identifikováno svojí dvouznakovou adresou, která může nabývat hodnot 0-255 (00-FF). Komunikaci iniciuje vždy master, a to dotazem nebo povelem k zařízení. Oslovené zařízení musí odpovědět do vypršení časového limitu, jinak je pokládáno za porouchané.

Všechny povely (neboli telegramy) mají podobný formát, který vypadá takto:

[řídicí znak][adresa][povel][kontrolní součet][cr]

Řídicí znak je buď %,$ nebo # a rozlišuje se podle něj typ telegramu. Následuje dvouznaková adresa zařízení, potom příslušný povel včetně parametrů, nepovinný kontrolní součet a nakonec znak cr (carriage return), označující konec telegramu. Odpověď zařízení má podobný formát, který je dobře patrný z následujícího seznamu povelů.

Možné povely pro ADAM-4060 jsou tyto:

- %AANNTTCCFF(cr) – Konfigurace zařízení. % je řídicí znak znamenající konfiguraci, AA jsou 2 znaky adresy zařízení (v hexadecimálním tvaru), NN jsou 2 znaky nové adresy, TT je identifikace typu modulu, pro typ 4060 je implicitně 40. CC je přenosová rychlost – 03 znamená 1200bps, 04 je 2400bps, 05 je 4800bps, 06 je 9600bps, 07 je 19200bps a konečně 08 znamená 38400bps. FF je nastavení kontrolního součtu – 00 znamená kontrolní součet zakázán, 64 kontrolní součet povolen. Po přijetí a úspěšném vyhodnocení povelu přístroj odpoví !AA(cr), což znamená povel úspěšně přijat a dekódován, případně odpoví ?AA(cr), došlo-li k chybě. Podle nové konfigurace začne přístroj pracovat až po resetu. Konfigurační příkaz pracuje pouze tehdy, je-li svorka INIT uzemněna. **Příklad:** %0012400600(cr) znamená nastavení přístroje s adresou 00 na novou adresu 12, přenosová rychlost 9600bps, kontrolní součet zakázán. Přístroj odpoví !00(cr).
- $AA6(cr) – dotaz na stav výstupů. $ je řídicí znak značící dotaz, AA je adresa přístroje, 6 kód povelu. Přístroj odpoví !(výstup)0000(cr), kde (výstup) je řetězec dvou znaků (tedy osmibitové číslo, 00 až FF), určující stav čtyř výstupních relé (jejich stav je určen stavem bitů 0 až 3 přijatého čísla). **Příklad:** $006(cr) znamená dotaz na stav výstupů přístroje na adrese 00. Přístroj odpoví například !020000(cr), což znamená, že je sepnuto druhé relé a ostatní jsou rozepnuta.
- #AABB(data)(cr) – povel nastavení výstupů. # je řídicí znak značící povel, AA je adresa přístroje. BB jsou řídicí znaky – pokud chceme nastavit všechny výstupy najednou je BB rovno 00, pokud chceme nastavit jen jeden výstup, je první znak 1 a druhý určuje číslo relé, jenž chceme nastavit – pro modul 4060 tedy 0 až 3. (data) je dvouznakové osmibitové hexadecimální číslo. Pokud nastavujeme všechna relé najednou (BB je 00), potom bity 0 až 3 čísla (data) určují žádaný stav relé (1 sepnuto, 0 rozepnuto). Pokud chceme nastavit jen jedno relé a ostatní ponechat jak jsou, je první znak povinně 0 a druhý znak je 0 nebo 1 (rozepnuto/sepnuto). Přístroj odpoví >(cr), byl-li povel platný, případně ?AA(cr), byl li povel neplatný. **Příklad:**

$001301(cr) znamená sepnout relé číslo 3 přístroje na adrese 00 a ostatní ponechat tak jak jsou. Přístroj odpoví >(cr), což znamená povel přijat.

- #** - povel synchronizovaného vzorkování. Na konci tohoto povelu není požadován znak (cr). Po obdržení tohoto povelu si všechny I/O moduly na sběrnici uloží stavy svých vstupů/výstupů do speciálního registru, odkud je lze později vyčíst pomocí povelu $AA4(cr) (který je popsán dále). Na povel #** zařízení nijak neodpovídají.
- $AA4(cr) – čtení dat synchronizovaného vzorkování. $ je řídicí znak značící dotaz, AA je adresa přístroje, 4 je kód povelu. Přístroj odpoví !(status)(data)0000(cr), kde (status) je znak 0 nebo 1 – 0 znamená, že data byla od posledního synchronizačního příkazu odeslána více než jedenkrát, 1 znamená právě jedenkrát. (data) jsou dva znaky určující osmibitové hexadecimální číslo, jehož bity 0 až 3 určují stav výstupních relé v době vzorkování. **Příklad:** $004(cr) znamená čtení synchronizovaných dat z přístroje na adrese 00. Přístroj odpoví například !102(cr), což znamená, že data byla od okamžiku vzorkování odeslána právě jednou a relé 1 (pouze bit 1 je nastaven v číslu 02) je sepnuto, ostatní relé jsou rozepnuta.
- $AA2(cr) – dotaz na aktuální konfiguraci zařízení. $ je řídicí znak značící dotaz, AA je adresa přístroje, 2 je kód povelu. Přístroj odpoví !AATTCCFF(cr), kde AA je adresa zařízení, TT je jeho typ (pro typ 4060 je to vždy 40, což značí digital I/O modul), CC je kód přenosové rychlosti (stejný systém jako u konfiguračního příkazu) a konečně FF je pro ADAM 4060 buď 1 (kontrolní součet zakázán) nebo 65 (kontrolní součet povolen). **Příklad:** $002(cr) znamená dotaz na aktuální konfiguraci přístroje na adrese 00. Přístroj odpoví například !00400601(cr), což znamená, že má adresu 00, je to typ digital I/O, přenosová rychlost 9600bps a přesný typ 4060 s vypnutým kontrolním součtem.
- $AA5(cr) – dotaz na reset status (tj. zjištění, jestli zařízení provedlo reset od posledního přijatého příkazu $AA5(cr)). $ je řídicí znak značící dotaz, AA je adresa přístroje, 5 je kód povelu. Přístroj odpoví !AAS(cr), kde AA je adresa zařízení a S je znak buď 0 nebo 1 – 0 v případě, že modul nebyl od posledního povelu resetován, 1 v případě, že byl resetován. **Příklad:** $005(cr) znamená dotaz na reset status přístroje na adrese 00. Přístroj odpoví například !001(cr), což znamená, že od obdržení posledního povelu $AA5(cr) byl resetován.
- $AAF(cr) – dotaz na verzi firmwaru zařízení. $ je řídicí znak značící dotaz, AA je adresa přístroje, F je kód povelu. Přístroj odpoví !AA(verze)(cr), kde (verze) je číslo verze firmwaru zařízení. **Příklad:** $00F(cr) znamená dotaz na verzi firmwaru přístroje na adrese 00. Přístroj odpoví například !004(cr), což znamená, že má firmware verze 4.
- $AAM(cr) – dotaz na jméno zařízení. $ je řídicí znak značící dotaz, AA je adresa přístroje, M je kód povelu. Přístroj odpoví !AA(jméno)(cr), kde (jméno) je řetězec znaků, určující ID zařízení. **Příklad:** $00M(cr) znamená dotaz na jméno přístroje na adrese 00. Přístroj odpoví například !004060(cr), což znamená, že má ID 4060.

2.3.4 Chemické senzory

2.3.4.1 Úvod

Jako senzory hydrochemických parametrů vody (v našem případě pouze pH vody a množství ve vodě rozpuštěného kyslíku) byly použity přístroje firmy GRYF, konkrétně GRYF 9401 (čidlo kyslíku) a GRYF 9202 (čidlo pH). Jedná se o mikroprocesorem řízené

inteligentní sondy, schopné komunikovat po sběrnici RS-485, což je vlastnost pro nás zvlášť výhodná.

Oba přístroje jsou vybaveny odporovým teploměrem Ni 1000, jehož údaj se využívá jednak pro korekci měření příslušného chemického senzoru (tu provádí přístroj sám), jednak ho využívá náš systém pro samostatné měření teploty. Teplotní rozsah obou přístrojů, ve kterém je zaručena jejich správná funkce, je 0-40 °C, což je pro naše účely dostatečné.

Přístroje GRYF mají příkon pouze 3W a lze je napájet střídavým napětím 12-24V AC nebo 12-35V DC. Náš systém využívá jednotného napájení 24V DC pro všechny přístroje.

Bližší informace o obou sondách lze získat na webových stránkách výrobce [13].

2.3.4.2 Komunikační protokol přístrojů GRYF

Obě sondy používají pro komunikaci po RS-485 stejný komunikační protokol. Ten je formálně shodný s protokolem přístrojů ADAM řady 4000 (viz. 2.3.3.2) a proto ho zde nebudeme znovu popisovat. Uvedeme pouze seznam povelů přípustných pro přístroje GRYF:

- %AANNTTCCFF(cr) – Konfigurace zařízení. % je řídicí znak znamenající konfiguraci, AA jsou 2 znaky adresy zařízení (v hexadecimálním tvaru), NN jsou 2 znaky nové adresy, TT jsou neplatné znaky a mohou být jakékoli – jsou tu pouze pro zachování kompatibility s ADAMy. CC je přenosová rychlost – 03 znamená 1200bps, 04 je 2400bps, 05 je 4800bps a 06 je 9600bps. Vyšší rychlosti přístroje GRYF nepodporují. FF je nastavení kontrolního součtu – 00 znamená kontrolní součet zakázán, 40 kontrolní součet povolen. Po přijetí a úspěšném vyhodnocení povelu přístroj odpoví !AA(cr), což znamená povel úspěšně přijat a dekódován, případně odpoví ?AA(cr), došlo-li k chybě. Podle nové konfigurace začne přístroj pracovat až po resetu. Konfigurační příkaz pracuje pouze tehdy, jsou-li zkratovány svorky S17 a S18. **Příklad:** %0012000600(cr) znamená nastavení přístroje s adresou 00 na novou adresu 12, přenosová rychlost 9600bps, kontrolní součet zakázán. Přístroj odpoví !00(cr).
- $AAM(cr) – dotaz na jméno zařízení. $ je řídicí znak značící dotaz, AA je adresa přístroje, M je kód povelu. Přístroj odpoví !AA(jméno)(cr), kde (jméno) je řetězec znaků, určující ID zařízení. **Příklad:** $00M(cr) znamená dotaz na jméno přístroje na adrese 00. Přístroj odpoví například !00G9202(cr), což znamená, že má ID G9202.
- #AAN(cr) – čtení vstupu. # je řídicí znak značící povel, AA adresa zařízení, N číslo 0-9, které určuje číslo kanálu. Na kanálu 0 je vždy hlavní měřená veličina (kyslík, pH), na kanálu 1 pomocná veličina (teplota). Přístroj odpoví >(data)(cr), kde (data) mají následující formát: RXXXX, kde R je číslo rozsahu přístroje a XXXX je dekadické číslo (naměřená hodnota). Podle čísla rozsahu R se do naměřené hodnoty XXXX doplní desetinná čárka. **Příklad:** #120(cr) je dotaz na naměřený údaj přístroje s adresou 12 na kanále 0. Odpověď je například >01055 (cr), což znamená naměřená hodnota 1055 na rozsahu 0, tedy 10,55.

Přístroje GRYF navíc disponují celou škálou chybových hlášek, které můžeme dostat v odpověď na dotaz na naměřenou hodnotu. Tyto hlášky se liší podle typu přístroje a čísla kanálu. Pro GRYF 9202 a kanál 0 (pH) jsou to:

- 0-ERR – překročení dolní meze rozsahu
- 00ERR – překročení horní meze rozsahu

Pro GRYF 9401 a kanál 0 (kyslík):

- 0TERR – teplota pro kompenzaci naměřené hodnoty mimo rozsah
- 0-ERR – malá úroveň vstupního signálu
- 00ERR – příliš velká úroveň vstupního signálu

Při měření teploty (u obou přístrojů kanál 1) přichází v úvahu pouze jedna chybová hláška, a to:

- 00ERR – teplota mimo rozsah

2.3.4.3 Čidlo rozpuštěného kyslíku

Jako čidlo ve vodě rozpuštěného kyslíku je použit přístroj GRYF 9401, který je vybaven senzorem Clarkova typu. Clarkův senzor je principielně založen na měření konduktivity elektrolytu obsahujícího analyzovanou látku mezi dvěma elektrodami. K měření se zpravidla používá stejnosměrného napětí. Bližší informace o Clarkovu senzoru lze nalézt například v [14].

Přístroj GRYF 9401 dokáže měřit množství ve vodě rozpuštěného kyslíku v rozsahu od 0 do 80mg/l s maximální relativní odchylkou ±0,5%. Měření je korektní při teplotách od 0 do 40°C. Přístroj je dále schopen měřit teplotu v rozsahu –50–200°C s maximální odchylkou ±0,2°C.

2.3.4.4 Čidlo pH

Jako čidlo pro měření pH vody je použit přístroj GRYF 9202, jež k měření využívá tzv. potenciometrické metody. Princip měření pH touto metodou je založen na měření koncentrace iontů v roztoku. Měřící zařízení se skládá ze dvou elektrod, z referenční elektrody ponořené v pufrovém roztoku o známém pH a z měřící elektrody ponořené v roztoku měřeném. Protože potenciál elektrody je závislý na koncentraci iontů v roztoku a potažmo tedy na jeho pH, mají obě elektrody různý potenciál. Rozdíl těchto potenciálů (tedy napětí mezi elektrodami) měříme. Více o měření pH touto metodou se lze dočíst například v [14].

Přístroj GRYF 9202 je schopen měřit pH v rozsahu 0 až 14 s maximální odchylkou ±0,01. Měření je korektní při teplotách 0–100°C. Pro korekci teplotní závislosti měřené hodnoty je vybaven stejným teploměrem jako čidlo GRYF 9401.

2.4 Použitý software

2.4.1 Operační systém PharLap ETS

2.4.1.1 Úvod

PharLap ETS (Embedded Tool Suite) je operační systém reálného času s aplikačním rozhraním kompatibilním s Win32 API. Používá se především pro náročné aplikace ve vestavěných systémech, využívajících průmyslová PC na bázi procesorů x86. Toto řešení umožňuje oproti jednočipovým průmyslovým počítačům či PLC vytváření složitých a výpočetně náročných aplikací přímo na řídicích počítačích, s vestavěnou vizualizací a GUI (graphic user interface).

Operační systém PharLap ETS je uzpůsoben běhu na relativně méně výkonném hardware (oproti běžným stolním PC), zachovávajíce však při tom všechny výhody Win32 programování. Tato vlastnost patří mezi jeho největší přednosti – umožňuje totiž tvorbu

aplikací pomocí standardních Windowsovských nástrojů, například Microsoft Visual C++, a s využitím většiny funkcí jeho standardních knihoven.

2.4.1.2 Základní vlastnosti PharLap ETS

PharLap ETS je systém multitaskový, nikoli však multiprocesní. To znamená, že v ETS existuje pouze jeden jediný proces kromě jádra, který ovšem může být složený z více vláken (threads). Základná rozdíl je v tom, že vlákna sdílejí více systémových prostředků (mj. například adresový prostor) než procesy, a proto při jejich přepínání připadá pouze minimální režie na přepínání kontextu (při přepínání vláken se ukládá a obnovuje se mnohem méně dat než při přepínání procesů). Proto v ETS scheduler (plánovač) ubírá vláknům mnohem méně strojového času než například ve Windows.

ETS má množství volitelných komponent. Mezi nejvýznamnější patří síťová komunikace po TCP/IP, webový server MicroWeb, který umožňuje snadno vystavit aplikaci na internetu a využít ji jako dálkové rozhraní a knihovna PEG, která obsahuje sadu funkcí pro práci s GUI (grafické uživatelské rozhraní).

Jádro ETS se skládá z tzv. monitoru a pomocných knihoven. **Monitor** je bootovací část systému, která se nahraje do paměti automaticky pro resetu počítače. Monitor nejprve inicializuje veškerý hardware a potom buď spustí aplikaci nebo očekává příkazy z nadřízeného systému. Knihovny, přilinkované k monitoru, tvoří vlastně API operačního systému. Monitor společně s knihovnami a aplikací tvoří ve výsledku jeden celek, který je nahraný v pevné paměti počítače (na harddisku, flashdisku, případně v ROM).

2.4.1.3 Real-time scheduling

Real-time scheduling neboli rozvrhování v reálném čase zajišťuje část jádra ETS zvaná **scheduler**. Vzhledem k požadavkům kladeným na real-time operační systém musí být scheduler striktně deterministický, tj. dá se vždy bezpečně určit, které vlákno v danou chvíli poběží a kdy přesně přijde řada na další.

ETS využívá systém dynamických priorit jednotlivých vláken. Vláknu je přiřazena priorita, která se dynamicky zvyšuje podle času, po kterou je vlákno neaktivní, a naopak se zmenšuje v době, kdy je vláknu přidělen procesor. Tímto způsobem se také systém brání proti deadlockům – může se například stát, že vlákno s vyšší prioritou, které má přiděleno procesor, čeká na sdílený prostředek rezervovaný neaktivním vláknem s nižší prioritou. V tomto případě se tomuto neaktivnímu vláknu postupně zvyšuje priorita až do chvíle, kdy na něj přijde řada a je mu přidělen procesor, a vlákno může tudíž dokončit svou činnost a sdílený prostředek uvolnit.

Frekvence přepínání mezi vlákny se stejnou prioritou je nastavitelná a délka její periody se nastavuje v celočíselných násobcích periody systémových hodin, což je u PC standardně 55 μs. Implicitně je nastavena na 10 ms.

2.4.1.4 Synchronizace vláken

Synchronizace vláken je v ETS zajišťována pomocí čtyř mechanizmů:

- **Mutex** – Mutex (neboli kritická sekce) je objekt, jímž se koordinují přístupy ke sdílenému zdroji. V principu je mutex vlastně binární semafor (viz. dále). V ETS na rozdíl od Win32 pojmy Mutex a kritická sekce splývají[1].

[1] Ve Win32 slouží mutexy ke sdílení zdrojů mezi vlákny různých procesů, kritická sekce je určena jen pro vlákna jednoho procesu. Jelikož v ETS existuje pouze jeden proces, rozdíl mezi mutexem a kritickou sekcí se stírá. Z důvodů kompatibility s Win32 API jsou ovšem v ETS zachovány obě metody.

- **Semaphore** (semafor) – Semafor je v principu jednoduchý objekt, mající jednu číselnou proměnnou a dvě metody – operace up a down. Hodnota semaforu určuje, kolik jednotek strojového času ještě vláknu zbývá. Při volání metody down se nejprve zkontroluje, není-li hodnota semaforu 0 a pokud ne, dekrementuje se a vlákno smí pokračovat. Pokud byl semafor nulový, vlákno se deaktivuje. Operace up nejprve zkontroluje, jestli před semaforem nečekají neaktivní vlákna. Pokud ano, jedo vybere a umožní mu dokončit operaci down, při níž bylo deaktivováno. Pokud žádná vlákna nečekají, inkrementuje operace up hodnotu semaforu. Obě operace (up a down) jsou součástí jádra OS[1] a probíhají atomicky (nedělitelně). Více informací o semaforech lze najít v [15].
- **Event** (událost) – Slouží k synchronizaci dvou procesů, kdy jeden proces využívá data, vyprodukovaná druhým procesem. Mechanizmus události zaručí, že čekající proces se aktivuje, až když jsou data k disposici.
- **Interlocked variable acces** (uzamčený přístup k proměnným) – slouží k synchronizaci přístupu vláken ke globálním proměnným. OS zaručuje, že když jeden proces operuje s nějakou globální proměnnou, je tato pro všechny ostatní procesy nepřístupná.

2.4.1.5 Mezivláknová komunikace

Pro účely vzájemné komunikace mezi jednotlivými vlákny se používají tzv. **roury** (podobně jako například v Unixu). Roury v ETS jsou jednosměrné, tj. na jednom konci lze pouze číst a na konci druhém pouze zapisovat. Každý konec roury přísluší právě jednomu vláknu. Každá roura je jednoznačně identifikována pomocí svého **handle**. Ten je rouře přidělen jádrem OS v době jejího vzniku. Roury ze zakládají pomocí metod jádra OS.

2.4.1.6 Systém souborů

Operační systém PharLap ETS používá systém souborů plně kompatibilní s MS-DOS. Může používat systémy FAT12, FAT16 a FAT32. ETS podporuje všechna standardní IDE zařízení, jak „klasické“ rotační harddisky (včetně CHS a LBA formátů), tak flashdisky. Mimo IDE zařízení Podporuje také floppy mechaniky, M-Systems flash a RAM-disk.

2.4.1.7 Ošetření vyjímek

Zpracování vyjímek v ETS je velmi podobné systému používanému ve Win32. ETS podporuje jak strukturované „Windowsovské“ vyjímky, tak klasické vyjímky jazyka C++. Vyjímky se „metají“ i ošetřují pomocí standardních C++ funkcí.

2.4.1.8 Práce s dynamickými knihovnami

S dynamickými knihovnami se v ETS pracuje pomocí standardních Win32 API nástrojů. Knihovny v ETS používají, podobně jako spustitelné soubory, klasický Widows Portable Executable formát.

2.4.1.9 Podpora TCP/IP

V dnešní době se stále více prosazují vestavěná zařízení, využívající jako komunikační médium síť Ethernet. Toto řešení má mnoho výhod, od možností použití standardních a dobře osvědčených přenosových protokolů existujících pro tento typ sítě a široce rozšířených

[1] OS = Operační Systém

vývojových prostředků pro ně až po snadné připojení zařízení na síť Internet. Z tohoto důvodu disponuje také ETS podporou TCP/IP protokolu.

Pro ETS existují ovladače pro všechny nejrozšířenější typy síťových karet od různých výrobců. ETS podporuje linkové protokoly Ethernet, slip a PPP. V síťové a linkové vrstvě používá ETS funkcí WinSock API, takže je možno implementovat v aplikační vrstvě jakýkoli protokol využívající TCP/IP nebo UDP protokol. V aplikační vrsvě jsou podporovány protokoly HTTP, FTP, SMTP, SNMP a DHCP.

2.4.1.10 MicroWeb Server

MicroWeb je jednoduchý webový server, používaný v ETS k účelům snadného připojení zařízení k internetu a k tvorbě prostředků vzdálené vizualizace a vzdálené správy zařízení.

MicroWeb podporuje jak statické, tak dynamicky vytvářené HTML stránky. Je kompatibilní s většinou používaných prohlížečů (Microsoft Internet Explorer, Netscape Navigator aj.).

2.4.1.11 Grafické uživatelské rozhraní

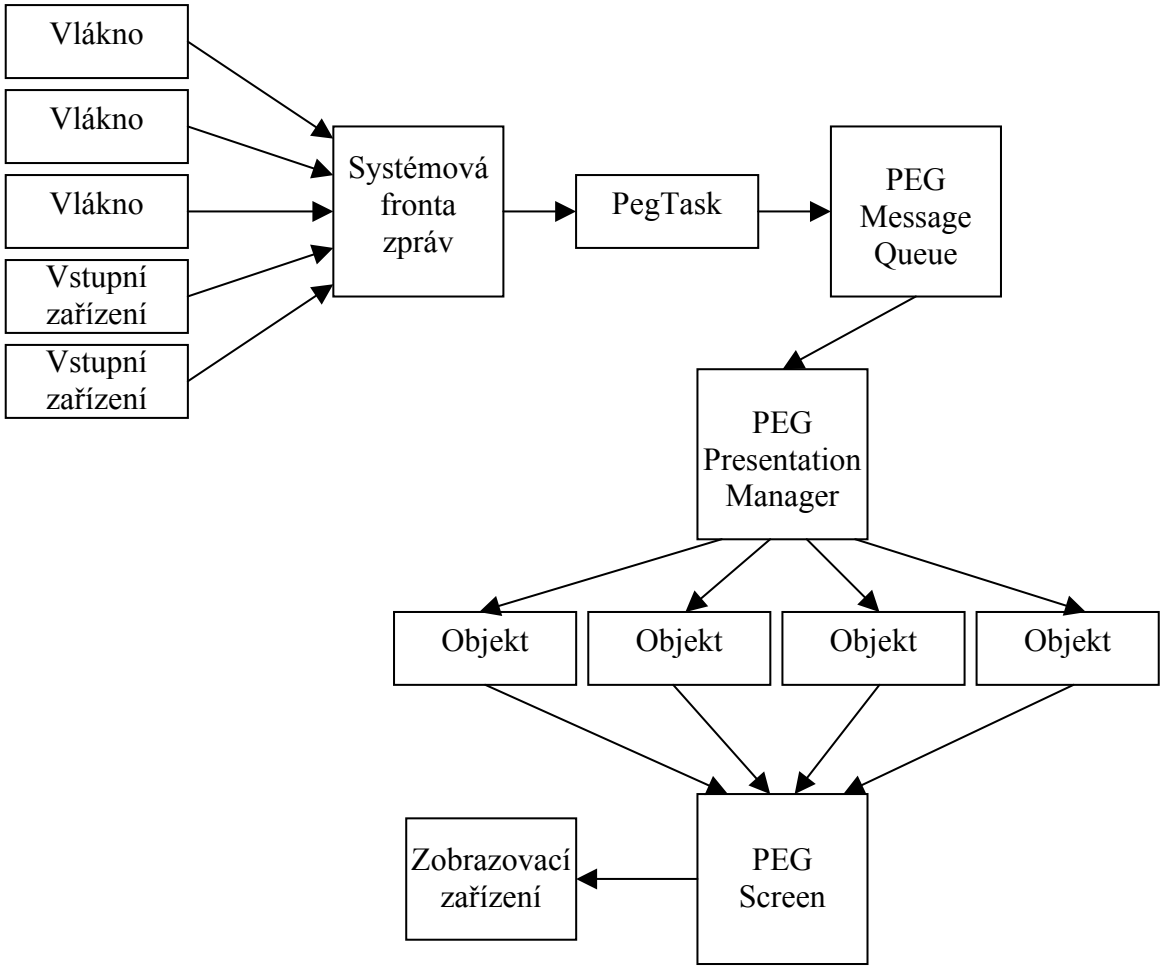
Pro účely tvorby grafického uživatelského rozhraní se v ETS používá komponenta PEG (Portable Embedded GUI). PEG se skládá ze čtyř hlavních částí:

- **PegTask** – PegTask je rozhraní mezi JOS[1] a PEGem. JOS se na GUI dívá jen jako na další vlákno s nízkou prioritou, a zachází s ním odpovídajícím způsobem – tj. přiděluje mu strojový čas pomocí scheduleru jako kterémukoli jinému vláknu. PegTask reprezentuje toto vlákno.
- **PegMessageQueue** – je fronta zpráv typu FIFO, sloužící pro zasílání zpráv mezi grafickými objekty a OS.
- **PegPresentationManager** – Rozhoduje o viditelnosti jednotlivých grafických objektů, přiděluje jim **focus** (tj. vybírá objekt, který potom dostává informace ze vstupů) a pamatuje si stavy jednotlivých objektů.
- **PegScreen** – přistupuje přímo na vykreslovací zařízení (pomocí příslušného ovladače) a zajišťuje vykreslování jednotlivých objektů. PegSreen je vlastně abstraktní objektová třída obsahující virtuální metody pro kreslení různých grafických objektů. Tyto metody jsou definovány podle možností ovladače konkrétního zobrazovacího zařízení.

Běh GUI je v ETS řízen pomocí systému zpráv, podobně jako je tomu ve Windows. Zprávy mohou přicházet od JOS, od vstupních zařízení nebo od jiných objektů PEGu a obsahují různé povely pro objekty, jako například „smaž se," „překresli se," „změň velikost" apod. O předávání zpráv se stará systémová fronta zpráv (OS Message Queue). Zprávy určené objektům PEG z ní vybírá PegTask, který je předává do PegMessageQueue. Odtud si je vybírají jednotlivé grafické objekty (ve Windows je pojem „grafický objekt" ekvivalentní pojmu „okno").

Funkce celého systému je nejlépe patrná z obrázku 2.5.

[1] JOS = Jádro Operačního Systému



Obr. 2.5 – Schéma předávání zpráv v ETS

2.4.1.12 Vývojové prostředí

PharLap ETS využívá jako vývojového prostředí Microsoft Visual Studio. Do něj je pouze třeba doinstalovat podpůrný balík, zvaný TNT Embedded tool suite. Ten se skládá ze dvou hlavních částí – Embedded Studio Express a LinkLoc.

Embedded Studio Express je přídavná komponenta k Microsoft developer studiu, umožňující především komunikaci s cílovým počítačem pro aplikaci (zvaným **target**). Komunikací se myslí především nahrávání zkompilované aplikace do targetu, kontrola jejího běhu v targetu (krokování pro účely ladění) aj.

LinkLoc je linker pro sestavování vložených (embedded) programů.

2.4.1.13 Shrnutí

Tato kapitola je pouze přehledová a nepopisuje zdaleka všechny aspekty systému PharLap. Detailnější informace lze nalézt v [16], s jejímž použitím byla tato kapitola zpracována.

2.4.2 Řídicí systém DisCO

2.4.2.1 Úvod

Řídicí systém DisCO patří do široké rodiny tzv. SCADA[1] systémů. Zkratka DisCO znamená Distributed Computing Objects a popisuje vlastně systémem používanou technologii – distribuovanou objektově orientovanou architekturu. Tato technologie je rozšířením klasického objektově orientovaného přístupu a je pro ni typické, že objekty jsou distribuovány na různé výpočetní stanice, propojené pomocí nějakého vhodného komunikačního média, zpravidla sítě ethernet.

Podle [17] je DisCO kompaktní maximálně obecný a otevřený řídicí systém, schopný běhu na libovolném počítači postaveném na bázi některého z procesorů řady x86. Je schopen běhu pod jakýmkoli operačním systémem vybaveným Win32 API. Je možné použít jej jako samostatný softwarový produkt, jako komponentu či soubor knihovních funkcí pro vlastní program a nebo jej doplnit vlastními součástmi.

2.4.2.2 Hlavní vlastnosti systému DisCO

Podle [17] se systém DisCO vyznačuje především těmito vlastnostmi:

- Data jsou strukturovaně distribuována do celého systému na pozadí jeho běžné činnosti.
- Aplikační skripty nebo povely obsluhy nad distribuovanými objekty mohou být vykonávány na kterékoli pracovní stanici systému.
- Možnost horké zálohy kterékoli stanice.
- Operátorské stanice systému používají GUI (grafické uživatelské rozhraní).
- Procesní stanice systému zabezpečují řízení nebo jiné akce v reálném čase. Z hlediska objektové komunikace jsou přímo integrovány do systému.
- Externí procesní stanice jsou připojovány standardními komunikačními protokoly.

2.4.2.3 Použití systému DisCO k řízení modelu chovné stanice

Jeden z nejefektivnějších způsobů využití systému DisCO je připojení vlastní nadstavbové aplikační dynamické knihovny, obsahující například regulátor pro řízený systém. To je přesně způsob, který využíváme i my k řízení našeho modelu chovné stanice. Zjednodušeně řečeno, necháváme plně na starosti systému DisCO, aby zajišťoval sběr dat, vizualizaci a komunikaci s uživatelem a komunikaci s akčními členy a připojíme k němu pouze vhodný regulátor, a to ve formě dynamické knihovny. Tato knihovna potom běží společně s Run-time modulem systému DisCO na hlavním řídicím počítači. Principielně je tento způsob využití systému DisCO načrtnut na obrázku 2.6 (podle [18]).

Obr. 2.6 – Princip komunikace připojeného regulátoru se systémem DisCO

[1] Supervisory Control And Data Acquisition – jak již název napovídá, jedná se o systémy dohledu, řízení a sběru dat.

Systém DisCO dodává regulátoru data ze systému, ať už data naměřená či zadaná uživatelem. Původ dat je jednoznačně identifikován jejich signaturou, jejich typ je definován v tzv. datovém modelu. Regulátor potom generuje povely a události pro systém DisCO.

Vývoj aplikační knihovny je prováděn v Microsoft Visual C++. Z pohledu uživatele systému DisCO se vlastně jedná o vytváření standardní windowsovské dynamické knihovny, využívající služeb systému DisCO.

2.5 Automatická krmička ryb

2.5.1 Úvod

V této kapitole se budeme zabývat hlavním tématem této diplomové práce, kterým je návrh a konstrukce automatické krmičky ryb. Důvodů pro nasazení automatických krmiček v chovné stanici je mnoho. Hlavní z nich jsou vypsány níže:

- Krmení patří mezi pracné a časově náročné každodenní činnosti v chovné stanici. Při ručním krmení je nutné provést následující činnosti: Naměřit krmnou dávku, nasypat ji do chovné nádrže a ručně zadat do řídicího systému, v kolik hodin, kolik krmiva a kam bylo dodáno; tuto činnost je třeba zopakovat pro každou chovnou nádrž zvlášť. Vezmeme-li v úvahu, že chovných nádrží může být ve skutečné stanici až několik desítek, je zřejmé, kolik práce a času tato činnost zabere. V malé experimentální chovné stanici představuje krmení pro jednoho pracovníka asi dvě hodiny práce denně. Ve skutečné stanici by to bylo ještě mnohem více.
- Krmivo patří mezi nejdražší vstupy technologie vůbec. Je proto velice důležité zacházet s ním co nejhospodárněji, tj. omezit jeho ztráty při různém přesýpaní a odvažování a dávkovat ho co nejpřesněji, aby ryby byly krmeny optimálně, ale přitom se krmivem zbytečně neplýtvalo. Při použití automatických krmiček jsou ztráty krmiva menší a dávky přesnější než při ručním krmení.
- Ryby lze krmit častěji a menšími dávkami než při ručním krmení, protože krmit například po celý den každou hodinu ručně by bylo pro člověka neúměrně pracné – ovšem při použití automatických krmiček to lze uskutečnit bez problémů. Tento způsob krmení (kratší časové intervaly a menší krmné dávky) by teoreticky měl být výhodnější než krmit například jen 3x denně většími dávkami.
- Automatická krmička je integrální součástí řídicího systému a ten tak má dokonalý přehled a kontrolu nad distribucí krmiva do jednotlivých chovných nádrží. Údaje o krmení už není třeba zadávat ručně, čímž se nejen šetří lidská práce, ale také vylučuje možnost chyby obsluhy.
- Automatická krmička zajistí krmení bez přítomnosti obsluhy, tj. i pokud v celé chovné stanici není nikdo přítomen.

Proto se začala hledat možnost, jak zajistit krmení ryb automaticky. Na automatickou krmičku, použitelnou v intenzivní chovné stanici, je kladeno několik základních požadavků:

- Musí být integrovatelná do existujícího řídicího systému. Je proto vhodné, aby se jednalo o autonomní stavebnicový modul, schopný připojení na sběrnici RS-485.
- Musí mít dostatečnou kapacitu (cca 0,5 kg krmiva), aby byla schopna zajistit krmení jedné chovné nádrže bez doplňování po rozumě dlouhou dobu (cca 1-2 dny). Přitom musí být schopná dávkovat krmivo dostatečně přesně a v malých dávkách (nejmenší dávka asi 10g, relativní odchylka do 15%).

- Musí být spolehlivá, a to i v agresivním vlhkém prostředí.
- Musí být cenově dostupná (její cena by se měla pohybovat v řádu jednotek tisíců korun).

Na začátku celého projektu jsem prozkoumal možnosti nasazení sériově vyráběných automatických krmiček různých typů. Těchto zařízení existuje na trhu celá řada, od malokapacitních, určených například pro akvaristy, až po velkokapacitní krmičky určené třeba pro krmení krav. Jmenujme namátkou například Automatic fish feeder A785 firmy Hagen Nutrimatic[1], Eheim 3581 stejnojmenného výrobce[2], Pond 21 fish feeder či třeba F14 fish feeder, oba výrobky firmy Animate[3]. Po důkladném prozkoumání možností těchto (a mnoha dalších) zařízení se ovšem ukázalo mnoho jejich nevýhod.

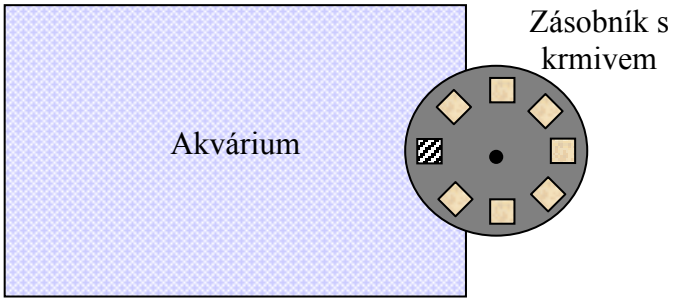
Hlavní z nich je fakt, že ať už bychom použili kteréhokoli z nich, bylo by třeba jej nejprve upravit tak, aby bylo lze připojit jej ke sběrnici RS-485 jako autonomní zařízení, schopné komunikace s nadřízeným systémem. Mimoto levné akvaristické krmičky nemají zpravidla dostatečnou kapacitu, aby bylo možné použít je ke krmení celé chovné nádrže (nemluvě vůbec o tom, že tato zařízení v žádném případě nemají průmyslové parametry, především pak spolehlivost). Naproti tomu velkokapacitní zařízení nejsou dostatečně přesná, k čemuž ještě přistupuje fakt, že bývají velmi drahá. Z těchto důvodů bylo rozhodnuto vyvinout krmičku vlastní, odpovídající našim požadavkům.

V následujících odstavcích je tato krmička dopodrobna popsána. V části 2.5.2 je rozebrána její mechanická konstrukce, následuje popis řídicí jednotky (část 2.5.3), dále popis programu řídicí jednotky (část 2.5.4) a na závěr jsou uvedeny výsledky praktických zkoušek zařízení a jeho možná budoucí zlepšení (části 2.5.5 a 2.5.7).

2.5.2 Mechanická konstrukce

2.5.2.1 Možnosti řešení

Na počátku návrhových prací bylo třeba vymyslet vhodnou mechanickou konstrukci krmícího zařízení. Možných řešení existuje mnoho. Po prozkoumání sériově vyráběných akvaristických zařízení se ukázalo, že s oblibou využívají struktury načrtnuté na obrázku 2.7 (pohled shora).



Obr. 2.7 – Struktura akvarijní krmičky

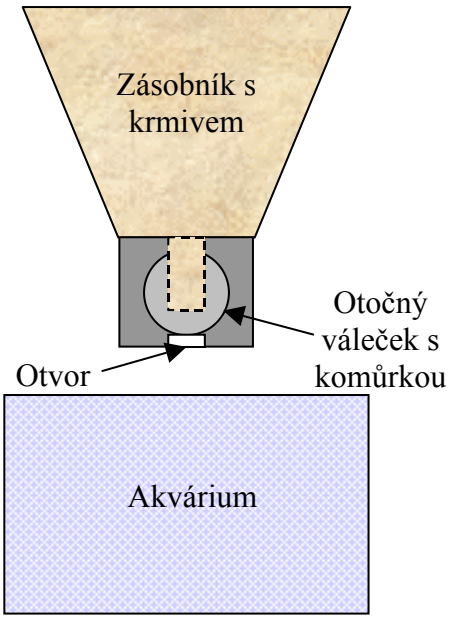
Zásobník s krmivem je tvořen válcem, otočným kolem své osy. V tomto válci jsou podélné komůrky naplněné krmivem (jedna komůrka představuje jednu dávku). Dolní podstava válce se otáčí po neprostupné podložce, ve které je pouze v jednom místě (nad

[1] http://www.hagen.com
[2] http://www.eheim.com
[3] Popis lze nalézt např. na http://www.wetpetusa.com

akváriem) vyvrtán otvor (na obrázku 2.7 vyznačen šrafovaně). Natočí-li se komůrka s krmivem nad tento otvor, krmivo vlastní vahou propadne do akvária.

Nevýhody tohoto uspořádání jsou na první pohled zřejmé. První z nich je, že počet krmných dávek je předem pevně stanoven, včetně jejich velikosti; to silně omezuje flexibilitu krmného zařízení. Dalším problémem je poměrně obtížné doplňování krmiva spojené s relativně vysokými ztrátami (rozsypání). Kapacita akvarijních zařízení tohoto typu je velmi malá, takže jejich použití nepřicházelo v úvahu.

Další možné uspořádání krmného zařízení je znázorněno na obrázku 2.8. Krmivo je umístěno v zásobníku, odkud může propadávat do komůrky, vyvrtané v otočném válečku. Když se váleček otočí, uzavře se přívod krmiva a odměřená dávka, daná velikostí komůrky, vypadne do akvária.



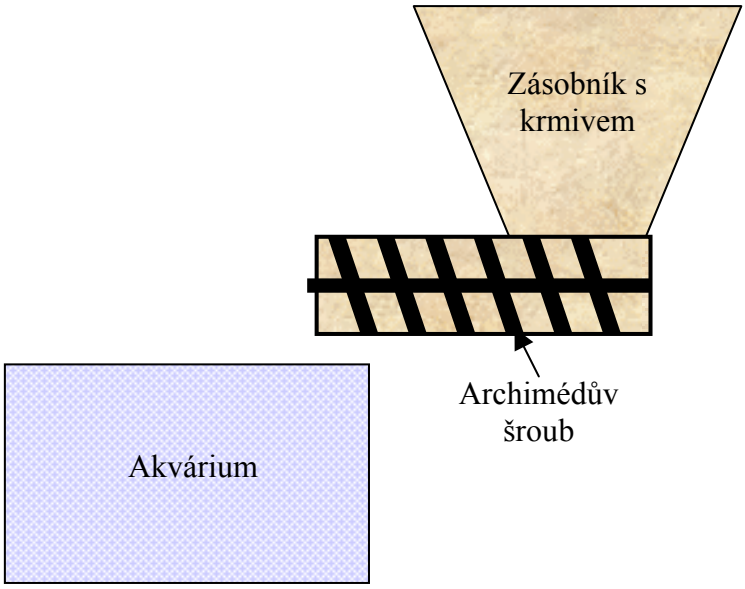
Obr. 2.8 – Struktura krmičky

S konstrukcí znázorněnou na obrázku 2.8 jsem nějaký čas experimentoval, protože se zprvu jevila jako relativně slibná. Její hlavní výhodou je snadné a relativně bezeztrátové doplňování krmiva a možnost montáže velkokapacitního zásobníku. Další výhodou je, že výrobně složité zařízení s otočným válečkem s komůrkou, otáčejícím se v pevné kostce, je na trhu běžně k dostání – velmi podobnou koncepci totiž používají šoupátkové karburátory k modelářským spalovacím motorům, rozdíl je pouze v tom, že u nich je otvor válečkem průchozí. Úprava takového karburátoru byla snadná, ovšem při praktických zkouškách záhy vyšly najevo též stinné stránky tohoto uspořádání.

Mezi ně patří především fakt, že volba velikosti otvoru v otočném válečku je silně závislá na granularitě krmiva (pro velké granule musí být otvor větší, ale tím se zase snižuje přesnost dávkování), čímž zařízení ztrácí na univerzálnosti. Dále se ukázalo, že při použití krmiva s velmi malou granularitou (prášek, který je silně lepivý), není krmivo schopné propadávat vlastní vahou do komůrky a bylo by nutné je nějakým způsobem pěchovat. Zařízení by se tak stalo příliš složitým, nehledě k malé „robustnosti“ a spolehlivosti. Proto bylo od této koncepce posléze upuštěno.

Další velmi slibnou koncepcí se jevilo použití krmičky s archimédovým šroubem (obr. 2.9), podobným, jako je například v mlýnku na maso. Krmivo je umístěno v zásobníku, který může mít značnou kapacitu, z nějž propadává do trubky, v níž se otáčí archimédův šroub. Ten

zajišťuje dopravu krmiva trubkou a počtem jeho otáček je dáno množství ven vytlačeného krmiva. Stejná struktura je použita například u velkokapacitní automatické krmičky krav.



Obr. 2.9 – Struktura krmičky s archimédovým šroubem

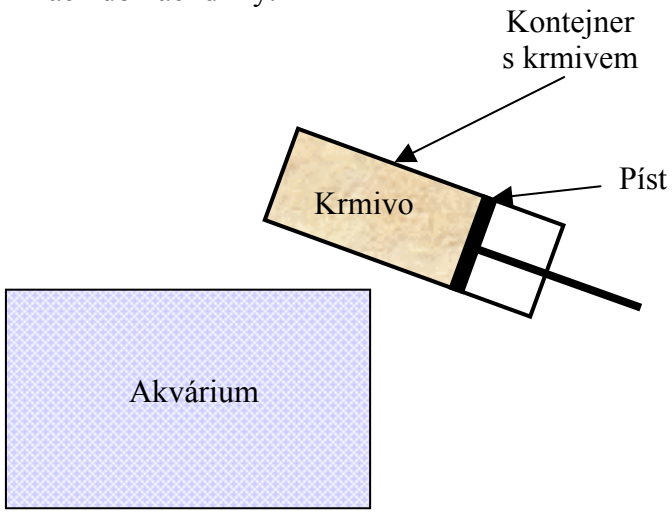
Tato koncepce má několik zásadních výhod. Jednak je to možnost použití velkokapacitního zásobníku krmiva, jenž lze nadto velmi snadno a bezeztrátově doplňovat (dokonce by bylo možné zautomatizovat i doplňování krmiva). Další nespornou výhodou je „robustnost“ celé koncepce oproti předchozím návrhům.

Ovšem i tento koncept má své stinné stránky. Dávkování krmiva pomocí archimédova šroubu není příliš přesné, zvláště tehdy, pokud šroub v trubce není dostatečně těsný. Dále je tu značná výrobní složitost a tím i cena zařízení – vyrobit archimédův šroub, a nadto vyrobit jej velmi přesně, není snadná ani levná záležitost. Ukázalo se, že použití sériově vyráběných zařízení tohoto typu není dost dobře možné, protože např. mlýnek na maso zdaleka nemá dostatečně těsný šroub, a proto na přesné dávkování krmiva s ním není ani pomyšlení. Použití krmičky pro krávy také není reálné, protože ta v žádném případě není schopná odměřovat malé dávky s námi požadovanou přesností. Další nevýhodou je také fakt, že tento typ krmičky může krmivo drtit, čímž dochází k jeho znehodnocení.

Společným jmenovatelem všech předchozích zařízení je propadávání krmiva systémem shora dolů, a to pouze za pomoci gravitace. Tento systém není příliš robustní, protože krmivo bývá nezřídka lepivé a mnohdy se někde uchytí a něco ucpe, takže potom nepropadává, jak by mělo. Vedoucí této práce, pan Dr.Ing. Zdeněk Hanzálek, přišel nakonec se spásným nápadem – proč nezkusit vytlačovat krmivo *zdola nahoru*, místo abychom spoléhali na jeho propadávání shora dolů? Podle této myšlenky vzniknul poslední zde představovaný koncept, který byl nakonec skutečně použit. Jeho princip je znázorněn na obrázku 2.10.

Systém sestává z nádoby (kontejneru), ve které je krmivo. Dno této nádoby je pohyblivé (je tvořeno pístem, který se může touto nádobou volně pohybovat) a horní konec nádoby je otevřený. Tato nádoba je upevněna nad akváriem, míříce šikmo vzhůru, tak, aby krmivo nemohlo samovolně vypadávat. Pokud má zařízení krmit, vytlačí příslušnou dávku krmiva pohybem pístu vzhůru. Velikost této dávky je jednoznačně a velmi přesně odměřena velikostí (délkou) posuvu pístu.

Hlavní výhodou této koncepce je její nepopiratelná robustnost – krmivo se nemá kde zachytit a co ucpat. Další výhodou je možnost přesného dávkování velmi malých objemů krmiva nezávisle na jeho tvaru a granularitě. Krmivo nemůže být rozdrceno. Zařízení je též výrobně nenáročné a jeho prototyp a malou ověřovací sérii jsem dokázal postavit i v amatérských podmínkách domácí dílny.



Obr. 2.10 – Struktura krmičky s posuvným pístem

Zařízení má také několik nevýhod. Především je to jeho omezená kapacita a obtížnější systém doplňování krmiva než například u krmičky s archimédovým šroubem. Další nevýhodou je nutnost získávání informace o poloze pístu v nádobě – Protože zařízení už není čistě rotačního charakteru, musíme nějakým způsobem ošetřit hranice pohybu pístu, a k tomu nutně potřebujeme znát jeho polohu. Znalost polohy pístu je také nutná k tomu, abychom věděli, kolik krmiva ještě v krmičce zbývá. Tyto problémy se nicméně podařilo uspokojivě vyřešit a jsem toho názoru, že výhody této koncepce předčí její nedostatky.

2.5.2.2 Popis realizované konstrukce

Jako nejvýhodnější koncepce byla nakonec vybrána struktura, schematicky znázorněná na obrázku 2.10. Skutečná krmička má kontejner obdélníkového průřezu, ve kterém se pohybuje píst. Píst musí být dostatečně těsný, aby krmivo nepropadávalo za něj, nebo aby se nezadíralo mezi píst a stěnu kontejneru (což byl problém, k němuž skutečně zpočátku docházelo). Současně ale nesmí nikde dřít, aby nadměrně nezatěžoval pohonné zařízení.

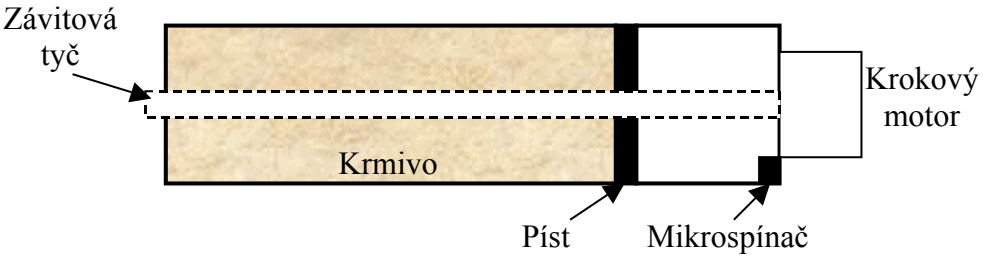
Píst má uprostřed otvor, v němž je vlepena matice M5. Celým kontejnerem je protažena závitová tyč, na které je píst našroubován. Tato tyč se otáčí (pomocí krokového motoru) a závitový převod tak mění rotační pohyb závitové tyče na translační pohyb pístu. Krokový motor je se závitovou tyčí spojen pomocí pružné spojky Oldham HUCO XY19. Tato spojka je díky svojí konstrukci schopna kompenzovat případnou malou nesouosost mezi hřídelí krokového motoru a závitem v pístu. Nákres spojky Oldham HUCO XY19 (převzatý z propagačního materiálu) je na obrázku 2.11.

Použití krokového motoru k pohonu pístu přináší možnost řídit posun pístu v přímé vazbě, protože u krokového motoru máme zajištěno, že vykoná přesně tolik otáček, kolik žádáme (pokud samozřejmě není mechanicky přetěžován). Stačí tedy pouze zjistit výchozí posici pístu po zapnutí krmičky a dále už můžeme řídit bez zpětné vazby. K tomuto účelu slouží mikrospínač, vlepený na konci kontejneru a určující zadní doraz posunu pístu. Řídicí

jednotka krmičky po zapnutí nejprve zacouvá s pístem až na zadní doraz (který je indikován stiskem mikrospínače). Nyní se píst nachází v referenční posici a je vynulováno počítadlo otáček závitové tyče (jež potažmo určuje posici pístu). Každý pozdější návrat pístu do referenční posice je znovu indikován stiskem mikrospínače, což slouží ke korekci chyby řízení bez zpětné vazby.

Schematický nákres krmičky je na obrázku 2.12.

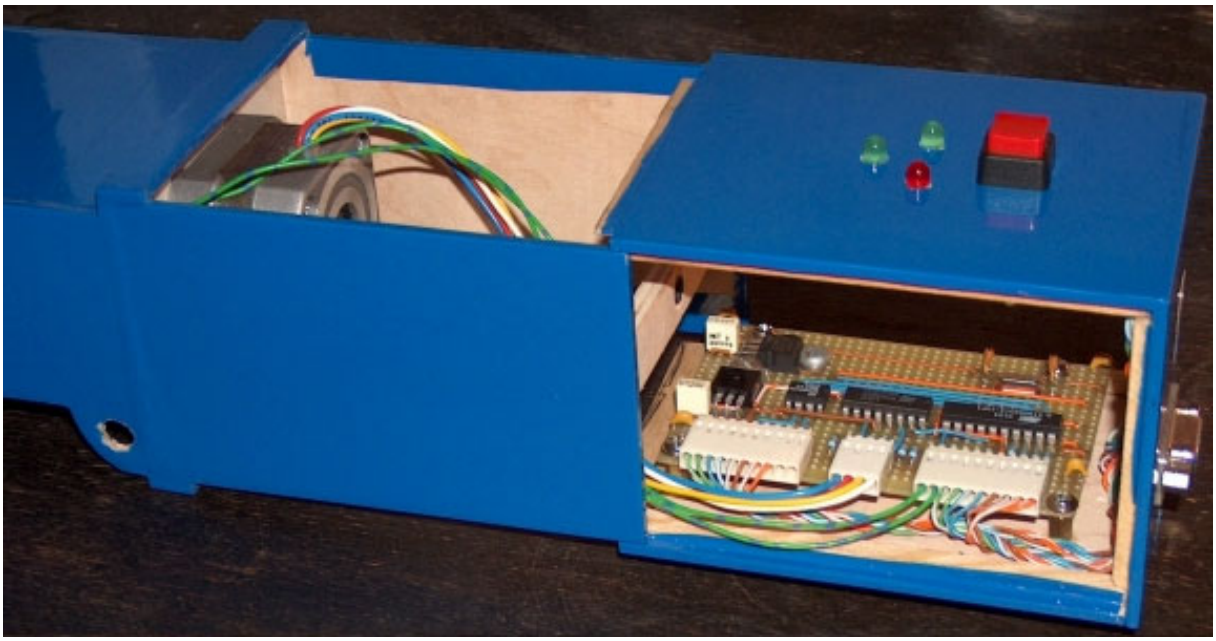
Obr. 2.11 – Spojka Oldham HUCO XY19



Obr. 2.12 – Schéma automatické krmičky ryb

Celé zařízení je postaveno převážně z překližky tl. 3 mm. Píst je vyroben z překližky tl. 6 mm. K ochraně dřevěné konstrukce krmičky proti vodě je použita nažehlovací plastová fólie ORACOVER. Tato fólie se běžně používá k potahování konstrukcí modelů letadel a je mechanicky velmi odolná, otěruvzdorná a vodovzdorná. Toto konstrukční řešení je třeba považovat za provizorní – bylo zvoleno proto, že překližka je materiál snadno dostupný a lehce opracovatelný. To mi umožnilo postavit jak dva vývojové prototypy, tak ověřovací sérii 3 kusů krmiček (pro účely testování v chovné stanici ve Vodňanech) svépomocí v podmínkách domácí dílny. Konstrukce se nicméně ukázala být dostatečně odolná i v náročných podmínkách průmyslového provozu.

Krokový motor je společně s řídicí jednotkou krmičky umístěn pod krytem na konci kontejneru. Tento kryt zaručuje ochranu proti vzdušné vlhkosti i vodě, stříkající z chovné nádrže. Kryt je odnímatelný a umožňuje snadný přístup ke klíčovým elementům zařízení. Čela krytu jsou zajištěna západkami a bočnice vodícími lištami, jenž jednak zabraňují průniku vody, jednak usnadňují přesné nasazování krytu. Kryt je na konci zajištěn polyamidovým šroubem M5. Na vrchním panelu krytu jsou umístěny LED diody, indikující provozní stavy krmičky a tlačítko, sloužící k jejímu manuálnímu ovládání. Ke spodnímu panelu je zevnitř připevněna deska plošných spojů s řídicí jednotkou krmičky. Uchycení desky je realizováno přišroubováním čtyřmi šrouby M3 k mosazným distančním sloupkům, vlepeným do spodního panelu krytu. K zadnímu panelu krytu je přišroubován multifunkční konektor, CANON 9F, sloužící k napájení, komunikaci a diagnostice krmičky. Celý systém je dobře patrný z fotografie na obrázku 2.13.



Obr. 2.13 – Ochranný kryt motoru a řídicí jednotky

Celá krmička je upevněna k chovné nádrži a v provozní posici je zajištěna pomocí aretační tyčky. Po jejím vyjmutí lze krmičku sklopit, na vrchol kontejneru nasadit trychtýř a doplnit krmivo (viz. obrázky 2.14 a 2.15).

Obr. 2.14 – Automatická krmička v provozní posici

Obr. 2.15 – Krmička připravená k doplňování krmiva

Protože mechanické řešení krmičky není hlavní náplní této práce, není v přílohách připojena přesná technická dokumentace strojní části. Je tomu tak především z důvodů udržitelnosti rozsahu této práce v patřičných mezích. Tato podkapitola byla včleněna pouze pro získání povšechného náhledu na princip funkce řízeného mechanizmu.

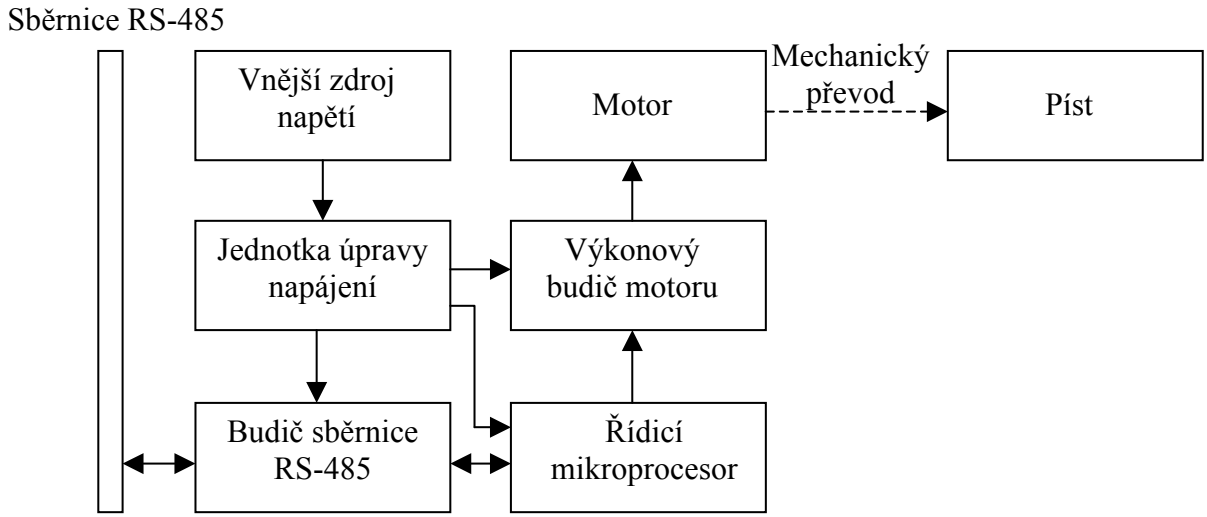
2.5.3 Řídicí jednotka

2.5.3.1 Úvod

V této podkapitole detailně popíšeme první z „vestavěných systémů“ z názvu této práce, kterým je řídicí jednotka automatické krmičky ryb. Protože krmička je autonomní zařízení, připojené na sběrnici RS-485 a schopné poslouchat povelů nadřízeného řídicího systému, je zřejmé, že musí obsahovat poměrně sofistikovanou řídicí jednotku, vybavenou jednočipovým mikropočítačem.

Popišme nejprve blokovou strukturu této jednotky. Její hlavní částí je řídicí mikroprocesor, který zajišťuje všechny „vyšší“ funkce. Ke sběrnici je připojen přes třístavový budič, který zajišťuje jednak transformaci úrovní signálů z TTL logiky do úrovní standardu RS-485, jednak připojování a odpojování zařízení od sběrnice. Dalším blokem je výkonový budič krokového motoru, který zajišťuje jeho silové buzení. Jeho vstupy jsou připojeny na řídicí mikroprocesor a jeho výstupy na buzený motor. Poslední část ve schématu je blok napájení. Protože krmičku musí být možno připojit i k nestabilizovanému zdroji a napájet ji kolísajícím napětím v rozmezí 12-32V, musí obsahovat jednotku, která zajistí napájení

patřičné kvality pro řídicí logiku (+5V) a pro motor (+12V). Blokové schéma popsané v tomto odstavci je nakresleno na obrázku 2.16.



Obr. 2.16 – Blokové schéma řídicí jednotky

2.5.3.2 Použité součástky

Jako hlavní řídicí procesor je použit jednočipový mikrokontrolér AT90S2313 firmy Atmel. Jedná se o 8-bitový RISCový mikroprocesor, běžící na taktovací frekvenci 10MHz, který je vybaven 2kB FLASH paměti pro uchování programu, 128B EEPROM, 128B SRAM, 32 multifunkčními 8-bitovými registry, dvěma nezávislými čítači/časovači (8 a 16 bitovým), PWM výstupem, integrovaným rozhraním pro sériovou komunikaci, integrovaným watchdogem a analogovým komparátorem. Velkou výhodou je možnost sériového programování (ISP) paměti FLASH. Díky tomu nepotřebujeme pro nahrávání programu do procesoru žádný programátor, a nadto je možno programovat jej přímo ve výsledném zapojení (řídicí jednotka je navržena tak, aby procesor AT90S2313 bylo možno programovat přímo v obvodu). Bližší informace viz. [19].

Silové buzení krokového motoru provádí výkonový budič L293D, výrobek firmy SGS-Thomson. Na všech výstupech má integrované ochranné diody proti potenciálně nebezpečným napěťovým pulsům[1]. Každý výstup je schopen dodávat proud až 0,6 A, což je pro tuto aplikaci silně předimenzované, a proto si můžeme dovolit provozovat ho bez chladiče. Detailní informace o tomto budiči lze získat z [20].

Připojení řídicího systému na RS-485 zajišťuje budič Linear technology LTC485, splňující standardní požadavky kladené na budiče sběrnice RS-485 (viz. 2.3.1). Podrobnosti lze najít v [21].

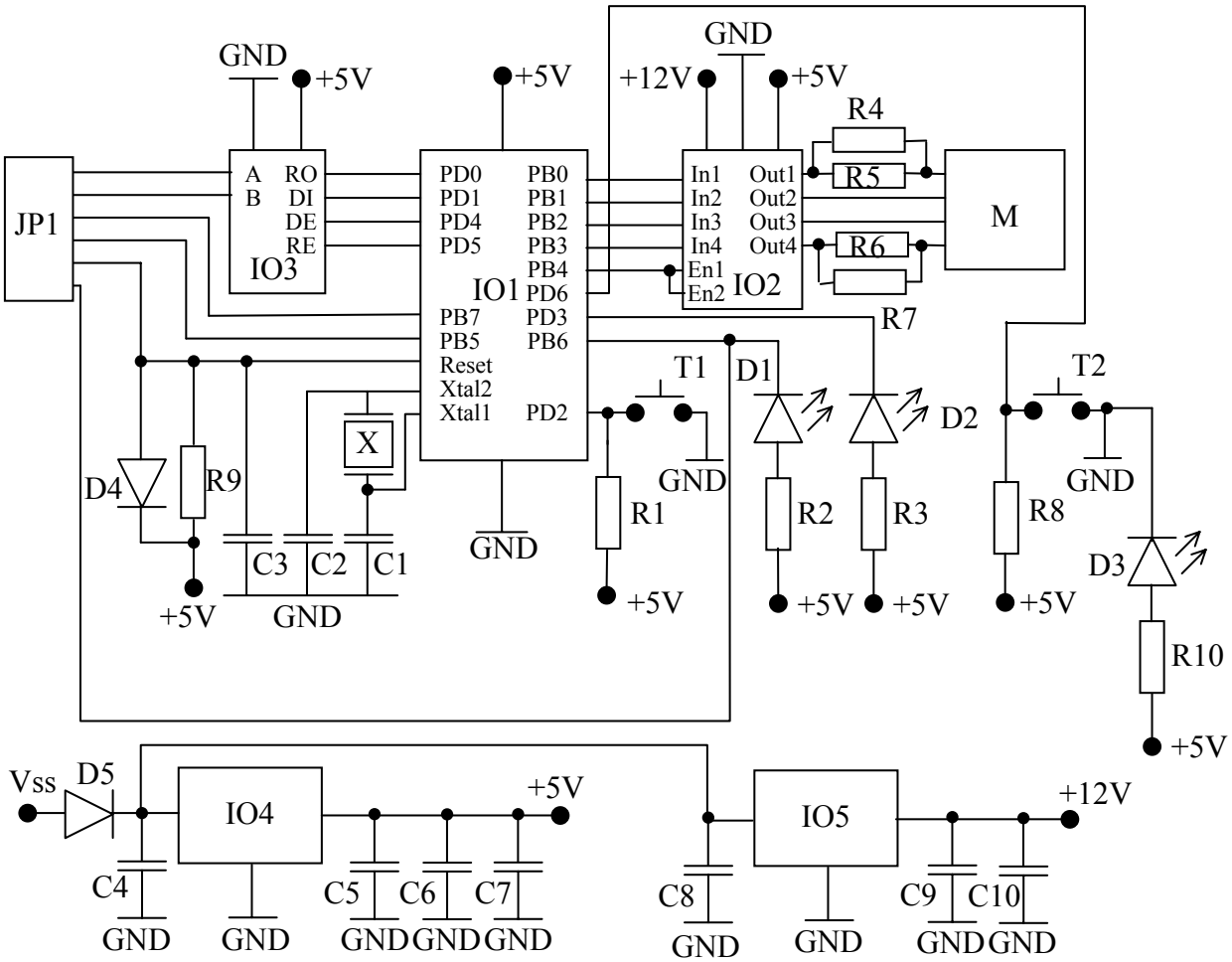
Jako pohonná jednotka pístu je použit krokový motor Microcon SL17-0301. Délka kroku je 1,8°, statický kroutivý moment 150mNm. Více informací lze získat na webových stránkách výrobce [22].

[1] Tyto napěťové pulsy vznikají při rozpínání induktivní zátěže, v našem případě vinutí krokového motoru. Bez patřičné ochrany by mohly být zničeny koncové tranzistory výkonového budiče. Tato ochrana se většinou řeší pomocí dvojice diod, připojených v závěrném směru od výstupu budiče proti napájecímu napětí a proti zemi.

2.5.3.3 Popis zapojení řídicí jednotky

V této kapitole detailně popíšeme zapojení řídicí jednotky, jehož schéma je na obrázku 2.17. Zapojení je maximálně jednoduché a účelné, ale přesto schopné zajistit všechny požadované činnosti a obsahující všechny nezbytné ochranné prvky tak, aby byla zajištěna jeho spolehlivá funkce.

Úpravu vstupního napájecího napětí zajišťují obvody IO4 a IO5. Jedná se o standardní napěťové stabilizátory 7805 a 7812. Na vstupu mají proti zemi připojeny svitkové kondenzátory C3 a C4 o kapacitě 330nF, sloužící k vyhlazení vstupního napětí[1]. Na jejich výstupech jsou ve schématu naznačeny blokovací kondenzátory C5 až C9. Ty jsou umístěny mezi napájecími vývody jednotlivých integrovaných obvodů a pravidelně rozmístěny mezi vodiči napájecího vedení. Funkce těchto kondenzátorů je detailněji popsána v části 2.5.3.5.



Obr. 2.17 – Schéma zapojení řídicí jednotky

Budič sběrnice LTC485 (IO3) je připojen jednak přímo na sběrnici RS-485, jednak na procesor AT90S2313 (IO1). Piny PD0 a PD1 na procesoru jsou přijímač/vysílač UART (sériového kanálu), piny PD4 a PD5 slouží k řízení připojení/odpojení vysílače, resp. přijímače obvodu IO3 od sběrnice. Z důvodů maximální jednoduchosti zapojení a nízké ceny nebylo použito impedanční oddělení budiče sběrnice. Při praktických zkouškách se toto řešení ukázalo být jako dostatečné a nesetkal jsem se s žádnými problémy z tohoto směru.

[1] Ve schématu označeno jako Vss

Výkonový budič motoru L293D má dvojí napájení, +5V pro řídicí logiku a +12V pro výkonovou část. Jeho čtyři vstupy jsou připojeny na piny PB0-PB3 řídicího procesoru. Jejich stav kopírují[1] výkonové výstupy Out1-Out4. Vstupy označené jako En1 a En2 slouží k nastavení výstupů Out1-Out4 do třetího stavu. Jsou ovládány společně výstupem procesoru PB4. Pokud motor stojí, nastaví procesor výstupy obvodu IO2 do třetího stavu, aby vinutími motoru zbytečně netekl proud. Rezistory R4-R7 na výstupech obvodu IO2 slouží k omezení proudu, tekoucího vinutími motoru.

LED diody D1 a D2 slouží k indikaci provozních stavů řídicí jednotky. Pin PB6, ke kterému je připojena dioda D1, je mimoto použit pro sériové programování (ISP) procesoru. LED dioda D3 slouží k indikaci zapnutí krmičky (svítí vždy, je-li připojeno napájecí napětí).

Dioda D5 je ochranná dioda na vstupu napájení. Zaručuje, že nedojde k poškození zařízení při opačné polaritě napájecího napětí.

Ovládací tlačítko T1 slouží k ručnímu řízení krmičky. Tlačítko T2 je mikrospínač, jehož stisk signalizuje dojezd pístu na zadní doraz. Obě tlačítka jsou vybavena zdvihacími rezistory[2] R1 a R8 o odporu 10kΩ.

Piny PB5, PB7 a RESET jsou použity k sériovému programování (ISP) mikroprocesoru, a proto jsou také vyvedeny na konektor JP1. Dioda D4, rezistor R9 a kondenzátor C3 jsou ochranné prvky pinu RESET, které zabraňují samovolnému resetování řídicího procesoru vlivem rušení. Jejich funkce je podrobněji vysvětlena v části 2.5.3.5.

2.5.3.4 Seznam součástek

Rezistory:
- R1, R8, R9 … 10kΩ, 0,3W
- R2, R3, R10 … 330Ω, 0,3W
- R4 – R7 … 33Ω, 0,6W

Kondenzátory:
- C1, C2 … 22pF, keramické
- C3 … 4,7nF, keramický
- C4, C8 … 330nF, svitkové
- C5-C7, C9, C10 … 220nF, keramické

Diody:
- D1, D3 … LED zelená, Ø3mm
- D2 … LED červená, Ø3mm
- D4 … usměrňovací dioda 1N4148
- D5 … usměrňovací dioda 1N4007

Konektory:
- JP1 … Canon 9F

Tlačítka:
- T1 … tlačítko dvoupólové
- T2 … mikrospínač dvoupólový

---

[1] Myšleno tak, že je-li například In0 v log.1 tak na výstupu Out1 je +12V a vice versa, je-li In0 v log.0 tak na Out1 je 0V.

[2] Mikroprocesor AT90S2313 používá vstupy/výstupy v zapojení s otevřeným kolektorem. Proto je nutné jejich napětí do log.1 „zdvihat“ pomocí rezistoru, připojeného na +5V. Procesor je sice vybaven integrovanými zdvihacími rezistory na každém vstupním/výstupním pinu, ale ty mají odpor kolem 100 kΩ, což je hodně. Proto jsou obě tlačítka pro zkrácení přechodového děje a potlačení vlivu rušení při přechodu z log.0 do log.1 (uvolnění stisku tlačítka) vybaveny externími zdvihacími rezistory o odporu o jeden řád menším.

Krystaly:

- X … krystal 10MHz

Motory:

- M … krokový motor Microcon SL17-0301

Integrované obvody:

- IO1 … mikroprocesor ATMEL AT90S2313
- IO2 … výkonový budič L293D
- IO3 … budič pro RS-485 LTC485
- IO4 … stabilizátor 7805
- IO5 … stabilizátor 7812

2.5.3.5 Ochranné prvky

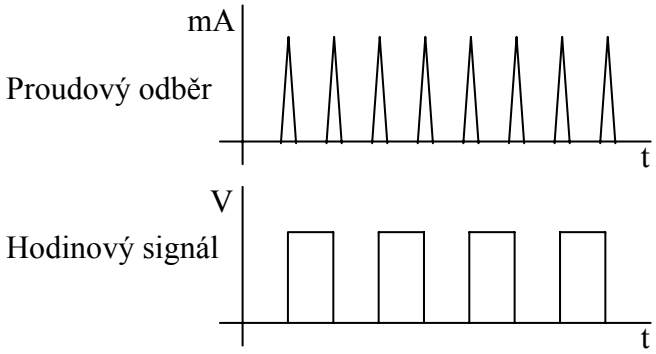
Protože řídicí jednotka automatické krmičky je průmyslové zařízení, musíme počítat s tím, že bude mnohdy vystavena všelikým rušivým vlivům. Proto je vybavena i prvky, které nemají přímý vliv na její funkci, ale zajišťují její ochranu proti těmto vlivům. V této části popíšeme podrobněji jejich funkci.

První ochranný prvek se nachází hned na vstupu napájení. Je jím dioda D5, která zaručuje, že nedojde k poškození zařízení, pokud bude omylem připojeno napájecí napětí opačné polarity.

Za ochranné prvky lze považovat i vyhlazovací kondenzátory C4 a C8. Tyto kondenzátory vyhlazují průběh napětí na vstupech stabilizátorů IO4 a IO5. Jejich funkce je důležitá zejména v případě, že je k napájení zařízení použit nestabilizovaný zdroj.

Důležitou funkci mají tzv. blokovací kondenzátory C5-C7, C9 a C10. Jsou důležité zvláště tehdy, když napájecí zdroj není dostatečně „tvrdý," tj. vykazuje nadměrný pokles napětí při zvětšení odběru. Vysvětleme si blíže jejich funkci.

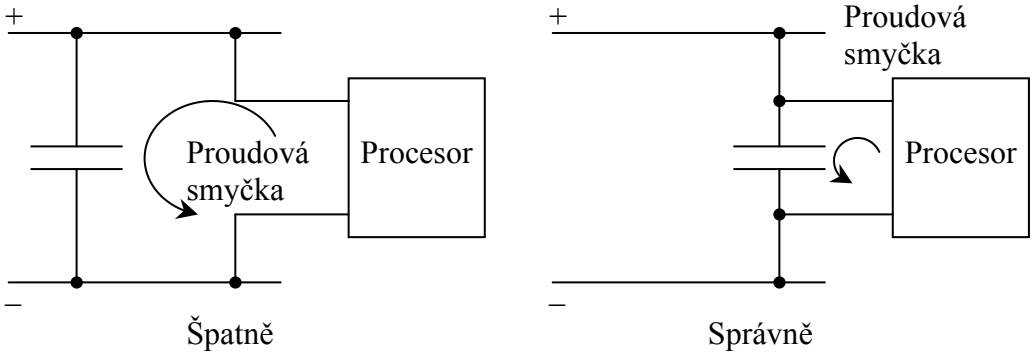
Nahlédneme-li letmo do datasheetu některého z použitých integrovaných obvodů, například procesoru AT90S2313, můžeme snadno dospět k mylnému názoru, že k jeho napájení postačí i relativně „měkký" zdroj napětí, vždyť jeho proudový odběr je pouhých 2,8 mA. Nenechme se ale mýlit tímto velmi malým údajem. Hodnota uváděna v datasheetu je hodnota průměrná. AT90S2313 a mikroprocesory obecně nemají v čase konstantní proudový odběr. Časový průběh jejich napájecího proudu spíše připomíná dirackovy impulsy. Proudový odběr je korelovaný s hodinovým signálem a nejvyšších hodnot dosahuje na jeho hranách. Tyto špičkové hodnoty mohou dosahovat řádově až několika set miliampér. Proudový odběr potom ovšem velice rychle spadne téměř k nule. Příklad časového průběhu napájecího proudu mikroprocesoru je na obrázku 2.18.



Obr. 2.18 – Časový průběh napájecího proudu mikroprocesoru

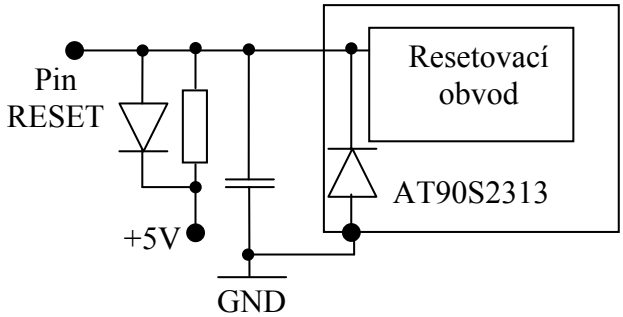
Pokud napájecí zdroj není dostatečně „tvrdý," může jeho napětí v čase špičkového odběru značně poklesnout. To může snadno způsobit „zamrznutí" mikroprocesoru. Tento neduh vyřeší blokovací kondenzátor, zapojený mezi napájecí vývody procesoru, jak je naznačeno na obrázku 2.19. Tento kondenzátor se nabije až na úroveň napájecího napětí v době, kdy je odběr procesoru zanedbatelný. Když pak tento prudce vzroste a napájecí napětí začne klesat, počne se kondenzátor vybíjet a zabrání tak jeho prudkému poklesu. Výsledkem je „vyhlazený" průběh napájecího napětí.

Důležité je, aby blokovací kondenzátor byl fyzicky co nejblíže napájecím vývodům a aby jeho přívody byly co možná nejkratší. Tím zmenšíme odpor přívodů a zpomalíme tak vybíjení kondenzátoru. Jeho účinek se tak zlepší. Na závěr je nutno podotknout, že blokovací kondenzátor nesmí mít příliš velkou kapacitu, aby se stačil nabít mezi jednotlivými proudovými špičkami. Jeho kapacita typicky bývá 100-220 nF.



Obr. 2.19 – Zapojení blokovacího kondenzátoru

Další ochranný prvek řídicí jednotky tvoří součástky D4, C3 a R9, které tvoří ochranný obvod zabraňující samovolnému resetu řídicího procesoru. Mikroprocesory řady Atmel AVR, do které patří i AT90S2313, jsou velmi náchylné k samovolnému resetování vlivem vnějšího rušení. U krmiček například často docházelo k samovolným resetům když bylo někde v jejich blízkosti spuštěno nějaké silové zařízení (například čerpadlo nebo vzduchovadlo). Tento problém byl vyřešen ochranným obvodem[1] naznačeným na obrázku 2.20. Z něj vidíme, že pin RESET má integrovanou pouze ochrannou diodu proti impulsům záporného napětí. Ochranné zapojení přidalo ještě ochrannou diodu proti kladným přepěťovým pulsům, zdvihací rezistor zapojený proti +5V a filtrační kondenzátor, zapojený proti zemi. Toto vše by samozřejmě nebylo potřeba, pokud bychom pin RESET připojili přímo na +5V. Tím bychom ovšem zároveň ztratili možnost sériového programování ISP, ke kterému je možnost ovládání pinu RESET nutná. Naznačené řešení nám tuto možnost zachová.



Obr. 2.20 – Ochrana pinu RESET

[1] Jedná se o výrobcem doporučené řešení, které lze najít v dokumentaci k procesorům řady AVR.

Poslední ochranný prvek, kterým je řídicí jednotka vybavena, není na schématu zapojení viditelný. Jedná se o interní watchdog procesoru AT90S2313. Tento ochranný čítač/časovač je zhruba každou vteřinu nutno softwarově vynulovat, jinak vyvolá reset procesoru. Tím je zaručeno, že pokud by snad řídicí procesor krmičky přestal pracovat, nepotrvá jeho výpadek déle než jednu vteřinu.

2.5.4 Firmware řídicí jednotky

2.5.4.1 Úvod

V této části popíšeme software[1] řídicí jednotky automatické krmičky ryb. Na úvod této kapitoly je třeba říct, že automatická krmička je zařízení kompatibilní s moduly ADAM firmy Advantech. Kompatibilní v tom smyslu, že používá formálně shodný komunikační protokol jako tyto přístroje (viz. 2.3.3.2). Automatická krmička využívá některé z povelů, používaných modulem ADAM-4080. Ačkoli jsou tyto povely formálně shodné, pro krmičku mají pochopitelně jiný význam.

Tento způsob komunikace byl zvolen především z důvodů snadné integrovatelnosti krmičky do řídicího systému DisCO. DisCO totiž obsahuje ovladače pro moduly řady ADAM. Díky použití formálně shodného komunikačního protokolu jsme se vyhnuli[2] nutnosti navrhovat nový ovladač specielně pro krmičku a bylo možné použít pro ni existující ovladač modulu ADAM-4080.

Dále je nutné předeslat, že krmičku lze ovládat v zásadě dvěma způsoby: Buď telegramem zaslaným po sběrnici RS-485, nebo manuálně, pomocí ovládacího tlačítka, umístěného na horním panelu krytu zařízení. Pro plné pochopení následujícího textu je vhodné přečíst si přílohu 6.3, kde je ovládání krmičky podrobně popsáno.

2.5.4.2 Obecný popis programu

Začněme nejprve obecným popisem hlavních funkcí firmwaru a jeho vývojovým diagramem. Nejobecněji můžeme jeho funkci shrnout například takto:

Po resetu (zapnutí) řídicí jednotky program nejprve provede inicializaci zařízení (kalibraci polohy pístu). To provede tak, že pístem zacouvá až na zadní doraz, což je indikováno stiskem mikrospínače. Na dorazu si vynuluje počítadlo polohy pístu a píst vrátí zase zpět do výchozí posice. Potom program přejde do klidového stavu, ve kterém poslouchá všechny telegramy, chodící po sběrnici a čeká, dojde-li k nějaké „události“. Tato „událost“ může být v zásadě dvojího druhu:

- Došlo ke stisku tlačítka manuálního ovládání.
- Přišel telegram, určený pro toto zařízení.

Dojde-li ke stisku ovládacího tlačítka, snaží se program nejprve zjistit, jakého typu tento stisk je (dlouhý stisk nebo dvouklik). Podle toho potom vykoná příslušnou akci (posun pístu vpřed nebo vzad po dobu stisku tlačítka či zacouvání pístem až do zadní zarážky).

Pokud přijde telegram určený pro toto zařízení (což se pozná podle adresy), tak jej dekóduje a zjistí, co se od něj žádá. Podle toho odpoví a případně ještě vykoná určenou akci (například posun pístu).

---

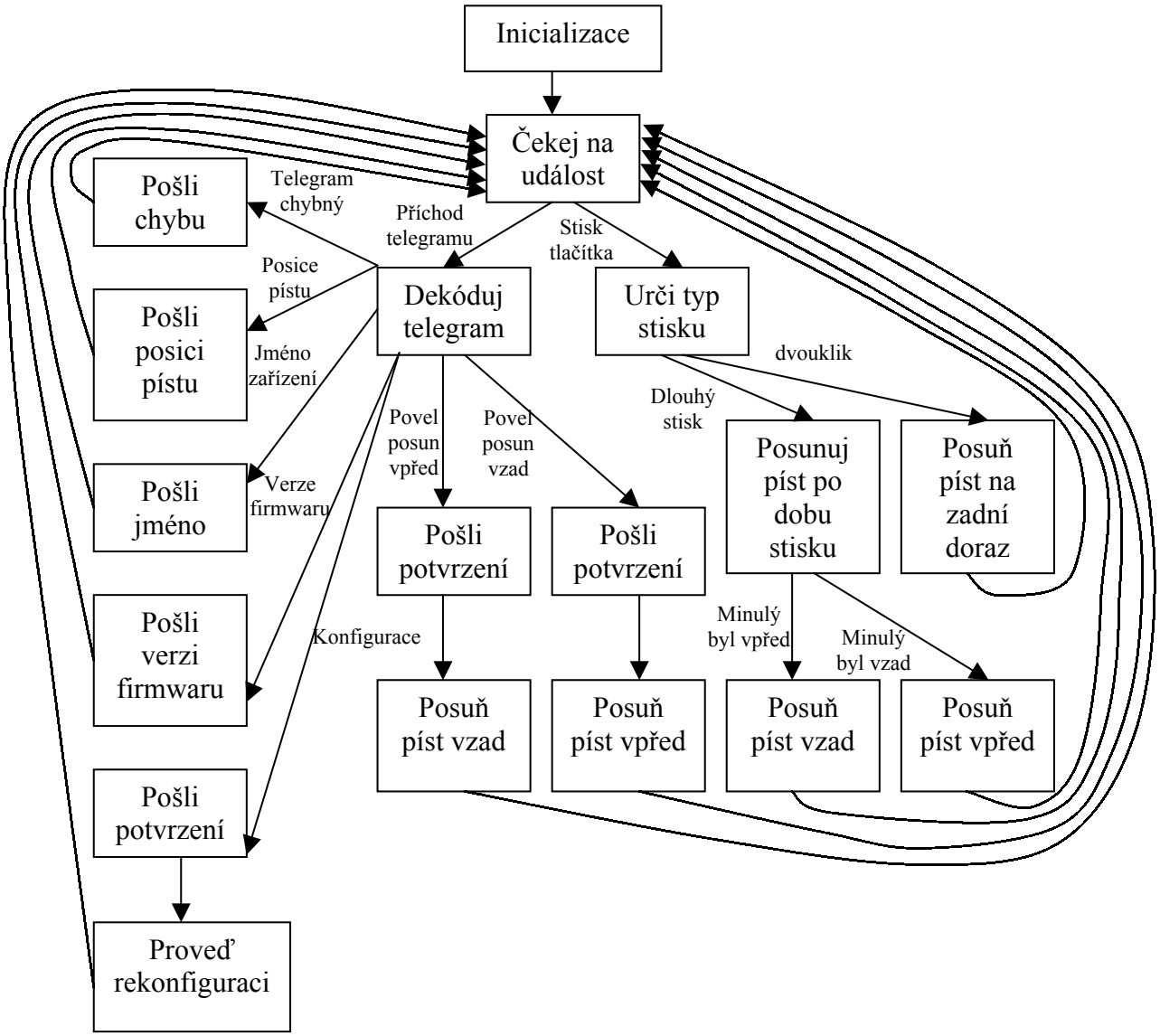
[1] Software vestavěných zařízení se často nazývá firmware.

[2] Množné číslo je zde zcela na místě. Integrace řízení krmičky do systému DisCO probíhala ve spolupráci s ing. Liborem Švehlou z firmy DataPartner, která systém DisCO vytvořila.

Akce nemusí být vždy nutná, například pokud telegram obsahuje dotaz na jméno zařízení, vyšle firmware pouze příslušnou odpověď a skončí. Pokud telegram obsahuje například povel pro posun pístu vpřed o x kroků, firmware nejprve odpoví, že povel přijal a poté akci skutečně vykoná. Toto pořadí je nutné z toho důvodu, že posun pístu je časově náročná akce, která trvá řádově vteřiny nebo i desítky vteřin, podle délky posunu. Zařízení ovšem musí odpovědět do vypršení krátkého časového limitu (viz. 2.3.3.2). Proto firmware nejprve odpoví masterovi a teprve potom uskuteční žádanou akci.

K přijmu telegramů je ještě dlužno dodat, že o tom, jestli je či není příchozí telegram určen tomuto zařízení firmware rozhoduje okamžitě po přijetí adresových znaků. Pokud nesouhlasí s adresou tohoto zařízení, zbytek telegramu se ignoruje. Z tohoto důvodu „událost“ vyvolá pouze telegram, adresovaný tomuto zařízení.

Vývojový diagram programu je na obrázku 2.21.



Obr. 2.21 – Vývojový diagram firmwaru

2.5.4.3 Architektura programu

Nyní již můžeme přistoupit k podrobnějšímu vysvětlení funkce řídicího programu. Program je napsán v assembleru procesoru AT90S2313 a jeho zdrojový kód s komentáři lze najít v příloze 6.2. Na tuto přílohu se budeme v následujícím textu často odvolávat. V této části zdrojový kód postupně projdeme a vysvětlíme alespoň rámcově funkci jeho jednotlivých bloků. Ještě předtím ale stručně popíšeme strukturu a filozofii celého programu.

Program maximálně využívá všech periferií a možností procesoru AT90S2313. V paměti EEPROM jsou uloženy konfigurační parametry zařízení (dva znaky jeho adresy, rychlost sériového přenosu a typ parity). Paměť SDRAM je využita k několika účelům. Jednak je v ní vytvořen zásobník procesoru, dále je v ní uložena aktuální posice pístu[1] a nakonec se jí využívá pro ukládání příchozích telegramů.

Jako ochranný prvek je využit integrovaný watchdog. Je nastaven tak, aby vyvolal reset, není-li softwarově nulován déle než jednu vteřinu. Jedno z externích přerušení (INT0) je využito k indikaci stisku ovládacího tlačítka. Integrované rozhraní pro sériový přenos (UART) je využito ke komunikaci po RS-485.

Program využívá obou integrovaných čítačů/časovačů. Osmibitový časovač je využit k časování řízení krokového motoru (generování pulsů dané délky) a k nulování watchdogu. Šestnáctibitový časovač se využívá k měření doby mezi dvěma stisky ovládacího tlačítka (detekce dvoukliku) a k řízení blikání LED diod.

K nahrávání programu do paměti FLASH je využito sériového programování ISP.

Program můžeme rámcově rozdělit do několika logických celků. Prvním z nich je inicializace funkcí samotného procesoru a hardwaru řídicí jednotky (výchozí nastavení jednotlivých registrů, výšky zásobníku, načtení konfiguračních parametrů z EEPROM, počáteční nastavení vstupů/výstupů, nastavení watchdogu a parametrů UART, definice a povolení některých vektorů přerušení, nastavení čítačů/časovačů atd.). Tato část je ve zdrojovém kódu vyznačena a končí oddělovačem ;***** Hlavní program (viz. příloha 6.2), který vyznačuje začátek vlastního programu.

Dalším logickým celkem je inicializace samotné krmičky, tj. zjištění aktuální posice pístu. To se děje následujícím způsobem: Vynuluje se počítadlo aktuální posice pístu a pístem se začne couvat. To se děje až do okamžiku, kdy stisk mikrospínače signalizuje dosažení zadní zarážky. V této chvíli se odečte aktuální stav počítadla a to se znovu vynuluje. Píst se potom opět přesune v před o stejný počet kroků, po který se couvalo. Tím se vrátí zpět do výchozí posice. To se děje proto, že pokud by například došlo k výpadku proudu někdy uprostřed krmícího programu a píst se po kalibraci nevrátil zpět do výchozí posice, krmení by přestalo správně fungovat. Takto ovšem může krmička po kalibraci pokračovat v přerušené činnosti tak, jako kdyby se nic nestalo.

Dalším celkem je „hlavní“ cyklus, ve kterém se program pohybuje po celou dobu běhu. Je označen návěstím „Main“ (viz. příloha 6.2). Tento cyklus je velice jednoduchý – program zde čeká na nějakou „událost,“ tj. detekci stisku tlačítka či přijetí telegramu určeného tomuto zařízení.

Jak detekce stisku tlačítka, tak přijetí telegramu se děje pomocí přerušení. Stisk tlačítka vyvolá přerušení INT0. Obslužná rutina tohoto přerušení „INT0_Han“ (viz. příloha

[1] Posice pístu se určuje v počtu kroků krokového motoru od zadní zarážky, určené mikrospínačem. Mezi počtem kroků (otoček) krokového motoru a polohou pístu v kontejneru je jednoznačný lineární vztah. Posice pístu je v paměti uložena jako dvoubajtové (šestnáctibitové) číslo bez znaménka.

6.2) nejprve zkontroluje, není li detekován „falešný stisk“[1]. Není-li tomu tak, nastaví T-bit (použitý jako flag stisku tlačítka) v registru SREG a skončí.

Přijímání telegramu se děje taktéž pomocí přerušení. Každý byte (znak) přenosu je přijímán automaticky integrovaným rozhraním UART. Po přijetí celého bytu (znaku) se vyvolá přerušení. Obsluha tohoto přerušení (rutina RX_Complete, viz. příloha 6.2) znak vyhodnotí a pokud je platný, uloží jej do paměti. Po obdržení znaku konce telegramu (cr) se nastaví flag přijetí nových dat. Přijímací rutina kontroluje hned na začátku přenosu adresové znaky, a všechny telegramy s adresou jinou než je adresa tohoto zařízení ignoruje.

Poslední logický celek je tvořen skupinou podprogramů, vykonávajících různé podúlohy. V následujících několika odstavcích stručně popíšeme jejich funkci. Zdrojový kód s komentáři všech níže popisovaných rutin lze najít v příloze 6.2.

Podprogram Vykonej_P se volá z hlavního cyklu v případě příchodu telegramu a je určen k jeho dekódování. Podle klíčových znaků povelu určí, o jaký povel se jedná a zavolá příslušnou obslužnou rutinu. Pokud povel nedokáže dekódovat, pošle masterovi kód chyby (blíže viz. 2.5.4.4).

Rutina Píst_Vzad slouží k obsluze povelu „posun pístu vzad“. Rutina nejprve dekóduje počet kroků, o které se má píst posunout, pošle masterovi potvrzení o vykonání povelu, rozsvítí červenou kontrolní LED a zavolá obslužnou rutinu realizující otáčení motoru směrem vzad. Jako parametr se jí předává žádaný počet kroků. Ošetření možného překročení rozsahu posunu (přejetí zadního dorazu) řeší přímo tato obslužná rutina.

Rutina Píst_Vpřed má podobnou funkci jako Píst_Vzad, ale slouží k posunu pístu opačným směrem. Vzhledem k tomu, že přední doraz pístu na rozdíl od zadního nemá žádnou zpětnou vazbu, ošetřuje tato rutina také případ možného překročení rozsahu pohybu. Je-li rozsah překročen, uskuteční se pohyb pístu pouze do maximální možné meze tak, aby píst nevypadl z kontejneru.

Rutina Píst_Pos slouží k obsluze požadavku na zaslání aktuální posice pístu. Číselný údaj nejprve převede do hexadecimálního tvaru a potom odešle příslušný telegram masterovi.

Rutiny Uprav_ZHex a Uprav_DoHex slouží k převodu čísla z/do hexadecimálního tvaru.

Rutiny Pošli_ID a Pošli_FR jsou velmi podobné a slouží k zaslání jména zařízení, respektive verze firmwaru masterovi (v odpovědi na jeho žádost). Jejich funkce je zřejmá.

Rutina Tl_Zm je obslužná rutina stisku ovládacího tlačítka. Volá se ihned po detekci prvního stisku. Rutina na začátku nejprve zjišťuje, jedná-li se o dlouhý stisk nebo dvouklik, a to tak, že čeká na druhý stisk tlačítka v předepsané době. Pokud k němu nedojde a tlačítko je stále stisknuto, přejde rutina do části, kde je ošetřen manuální posun pístu. Dojde-li k druhému stisku do vypršení časového limitu, přejde rutina do části, kde je vyřešeno zacouvání pístu až na zadní doraz. Pokud druhý stisk tlačítka není detekován do vypršení časového limitu a stisk tlačítka se mezitím uvolní, rutina se ukončí a program se vrátí zpět do hlavního cyklu.

Rutina Konfigurace se volá tehdy, je-li přijatý telegram povelem k rekonfiguraci zřízení. Rutina přepíše konfigurační údaje v EEPROM novými. K tomuto účelu využívá celkem tří pomocných rutin: Rutina Přepočti_BR složí k určení kalibrační konstanty

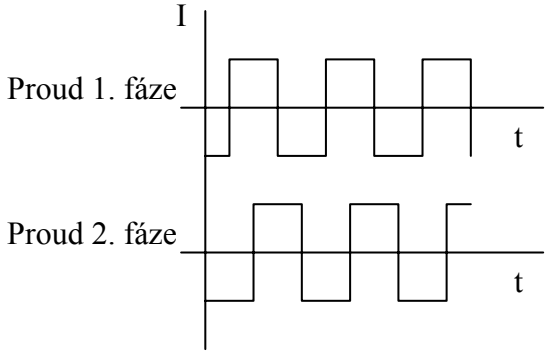
[1] Při stisku tlačítka dochází k přechodovému ději, kdy napětí na vstupním pinu procesoru klesá z +5V na 0V. Protože přerušení se volá na sestupnou hranu signálu, mohlo by dojít k jeho vícenásobnému vyvolání, než přechodový děj odezní. Proto se při vyvolání přerušení kontroluje, jestli už od posledního volání uplynula dostatečně dlouhá doba, takže je jisté, že přechodový děj už odezněl.

časovacího obvodu UART z konfiguračního parametru (viz. popis konfiguračního povelu v 2.5.4.4). Rutiny Čti_EP a Zapiš_EP slouží ke čtení, resp. zápisu bytu do paměti EEPROM.

Rutina Pošli_Byte je pomocná rutina, která vyšle znak po sériové lince RS-485 (UART). Rutina nejprve zjistí, je-li vysílán první znak telegramu. Pokud ano, počká několik mikrosekund (aby bylo jisté, že vysílač mastera už je odpojen od sběrnice a nedojde ke konfliktu). Potom vypočte paritu (je-li nastaven paritní přenos) vysílaného znaku, nastaví parametry UART a připojí vlastní vysílač ke sběrnici. Nakonec znak vyšle a čeká na vyprázdnění vysílacího registru, což je signalizováno přerušením. Obslužná rutina tohoto přerušení Tx_Complete nastaví příslušný flag. Rutina Pošli_Byte neodpojuje vysílač od sběrnice až do doby, kdy je vyslán poslední znak (cr). Potom vysílač nastaví do stavu vysoké impedance, připojí přijímač a obnoví původní konfiguraci UART.

Rutina Pošli_Busy slouží k vyslání chybové hlášky.

Rutiny Dopředu a Dozadu jsou velice podobné a souží k buzení krokového motoru tak, aby se otáčel tím či oním směrem. To se děje tím způsobem, že obě fáze krokového motoru jsou buzeny obdélníkovým signálem se střídou 1. Signály první a druhé fáze jsou fázově posunuty, viz. obr. 2.22. Pokud buzení první fáze předbíhá buzení fáze druhé, motor se otáčí doprava a píst se pohybuje vzad. V opačném případě se motor otáčí vlevo a píst se posunuje vpřed (bližší informace o krokových motorech a jejich buzení viz. [23]). Rutina Dozadu v každém cyklu kontroluje, nedošlo-li k dosažení zadního dorazu (stisk mikrospínače). Pokud se tak stane, rutina motor okamžitě zastaví. K časování budících signálů pro krokový motor se používá pomocná rutina Wait_M.



Obr. 2.22 – Buzení krokového motoru

2.5.4.4 Komunikační protokol

V této části popíšeme komunikační protokol krmičky. Jak již bylo řečeno, je formálně zcela shodný s protokolem přístrojů ADAM řady 4000, viz. 2.3.3.2. Proto jeho detailní popis vynecháme a vypíšeme pouze povely relevantní pro krmičku a jejich význam.

- %AANNTTCCFF(cr) – rekonfigurace zařízení. % znamená konfiguraci, AA jsou 2 byty adresy zařízení (v hexadecimálním tvaru), NN jsou 2 byty nové adresy, TT jsou neplatné byty a mohou být jakékoli – jsou tu pouze pro zachování kompatibility. CC je přenosová rychlost – 05 znamená 4800bps, 06 je 9600bps, 07 je 19200bps, 08 je 38400bps a konečně 09 je 115200bps. Kódy 05 až 08 mají shodný význam s ADAMem 4080, kód 09 je přidán navíc (ADAM 4080 nepodporuje vyšší rychlost než 38400bps). Naopak krmička nepodporuje rychlosti nižší než 4800bps. FF je typ parity – u ADAMu 4080 se tyto dva byty používají k nastavení jiných parametrů a v mohou nabývat pouze hodnot 00, 04, 64 a 68. Pro krmičku znamená hodnota 00 žádná parita, 04 a 64 znamenají lichá parita a 68 znamená sudá parita. Krmička

nepodporuje ADAMovský kontrolní součet. Po přijetí a úspěšném vyhodnocení povelu krmička odpoví !AA(cr), což znamená povel úspěšně přijat a dekódován, naposledy podle staré konfigurace. Potom se překonfiguruje a vizuálně tuto skutečnost potvrdí delším bliknutím obou LED. Poté se s ní dá komunikovat již jen podle nového nastavení, které je uloženo v EEPROM. **Příklad:** %0001000604(cr) znamená nastavení krmičky, která má aktuální adresu 00, na novou adresu 01, 9600bps, lichá parita. Krmička odpoví !00(cr).

- $AAF(cr) – vrátí verzi firmwaru. AA jsou 2 byty adresy zařízení (v hexadecimálním tvaru). Krmička odpoví !AA(verze)(cr), kde (verze) je znak, jež určuje číslo verze firmwaru. **Příklad:** $00F(cr) znamená dotaz na verzi firmwaru krmičky na adrese 00. Krmička odpoví například !004(cr), což znamená, že má firmware verze 4.
- $AAM(cr) – dotaz na jméno zařízení. Krmička odpoví !AA(jméno)(cr), kde (jméno) je řetězec znaků, určující ID zařízení. **Příklad:** $00M(cr) znamená dotaz na jméno krmičky na adrese 00. Krmička odpoví například !00krmitko(cr), což znamená, že má ID krmitko.
- @AARP(cr) – vrátí aktuální posici pístu. Krmička odpoví !AA(data)(cr), kde (data) jsou znaky čísla (v hexadecimálním tvaru) určujícího posici pístu. Možný rozsah je 0 – 12031. 0 znamená píst zcela vzadu (krmička plná), 12031 píst zcela vpředu (krmička prázdná). **Příklad:** @00RP(cr) je dotaz na posici pístu krmičky na adrese 00. Krmička odpoví například !00000015AC(cr), což znamená, že poloha pístu je 000015AC, tedy v desítkové soustavě 5548.
- @AAPA(data)(cr) – posun pístu vpřed. (data) je hexadecimální číslo určující relativní posun, tj. znamenají, o kolik chceme pístem pohnout, a ne absolutní hodnotu nové pozice! Číslo (data) musí mít povinně 8 znaků. Krmička odpoví !AA(cr) (povel OK). Pokud by se píst vysunul mimo hranice, pohyb se přesto provede, ale pouze do povolené hranice rozsahu, nikoli dál – pokud tedy například pošleme jako parametr číslo větší než 12031, krmička odpoví !AA(cr), nicméně provede pohyb pouze do přední hranice, tj. na okraj nádoby. **Příklad:** @00PA000015AC(cr) posune píst o 5548 kroků vpřed. Krmička odpoví !00(cr).
- @AASA(data)(cr) – posun pístu vzad. (data) je hexadecimální číslo určující relativní posun, tj. znamenají, o kolik chceme pístem pohnout, a ne absolutní hodnotu nové pozice! Číslo (data) musí mít povinně 8 znaků. Krmička odpoví !AA(cr) (povel OK). Pokud by se píst vysunul mimo hranice, pohyb se přesto provede, ale pouze do povolené hranice rozsahu, nikoli dál – pokud tedy například pošleme jako parametr číslo větší než 12031, krmička odpoví !AA(cr), nicméně provede pohyb pouze do zadní hranice, tj. na zadní okraj nádoby. **Příklad:** @00SA000015AC(cr) posune píst o 5548 kroků vzad. Krmička odpoví !00(cr).

Existuje ještě jedna další možná odpověď, kterou nám může krmička vrátit jako odpověď na jakýkoli povel. Je to ?AA(cr), což znamená stav Busy. Pokud se právě vykonává nějaký povel – například posun pístu, a motor je v pohybu – krmička odpoví na jakýkoli povel ?AA(cr), a řídicí systém musí povel zopakovat až za chvíli. Je-li krmička ve stavu Busy, nebere povely na vědomí. Odpověď ?AA(cr) může také znamenat chybu telegramu – byl obdržen neplatný telegram.

2.5.5 Testování automatického krmícího systému

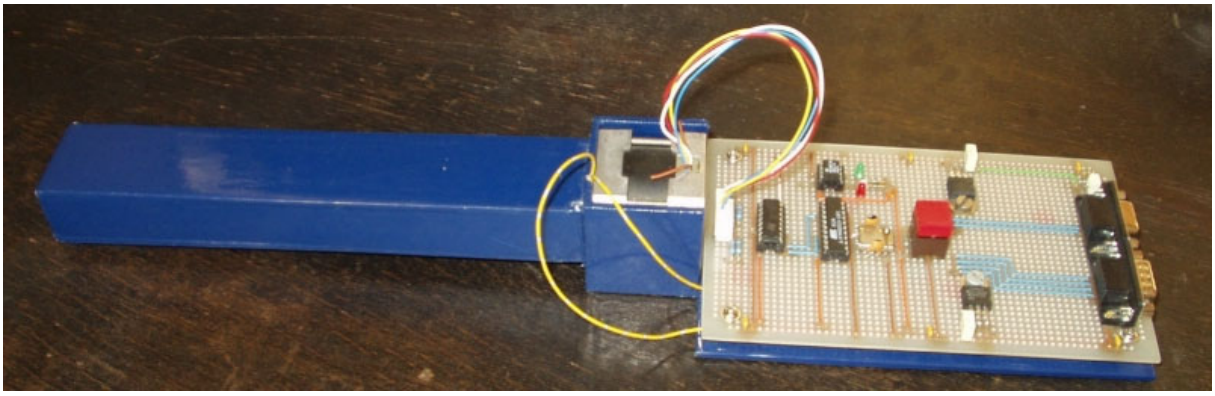
Poté, co jsme v předchozích kapitolách zevrubně popsali konstrukci a funkci automatické krmičky ryb, přistoupíme nyní ke krátkému pojednání o praktických zkouškách a experimentech s ní provedených a o dosažených výsledcích.

Testování automatické krmičky můžeme přehledně rozdělit do tří fází. Ve fázi první byl zkoušen první vyrobený prototyp krmičky, sloužící pouze k ověření základních principů funkce zařízení a jako platforma k naprogramování a otestování nejzákladnějšího softwarového vybavení. Tato fáze probíhala zhruba v období od konce prosince 2002 až do března 2003.

Ve druhé fázi, započaté koncem dubna 2003, probíhaly první praktické testy druhého vylepšeného prototypu zařízení ve Výzkumném ústavu rybářském ve Vodňanech. Tato fáze vývoje, ukončená v červenci 2003, byla zaměřena především na dlouhodobé testování spolehlivosti zařízení v průmyslových podmínkách, odstraňování chyb a vylepšování a přizpůsobování zařízení podle požadavků uživatelů (odborníků na chov ryb).

V polovině srpna 2003 byla do provozu uvedena ověřovací série 3 nových krmiček, zkonstruovaných podle poznatků a ponaučení z druhé fáze zkoušek. Tyto krmičky byly po otestování a „zaběhnutí“ použity ve srovnávacím experimentu automatizovaného a klasického chovu (viz. příloha 6.3). Tento experiment probíhal od začátku září do konce října 2003. Tuto část lze považovat za třetí logickou fázi vývoje.

První prototyp automatické krmičky (obr. 2.23), dokončený koncem roku 2002, byl použit výhradně k otestování způsobilosti zařízení a k vývoji nezbytného softwarového vybavení. Nikdy nebyl zamýšlen k nasazení v průmyslových podmínkách, čemuž odpovídá i jeho maximálně jednoduchá konstrukce. Jen díky němu bylo ovšem možno naprogramovat a odzkoušet první verze firmwaru i nezbytného dalšího softwarového vybavení. Protože krmičku samu o sobě by nebylo možné testovat, naprogramoval jsem k ní různé pomocné programy – program pro nahrávání firmwaru do řídicího procesoru, diagnostický program a první verzi řídicího programu, pomocí kterého bylo možno zkoušet automatické krmení. Řízení krmičky v té době ještě pochopitelně nebylo začleněno do systému DisCO, proto jsem pro účely jejího testování naprogramoval vlastní „provizorní“ řídicí program. Protože jsem nepředpokládal delší životnost těchto utilit, naprogramoval jsem je z důvodů co nejrychlejšího vývoje v Borland C++ pod MS-DOS.



Obr. 2.23 – První prototyp automatické krmičky

V období března a dubna 2003 jsem postavil druhý prototyp (obr. 2.24), schopný nasazení v reálném provozu. Zkušenosti s jeho dlouhodobým provozem v náročných

podmínkách chovné stanice byly neocenitelné a přinesly mnohá vylepšení a odstranění slabých míst konstrukce.

Obr. 2.24 – Druhý prototyp při testech v chovné stanici

Tyto poznatky byly využity v průběhu července-srpna 2003, kdy jsem zařízení překonstruoval a postavil ověřovací sérii tří kusů krmiček nového typu (větší kapacita, schopnost autokalibrace, lepší systém doplňování krmiva, přidání různých hardwarových i softwarových ochranných prvků pro zlepšení spolehlivosti). Ty potom byly nasazeny ve srovnávacích testech automatického a ručního řízení technologie (obr. 2.25).

Podrobnou zprávu o průběhu a výsledcích tohoto experimentu lze najít v příloze 6.3. V zásadě probíhal následovně:

Experiment byl rozdělen do tří fází. Šlo o zjištění vlivu automatického versus ručního řízení okysličování vody a automatického versus ručního krmení. Do šesti chovných nádrží byly nasazeny nové ryby o srovnatelné hmotnosti. Po dobu dvou měsíců byl stav ryb ve všech nádržích, především pak přírůstek jejich hmotnosti, průběžně vyhodnocován. V první fázi byl provoz tří nádrží řízen plně automaticky a druhých tří ručně. Ve druhé fázi bylo u všech nádrží instalováno automaticky řízené okysličování a u tří z nich bylo použito automatické krmení. V poslední fázi byly všechny nádrže krmeny ručně a rozdíl byl v řízení kyslíkování (3 automaticky a 3 ručně).

Po skončení experimentu byly výsledky statisticky zpracovány. Zatímco u kyslíkování se ukázaly značné přednosti automatického řízení (přírůstek biomasy při automatickém řízení byl o 70,2% větší oproti řízení ručnímu), výsledky ručního a automatického krmení byly prakticky stejné (automatické krmení vyšlo nepatrně lépe, nejednalo se ovšem o statisticky

významný rozdíl – přírůstek biomasy byl o 2,7% větší při použití automatických krmiček). Vliv automatického krmení na růst ryb proto nebyl prokázán. Ve srovnání plně automatického provozu (kyslíkování + krmení automatické) a plně manuálního provozu byl rozdíl v přírůstku biomasy o 73,7% větší při použití plně automatického provozu.

Obr. 2.25 – Nasazení krmiček ve skutečném provozu

Výsledné zhodnocení automatických krmiček po zkušenostech s jejich dlouhodobým provozem, sepsané ing. Janem Kouřilem, PhD., ředitelem Výzkumného ústavu rybářského, je uvedeno v příloze 6.5.

2.5.6 Integrace ovládání automatických krmiček do řídicího systému DisCO

Začlenění ovládání krmičky do řídicího systému DisCO bylo řešeno ve spolupráci s ing. Liborem Švehlou z firmy DataPartner. Z mé strany se jednalo pouze o implementaci komunikačního protokolu dle požadavků páně Švehlových tak, aby bylo možné použít existující ovladač pro modul ADAM-4080. Díky tomu stačilo pouze upravit datový model aplikace a přidat to něj povely pro krmičku.

Použití automatické krmičky řízené systémem DisCO je prakticky demonstrováno na školním modelu chovné stanice, popsaném v části 2.2.2.

2.5.7 Možná zlepšení systému

Jako všechna inženýrská zařízení, i automatická krmička je věc procházející kontinuálním vývojem. Jakmile se podaří implementovat jedno zlepšení, člověka už napadá

další. Proto i po uzavření tohoto projektu zbývá několik námětů pro další možná budoucí vylepšení celého systému.

Jednou z možností, jak celé zařízení zlevnit, by bylo použití jiného pohonného systému. Krokové motory mají tu nevýhodu, že patří mezi zařízení relativně drahá (krokový motor je zdaleka nejdražší součást automatické krmičky a tvoří asi 50% její celkové ceny). Jako perspektivní se jeví použití obyčejného stejnosměrného motoru, doplněného o inkrementální čidlo otáček (například motor pohonu stěračů z osobního automobilu doplněný o hallovu sondu otáček). Úprava celého zařízení by pravděpodobně nebyla příliš obtížná.

Dalším směrem možného vývoje je použití mechanického systému na bázi archimédova šroubu (viz. část 2.5.2.1). Řídicí jednotka by mohla zůstat prakticky beze změn. Výhody tohoto řešení jsou popsány v části 2.5.2.1. Pokud by se zařízení někdy vyrábělo sériově, vyřešilo by to pravděpodobně zde uváděné problémy se složitostí výroby.

2.6 Shrnutí

Ačkoli se nepodařilo prokázat vliv automatického krmení na rychlost růstu ryb, jedná se přesto o velmi perspektivní rybochovnou technologii. Nesporná je úspora lidské práce a snížení množství spotřebovaného krmiva, stejně jako výhodnost integrace krmení do komplexního řídicího systému technologie.

Zde představovaný model automatické krmičky se v dlouhodobých praktických testech dobře osvědčil a lze reálně uvažovat o jeho hromadné výrobě (samozřejmě za použití pokrokovějších materiálů a technologií, než jaké byly použity ke stavbě modelů). Dá se očekávat, že pokud se v zemědělské praxi podaří prosadit model automatické rybochovné stanice, budou automatické krmičky jeho nedílnou součástí. Zde řešený úkol lze tedy považovat za úspěšně splněný.

3. Automatické řízení vrtulníku

3.1 Úvod

V této části bude popsán druhý řešený problém, kterým bylo provést rešerši na téma automatické řízení vrtulníku a koncepčně navrhnout řídicí jednotku servomotorů pro vrtulník. V následujícím textu budou proto uvedeny převážně poznatky převzaté z rozličných literárních zdrojů, pouze části 3.3 a 3.4.7 obsahují některé původní myšlenky.

Pro lepší porozumění odkazům a poznámkám v dalším textu je nutné podotknout, že do projektu automatického řízení vrtulníku jsou zapojeni lidé z naší fakulty a lidé z Vojenské akademie (VABO) v Brně, kteří se zabývají vývojem bezpilotního letounu. Dále se nám podařilo získat velmi cenného externího konzultanta, kterým je pan Marco Sanvido, v současné době působící na University of California v Berkeley. Pan Sanvido je autorem práce [1] a významnou měrou se podílel na úspěšném vrtulníkovém projektu, který svého času probíhal na Eidgenössische Technische Hochschule (ETH) ve Švýcarském Curychu. Jeho zkušenosti a rady jsou pro nás neocenitelné a výsledky různých konzultací s ním vedených jsou zmiňované na mnoha místech této práce.

První kapitola této části obsahuje stručný popis řízeného systému. Dále následuje popis navrhovaného řídicího systému pro vrtulník a požadavky na něj kladené. V další kapitole je rámcově popsán hardware řídicího systému. Dále následuje stručný popis matematických modelů vrtulníku a nástin možných způsobů jeho řízení.

3.2 Popis řízeného systému

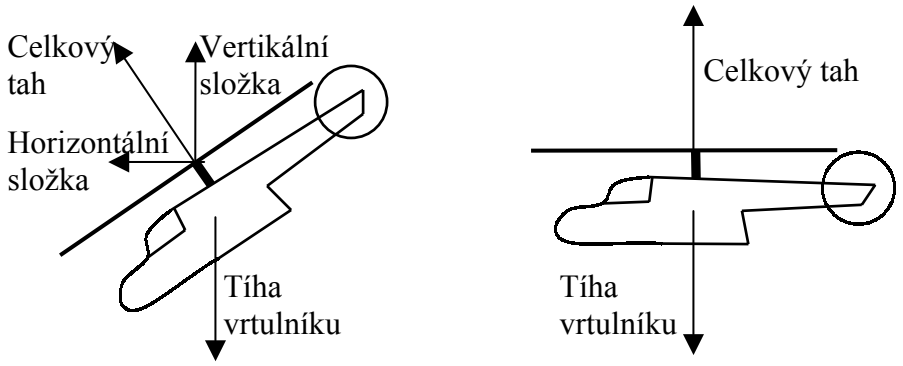
Hlavní obtíž automatického řízení vrtulníku spočívá v tom, že vrtulník je bytostně nestabilní a nelineární systém s velmi malým přirozeným tlumením a velmi rychlou dynamikou. Další obtíží je fakt, že vrtulník má celkem 6 stupňů volnosti (dof, z anglického degrees of freedom) – jsou to 3 složky prostorových souřadnic jeho referenčního bodu (zpravidla těžiště) a 3 úhly natočení kolem tří hlavních os – podélné osy procházející těžištěm, příčné osy procházející těžištěm a osy rotoru, která také zpravidla prochází těžištěm, kde se všechny 3 osy protínají. Pro plné pochopení celého rozsahu a obtíží této problematiky je velmi dobré znát alespoň základy aerodynamiky a mechaniky letu vrtulníku a principů jeho řízení. Tyto poznatky jsou podrobněji popsány příloze 6.1. Zde se omezíme pouze na velmi stručný popis jeho vlastností.

Vrtulník má ze systémového pohledu celkem 5 vstupů. Jsou to otáčky rotoru, velikost tahu rotoru ovládaná **kolektivním řízením** rotorových listů, dále rychlost klopení a klonění (otočení kolem příčné, resp. podélné osy) daná předozadním, resp. bočním **cyklickým řízením** a rychlost zatáčení (otočení kolem osy rotoru), která je dána kolektivním řízením listů pomocného rotoru (zadní vrtulky). Z důvodů popsaných v části 6.1.5.4 je výhodné udržovat otáčky rotoru konstantní, k čemuž se zpravidla používá samostatný PID regulátor. Tomu se pouze na začátku letu nastaví konstantní žádaná hodnota, která se po dobu letu nemění. Tím se počet vstupů z hlediska hlavního řídicího systému fakticky redukuje na 4. Dále je ještě nutno podotknout, že ke stabilizaci ocasu vrtulníku se používá **gyro**, taktéž nezávislé na hlavním řídicím systému, kterému hlavní řídicí systém jako žádanou hodnotu zadává rychlost zatáčení vrtulníku. Tím je hlavní řídicí systém zbaven starosti o stabilitu ocasu vrtulníku. Funkce gyra je podrobněji popsána v části 6.1.5.5.

Základní princip řízení vrtulníku vyplývá z možností daných jeho vstupy. Je zřejmé, že orientaci vrtulníku v prostoru (natočení kolem jeho tří hlavních os) můžeme ovlivňovat

přímo nastavením žádaných hodnot daných vstupů (cyklického řízení rotorových listů a kolektivního řízení listů pomocného rotoru). Situace je poněkud složitější, chceme-li vrtulník dopravit v prostoru z bodu *A* do bodu *B*. Chceme-li stoupat či klesat, je situace relativně jednoduchá; je-li vrtulník orientován vodorovně se zemí (rovina otáčení rotoru je rovnoběžná s rovinou země), potom můžeme rychlost a směr vertikálního pohybu vrtulníku řídit přímo pomocí kolektivního řízení rotorových listů. Situace se ovšem poněkud komplikuje, chceme-li např. letět dopředu či do strany. V tomto případě nám nezbývá než vrtulník pomocí cyklického řízení naklonit a tím změnit směr tahu rotoru. Tím ovšem dojde k destabilizaci vrtulníku, jak si ukážeme na jednoduchém příkladu.

Předpokládejme například, že vrtulník se nachází ve visu a my chceme vrtulník uvést do dopředného letu. Nezbývá nám, než ho pomocí předozadní cykliky naklonit směrem vpřed kolem jeho příčné osy. Pokud byl ovšem vrtulník ve visu dokonale stabilizován, byla velikost složky tahové síly rotoru mířící kolmo vzhůru stejně velká jako jeho tíha (vrtulník není ve visu orientován rovnoběžně se zemí, jak by se na první pohled mohlo zdát; situace je ve skutečnosti poněkud delikátnější, viz. článek 6.1.3.1). Nakloněním vrtulníku ovšem dojde k tomu, že se tato složka poněkud zmenší, viz. obr. 3.1, a vrtulník proto začne klesat. Toto klesání musíme kompenzovat zvýšením tahu rotoru, čímž ovšem jednak změníme nejen jeho vertikální složku, která vrtulník „drží ve vzduchu," ale také jeho složku horizontální, která ho urychluje ve směru letu. Změnou tahu rotoru ovšem také dojde ke změně reakčního silového momentu, jímž rotor působí na trup. Tuto změnu sice vykompenzuje gyro, ovšem přesto dojde k otočení vrtulníku kolem osy rotoru, viz. článek 6.1.5.4, které musí vyrovnat hlavní řídicí systém pomocí kolektivního řízení listů pomocného rotoru. Aby toho všeho nebylo málo, vrtulník se také dost pravděpodobně začne naklánět kolem podélné osy vlivem rozdílných rychlostí proudění okolního vzduchu kolem rotorových listů na návětrné a závětrné straně, jak je vysvětleno v článku 6.1.4.1. situace je obdobná při letu „do strany". Vidíme tedy, že i poměrně jednoduchý požadavek na změnu režimu letu vyvolá komplikovanou reakci stroje, kterou musí řídicí systém vhodně kompenzovat, což představuje velmi komplexní zásah do řízení.



Obr. 3.1 – Změna velikosti vertikální složky tahu rotoru vlivem klonění vrtulníku vpřed

Z obtíží popsaných výše plyne tedy značná nesnáz při návrhu spolehlivého stabilizujícího regulátoru pro vrtulník. Protože experimentální nastavování klasických regulátorů použitých na vrtulník je velmi nesnadné a nebezpečné, jelikož identifikační experimenty by vrtulník mnohdy uvedly do velmi složitých a náročných letových situací, je nezbytné použít nějaký vhodný matematický model, případně použít vhodně naučený fuzzy regulátor, jak je popsáno v části 3.6.

Exaktní matematické modely vrtulníku ovšem nejenže jsou silně nelineární, ale bývají velmi složité, mají-li respektovat například dynamiku rotoru a vertikální pohyb rotorových listů, viz. 6.1.3.2 a 6.1.4.1. Proto se jich pro účely automatického řízení zpravidla nepoužívá, a na místo toho se používá celá škála jednodušších a linearizovaných modelů, jejichž rozsah platnosti je ovšem omezen pouze na specifické režimy letu, blíže viz. část 3.5. Proto se pro různé letové režimy mnohdy používá různých regulátorů, mezi kterými se přepíná.

Z výše popsaného plyne, že použité stabilizující regulátory musí být značně robustní, protože matematické modely použité pro jejich návrh zdaleka nepopisují všechny aspekty chování vrtulníku. Dalším problémem je beznárazové přepínání mezi jednotlivými regulátory při přechodech mezi různými fázemi letu.

3.3 Popis řídicího systému

3.3.1 Úvod

V této podkapitole je popsán návrh hardwaru řídicího systému modelu vrtulníku, tak jak byl vytvořen ve spolupráci s dalšími lidmi zapojenými do projektu automatického řízení vrtulníku.

Řídicí systém na palubě vrtulníku musí plnit mnoho rozličných úkolů:

- stabilizovat vrtulník, což je úkol hlavní a mající v průběhu letu hlavní prioritu
- řídit vrtulník po předem zadané trajektorii letu
- komunikovat s nadřízeným řídicím systémem na zemi a plnit jeho povely
- sbírat a nahrávat telemetrické údaje (černá skříňka)
- obsluhovat užitečnou zátěž, viz. dále

Z výše uvedeného seznamu plyne, že se jedná o komplikovaný strukturovaný systém, který bude složen z mnoha víceméně nezávislých jednotek, zajišťujících různé dílčí funkce. Rozeberme nyní význam jednotlivých požadavků, vyjmenovaných výše, pro hardwarovou strukturu systému.

V prvním bodě požadujeme, aby námi navržený řídicí systém vrtulník stabilizoval. To samo o sobě je dosti náročný úkol. Evidentně budeme potřebovat relativně výkonný počítač, na kterém poběží námi navržený řídicí algoritmus. Nabízí se několik možností; Buď můžeme použít průmyslové PC, nebo si takový počítač navrhnout sami. Obě řešení mají svoje výhody a nevýhody.

Použití průmyslového PC znamená značnou úsporu času i finančních prostředků, které by bylo nutné vynaložit na vývoj vlastního počítače. Na průmyslovém PC lze také použít celou škálu nám známých operačních systémů a prostředků pro vývoj softwaru, nám dobře známých a snadno dostupných. Naproti tomu má průmyslové PC i své stinné stránky. Těmi jsou především relativně vysoká spotřeba energie, která na palubě modelu vrtulníku hraje významnou roli, jelikož máme k disposici baterie se značně omezenou kapacitou (přibližně 1500 mAh), které musí napájet všechny elektrické systémy na palubě. Použitím průmyslového PC tedy zkracujeme možnou dobu letu vrtulníku. Průmyslové PC má také zpravidla dosti omezenou škálu periferií, bude tedy nezbytné požít do něj rozšiřující karty, které zajistí některé další námi požadované funkce (např. CANBUS, viz. dále). Naopak má celou řadu periferií, které nevyužijeme (sériové a paralelní porty, rozhraní pro disketovou mechaniku atd.). To nejen zvyšuje příkon energie, ale taktéž hmotnost a rozměry, což jsou další dva velmi významné limitující faktory.

„Na míru ušitý“ řídicí počítač odstraňuje všechny výše uvedené nevýhody. Protože je od počátku navrhován se zřetelem na jeho budoucí použití, má právě a pouze ty periferie, o

kterých víme, že je budeme potřebovat. Také lze použít speciální procesor, nejvhodnější pro náš typ aplikace. Tím lze dosáhnout značné úspory hmotnosti, rozměrů a spotřeby elektrické energie. Je ovšem zřejmé, že vývoj takového počítače je relativně nákladný a časově náročný.

Nadmíru poučné je srovnání obou variant, stručně popsané v [1]. Švýcarský tým z technické university v Curychu, řešíce stejný problém, vyzkoušel postupem času obě varianty a proto s nimi má rozsáhlé praktické zkušenosti. Protože informace obsažené v [1] jsou až příliš stručné, obrátil jsem se dotazem přímo na pana Marca Sanvida, autora [1]. Díky své práci na vrtulníkovém projektu ETH má pan Sanvido velmi rozsáhlé znalosti úskalí obou řešení. Potvrdil, že použití průmyslového PC šetří velmi mnoho času i peněz. Dle jeho informací byl vývoj vlastního řídicího počítače velice zdlouhavý a obtížný, doprovázený i několika haváriemi vrtulníku, způsobenými selháním jimi navrženého hardwaru v obtížných podmínkách na palubě (jednalo se především o problémy s vibracemi, způsobenými spalovacím motorem). Na ETH ve Švýcarsku zašli dokonce tak daleko, že pro svůj účelově zaměřený počítač vyvinuli i vlastní real-time operační systém (HeliOS) a vlastní programovací jazyk (OBERON SA). Je ovšem otázkou, zdali výhody tím dosažené skutečně ospravedlňovaly vynaložené úsilí. Po důkladném zvážení obou možností se náš tým rozhodl použít průmyslové PC, osvědčené PC104 (blíže viz. 3.4.1).

Samotný řídicí počítač, byť jakkoli dokonalý, nám ovšem nezajistí řízení vrtulníku. Regulační algoritmus potřebuje jednak zajistit přísun potřebných vstupních veličin, jednak potřebuje akční členy, kterými by mohl vrtulník řídit. Těmito vstupy je samozřejmě informace o poloze, orientaci, rychlostech a zrychleních vrtulníku a akčními členy jsou servomotory, ovládající nastavení listů hlavního a pomocného rotoru. Proto budeme potřebovat jednotku, zajišťující přísun vstupních veličin, a jednotku zajišťující řízení servomotorů. Těmito jednotkami budou **jednotka inerciální navigace** (viz. 3.4.6) a **jednotka řízení servomotorů** (viz. 3.4.7).

V dalším bodě našeho seznamu požadujeme řízení vrtulníku po zadané trajektorii. Jednotka inerciální navigace nám sice poskytuje informaci o poloze vrtulníku, ovšem vzhledem k tomu, že ta se vypočítává z údajů akcelerometrů, měřících rozličná zrychlení (viz. článek 3.4.6), není tento údaj příliš přesný. To je způsobeno nepřesnostmi a driftem použitých senzorů, chybou jejich měření způsobenou vibracemi vrtulníku (způsobovanými především motorem, ale i aerodynamickými vlivy), atd. Proto je nutné zajistit přesnější informaci o poloze vrtulníku v prostoru, která nám také bude sloužit jako korekce pro údaje z jednotky inerciální navigace. Touto jednotkou buje jednotka **GPS**, viz. 3.4.4. Ačkoli GPS je sice vhodná metoda pro dlouhodobější navigaci vrtulníku (tzv. mission management), není vhodná např. pro řízení startu a přistání, protože například údaj o výšce vrtulníku nad terénem získaný z GPS je velice nepřesný. Proto použijeme ještě **elektronický výškoměr** (viz. 3.4.5) a **kompas** (viz. 3.4.3). S pomocí těchto jednotek již budeme schopni zajistit jak start, tak přistání vrtulníku a jeho řízení po žádané trajektorii.

V dalším bodu je požadována komunikace s nadřízeným systémem (tím je myšlena pozemní řídicí stanice, viz. 3.4.8). K tomu budeme ovšem potřebovat nějakou formu bezdrátového spojení se zemí. Tuto funkci bude zajišťovat **jednotka datového spojení** se zemí, viz. 3.4.2. Mimoto ještě zachováme původní **modelářský přijímač**, pomocí kterého budeme schopni ovládat vrtulník manuálně, pomocí standardního modelářského vybavení. To bude účelné nejen jako záloha pro případ selhání automatického řízení, ale taktéž pro identifikační experimenty prováděné s vrtulníkem.

Sběr telemetrických údajů může zajišťovat jak samostatná jednotka, tak sám řídicí počítač. Z důvodů úspor a maximální jednoduchosti systému bylo rozhodnuto, že tuto funkci bude v našem systému zajišťovat hlavní řídicí počítač.

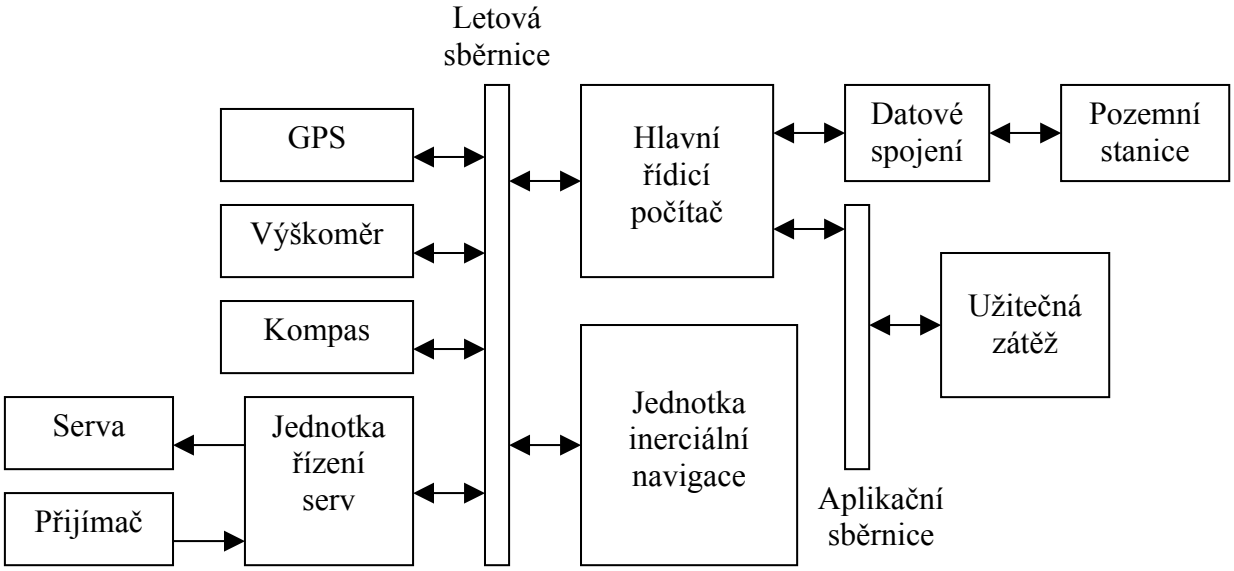
Obsluhu **užitečné zátěže**, za níž jsou považovány např. různé vědecké přístroje, kamera aj., tj. periferie, potřebné k plnění letové mise, ovšem neovlivňující přímo řízení vrtulníku, bude taktéž zajišťovat hlavní řídicí počítač.

3.3.2 Bloková struktura řídicího systému

Nyní, když už víme, z jakých jednotek bude řídicí systém složen, můžeme blíže popsat jeho hardwarovou strukturu. Je zřejmé, že musíme nějakým vhodným způsobem zajistit spojení jednotlivých funkčních bloků s řídicím počítačem. Protože by nebylo účelné připojovat každou jednotku zvlášť nějakým k tomu zvlášť určeným datovým kanálem, rozhodli jsme se vzhledem ke složitosti systému použít sběrnicovou architekturu. Za vhodné propojovací médium byla vybrána osvědčená průmyslová sběrnice CANBUS.

Předkládaný řídicí systém má sběrnice dvě, zcela nezávislé a oddělené. Obě sběrnice jsou stejného typu, ale použité k různým účelům. První sběrnice se nazývá letová a jsou na ní připojeny všechny funkční bloky, mající přímý vliv na řízení letu. Takzvaná „užitečná zátěž“ (viz. výše) je připojena na samostatné sběrnici, nazvané sběrnice aplikační. Obsluha letové sběrnice má před obsluhou sběrnice aplikační absolutní prioritu. Díky této architektuře je možné navrhnout a odladit všechny části nezbytné pro řízení letu zcela odděleně od užitečné zátěže. Protože ta je připojena na zcela oddělené sběrnici, nemusíme se bát, že by velký datový tok od užitečné zátěže (např. od kamery) nadměrně zatěžoval letovou sběrnici a tím ohrožoval přísun životně důležitých informací pro řízení letu. Také případná porucha či nevhodný návrh užitečné zátěže (která by se ta například neodpojila od sběrnice a trvale tak narušovala její provoz) už nemůže bezpečnost letu ohrozit. Díky této architektuře se tedy už nemusíme bát, že by nám různé typy užitečné zátěže jakkoli měnili poměry na letové sběrnici.

Z výše popsané sběrnicové architektury vybočují pouze jednotky bezdrátového spojení se zemí. Ty jsou, jak již bylo řečeno dvě, jedna zajišťující datové spojení s nadřízeným řídicím systémem a druhá je vlastně obyčejný modelářský přijímač, sloužící k přenosu povelů manuálního řízení z modelářské vysílačky. Protože jednotka datového spojení bude pravděpodobně WiFi access point (blíže viz. 3.4.2), který bude s hlavním řídicím počítačem spojen pomocí ethernetu, není účelné připojovat ji na sběrnici, ale je mnohem jednodušší připojit ji přímo k řídicímu počítači. Obdobný argument platí pro modelářský přijímač, který je mnohem jednodušší připojit přímo k jednotce řízení servomotorů, která bude zajišťovat i přepínání řízení automatika/manuál. Blokové schéma celého systému je na obrázku 3.2.



Obr. 3.2 – Blokové schéma řídicího systému

Protože jednotky GPS, výškoměr a elektronický kompas budou nějaké sériově vyráběné a běžně dostupné komerční produkty, nedá se předpokládat, že by byly vybaveny námi vyžadovaným interfacem pro připojení ke sběrnici CANBUS. Proto každá z těchto jednotek bude muset být ještě doplněna přídavným modulem, který bude z jedné strany připojen na sběrnici CANBUS a bude schopen komunikovat s nadřízenou jednotkou (řídicím počítačem) pomocí nějakého standardního komunikačního protokolu. Na straně druhé bude tento modul připojen k příslušné jednotce (GPS, výškoměr či elektronický kompas) a bude zajišťovat komunikaci s ní podle jejích požadavků. Tento modul tedy bude vlastně zastávat funkci jakéhosi „překladatele“ mezi konkrétní použitou jednotkou a nadřízeným řídicím systémem. Protože jednotky řízení servomotorů a inerciální navigace budou vyvíjeny naším týmem, budou mít samozřejmě interface odpovídající našim požadavkům a žádného „překladatele“ nebudou potřebovat.

Bylo rozhodnuto, že výše zmíněné moduly budou všechny postaveny na bázi mikroprocesoru HITACHI H8S2638F. Tento mikroprocesor je sice pro tento účel snad až zbytečně výkonný, nicméně jeho použití je výhodné z toho důvodu, že na jeho bázi bude postavena též autorem vyvíjená jednotka řízení servomotorů (viz. 3.4.7) a jednotka inerciální navigace (viz. 3.4.6). Proto budeme mít k disposici všechny vývojové prostředky pro tento procesor a budeme dostatečně sběhlí v jeho programování, což podstatně zkrátí dobu vývoje těchto jednotek.

3.4 Použitý hardware

3.4.1 Hlavní řídicí počítač

Jako hlavní řídicí počítač letu plánujeme použít průmyslové PC, konkrétně Microspace PC104, typové označení MSM586SEV. Jedná se o průmyslové PC klasického PC104 formátu, vybavené Procesorem ELAN 520 na 133 MHz se 128 MB RAM. Má všechny standardní periferie, jako je např. ethernetová karta, sériové a paralelní porty, rozhraní nejen pro IDE harddisk, ale i flash disk, případně tzv. DiskOnChip. Napájení je stejnosměrným napětím o velikosti 5V, proudový odběr je 900 mA. Bližší informace lze získat z datasheetu, ketý je k disposici na [7]. Toto PC je pro účely komunikace po sběrnici CANBUS vybaveno CANBUSovou kartou ADVANTECH PCM-3680, blíže viz. [8].

3.4.2 Jednotky bezdrátového spojení se zemí

Jednotky bezdrátového spojení se zemí jsou, jak již bylo řečeno, celkem dvě, na sobě zcela nezávislé. Jedná se o jednotku datového spojení a jednotku přenosu manuálního řízení. Jednotka přenosu manuálního řízení je standardní modelářský přijímač, připojený na jednotku řízení servomotorů. Bližší popis této jednotky je v článku 3.4.7.

Druhou bezdrátovou jednotkou je jednotka datového spojení se zemí. Tato jednotka slouží k přenosu povelů do vrtulníku, on-line telemetrie z vrtulníku, nahrávání nových verzí řídicího algoritmu do řídicího počítače, případně přenos GPS údajů ze země (při použití diferenciální GPS, viz. článek 3.4.4). Po konzultaci s panem Marcem Sanvidem bylo rozhodnuto pro tento účel použít standardní modul WiFi (Wireless Fidelity, bezdrátová síťová karta). Nejspíše bude použit WiFi Acces Point, který umožňuje jednak ethernetové spojení s řídicím počítačem (tj. přes jeho standardní síťovou kartu), jednak připojení větší externí antény – pravděpodobně použijeme všesměrovou anténu se ziskem 6 dB.

Protože je zřejmé, že připojování těchto jednotek na letovou sběrnici by bylo pouze nesmyslnou komplikací a nepřineslo by žádný užitek, jsou tyto dvě jednotky připojeny přímo k jim nadřízeným modulům.

3.4.3 Elektronický kompas

Elektronický kompas bude poskytovat informaci o orientaci vrtulníku vzhledem ke světovým stranám. Ačkoli kompas je striktně vzato redundandní (tuto informaci by také bylo možno získat z jednotky inerciální navigace), je potřebný právě kvůli korekci možných chyb této jednotky. Údaj z jednotky inerciální navigace o celkové poloze vrtulníku ve světovém souřadném systému bude totiž zatížen s časem stále rostoucí chybou, způsobenou například driftem senzorů, vibracemi vrtulníku, které budou ovlivňovat údaje naměřené akcelerometry, a jiné vlivy. Díky údajům z GPS a kompasu můžeme tuto chybu korigovat. Kompas se také bude požívat pro plánování dlouhodobější trajektorie letu, údaje z inerciální navigace pouze pro okamžité řízení vrtulníku (tj. především jeho stabilizaci).

Konkrétní typ elektronického kompasu dosud nebyl vybrán.

3.4.4 Jednotka GPS

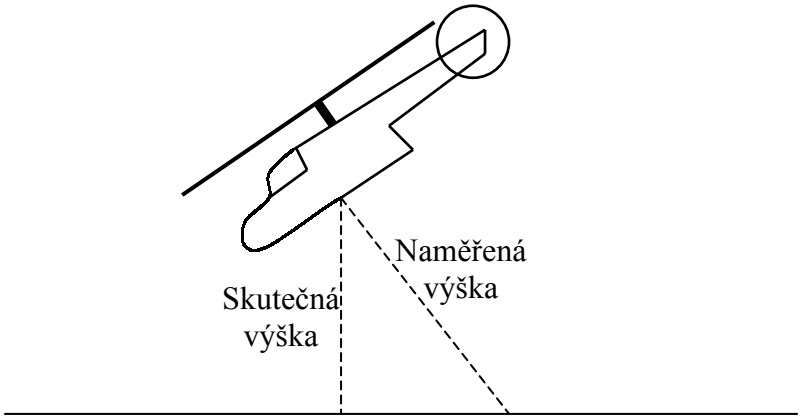
Účel GPS je zřejmý. Údajů z GPS bude použito především k dlouhodobému plánování trajektorie letu a ke korekcím chyb jednotky inerciální navigace.

Pro dosažení vyšší přesnosti je výhodné požít tzv. diferenciální GPS, kdy jedna přijímací stanice je umístěna na zemi ve známém bodě (referenční stanice) a druhá je nesena ve vrtulníku. Údaj z referenční stanice se používá ke korekci údaje stanice nesené vrtulníkem. Chyba jedné stanice se pohybuje v řádu desítek metrů, chyba diferenciální GPS je dle zkušeností z projektu ETH menší než 2cm. Diferenciální GPS je dokonce tak přesná, že podle ní (ve spolupráci s přesným výškoměrem) lze automaticky řídit i start a přistání vrtulníku, jak prokázaly výsledky z projektu na ETH a také vrtulníkového projektu na MIT.

Konkrétní typ jednotky GPS nebyl dosud vybrán.

3.4.5 Výškoměr

Výškoměrem je míněn přístroj udávající skutečnou výšku nad terénem, nikoli výšku nadmořskou. Proto nepoužijeme klasický barometrický výškoměr, ale ultrazvukový či laserový. Údaj z tohoto typu výškoměru bude samozřejmě nutné přepočítat na skutečnou výšku nad zemí, protože měří vzdálenost od země v souřadném systému vrtulníku a je tak závislý na náklonu stroje, viz. obr. 3.3. Údaje z výškoměru budou důležité zejména pro přistávání.



Obr. 3.3 – Zkreslení údaje o výšce vlivem náklonu

Konkrétní typ výškoměru nebyl dosud vybrán.

3.4.6 Jednotka inerciální navigace

Jednotka inerciální navigace je z hlediska automatického řízení vrtulníku nejzásadnější. Poskytuje informaci o poloze a orientaci vrtulníku v prostoru. Protože vrtulník má 6 stupňů volnosti, bude v této jednotce minimálně 6 senzorů, jmenovitě 3 akcelerometry měřící zrychlení ve třech na sebe kolmých směrech a 3 náklonoměry (nebo spíš gyra), měřící rychlosti otáčení kolem tří hlavních os vrtulníku.

Z údajů těchto senzorů se dá vypočítat 12 stavových veličin vrtulníku (viz. část 3.5.2.2) – rychlosti otáčení kolem tří hlavních os, úhly aktuálního natočení kolem těchto os, zrychlení vrtulníku ve třech hlavních směrech, aktuální rychlosti v těchto směrech a konečně poloha referenčního bodu na vrtulníku (těžiště) v prostoru (ve světových souřadnicích).

Jednotka inerciální navigace je vyvíjena na VABO v Brně a bylo dohodnuto, že bude použita jak pro jejich bezpilotní letoun, tak pro náš vrtulník.

3.4.7 Jednotka řízení servomotorů

3.4.7.1 Úvod

V této podkapitole bude popsán koncepční návrh jednotky servomotorů, jež byla již mnohokrát zmiňována v předchozím textu. Tato jednotka je potřebná, protože všechny akční členy vrtulníku jsou ovládány pomocí servomotorů.

Jednotka řízení servomotorů musí plnit několik úkolů. Její hlavní funkcí je přijímat po sběrnici CANBUS povely z nadřízeného systému a podle nich nastavovat polohy jednotlivých servomotorů. Tento úkol ovšem není tak přímočarý, jak by se na první pohled mohlo zdát. Předpokládáme totiž, že povely budou mít význam jako například „nastav kolektivní řízení na 5°," „nastav cyklické řízení na 2° doleva" a podobně. Problém spočívá v tom, že například cyklické a kolektivní řízení je mechanicky realizováno pomocí zařízení, jež je ovládáno celkem třemi servomotory, jejichž polohy se musí správně nastavit, aby bylo dosaženo kýženého efektu. Pohyb těchto servomotorů tedy musí být vhodně „smixován". Takže například na povel „nastav cyklické řízení na 2° doleva" musí jednotka „vhodně" nastavit hned 3 servomotory a jejich pohyb korelovat tak, aby přechod do cílového stavu byl „beznárazový".

Další funkcí zajišťovanou touto jednotkou je přepínání ovládání servomotorů mezi automatickým a manuálním řídicím systémem. Jednoduše řečeno, jednotka řízení servomotorů buď servomotory řídí sama podle povelů nadřízeného systému, nebo jejich vstupy připojí přímo na modelářský přijímač, který přijímá povely řídicí vysílačky na zemi, kterou ovládá lidský pilot. Je vhodné, aby toto přepínání jednotka realizovala nikoli na pokyn z nadřízeného systému, ale aby tato funkce byla řízena přímo jedním kanálem z modelářského vysílače na zemi. Je také vhodné, aby tato funkce byla realizována na co nejnižší úrovni řídicí logiky, aby bylo možno přepnout řízení na manuál i tehdy, kdyby například došlo k „zamrznutí" řídicího procesoru nebo nadřízené řídicí jednotky.

V případě, že řízení je přepnuto na manuál, stačí pouze propojit vstupy servomotorů s výstupy modelářského přijímače bez jakýchkoli dalších zásahů. „Mixování" poloh jednotlivých servomotorů podle žádané akce totiž v tomto případě zajišťuje modelářský vysílač na zemi a na výstupech přijímače už je přímo PWM[1] signál pro příslušné

[1] Všechny použité servomotory mají vlastní jednoduchou řídicí jednotku, která zajišťuje jejich silové buzení. Každý servomotor má jediný logický vstup, kompatibilní s TTL logikou. Na vstupu se očekává PWM signál s periodou délky přibližně 20 ms, klidová hodnota je v logické 1. Šířkou pulsu do logické 0 se určuje úhel

servomotory. Jednoho kanálu přijímače bude, jak již bylo řečeno, využito k přepínání řízení. Tento kanál má také PWM výstup, přepínání tedy bude fungovat například tak, že na plnou levou výchylku přepne na automatické řízení a na plnou pravou naopak na řízení manuální. O změně řízení bude jednotka vždy informovat nadřízený řídicí systém.

S přepínáním mezi automatickým a manuálním řízením souvisí ještě problematika beznárazového přepínání řízení, která je u vrtulníku obzvlášť důležitá. Prudká změna nastavení akčních členů při předání řízení by mohla vrtulník natolik destabilizovat, že jeho vyvedení z nastalé nebezpečné letové situace by bylo velmi obtížné a mohlo by vést dokonce až k havárii stroje. Při přepnutí z automatického řízení na manuální není bohužel beznárazové předání řízení dost dobře možné, protože není možno nastavit polohu řídicích pák na vysílači podle aktuální polohy servomotorů. Při opačném přepnutí, tj. z řízení manuálního na automatické je ovšem možné a nutné předat nadřízenému systému informaci o aktuální poloze servomotorů (respektive přímo o nastavení jednotlivých akčních prvků, tj. ve formátu například „kolektivní řízení je 5°," „cyklické řízení je 2° doleva" apod.). Je potom na nadřízeném řídicím systému, aby se s nastalou situací vyrovnal.

S funkcí popsanou v předchozím odstavci úzce souvisí poslední služba zajišťovaná jednotkou řízení servomotorů, a to je informování nadřízeného systému o aktuálním stavu akčních členů při manuálním řízení, pokud se tento zeptá. Tato funkce je velmi užitečná pro potřeby identifikačních experimentů, kdy bude vrtulník řízen lidským pilotem a řídicí počítač bude ukládat vzorky aktuálního stavu systému (tj. vrtulníku) získané z jednotky inerciální navigace a spolu s nimi bude ukládat také aktuální stav všech akčních členů.

3.4.7.2 Geometrie táhlových spojů

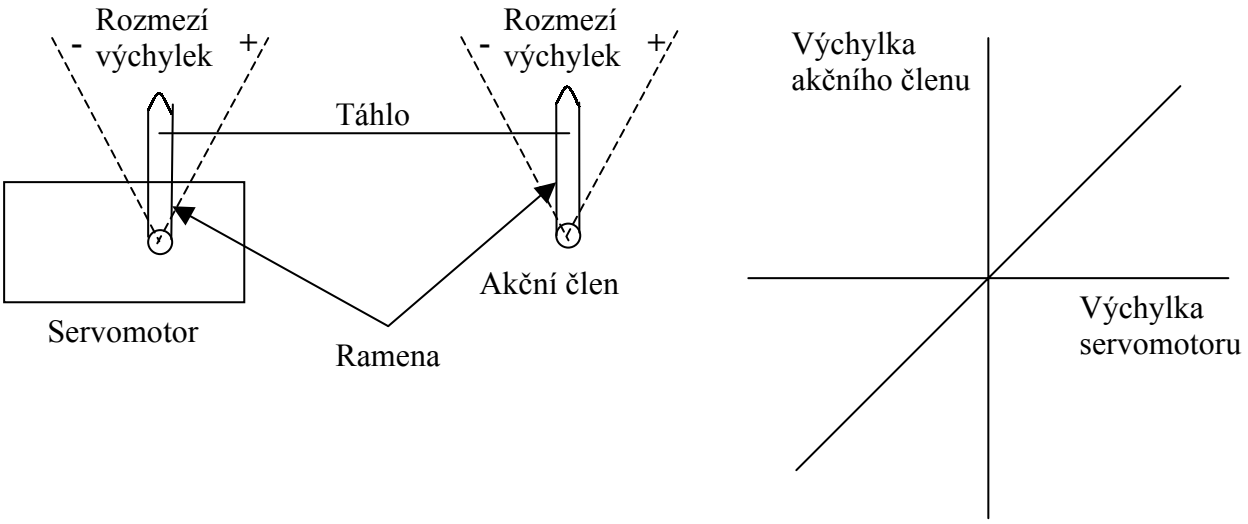
V této podkapitole bude stručně popsána mechanická konstrukce táhlových spojů a jejich vliv na řízení vrtulníku. Všechny mechanické akční členy, ať už je to mechanika kolektivního a cyklického řízení rotorových listů nebo například karburátor spalovacího motoru, jsou s jimi příslušejícími servomotory propojeny pomocí táhel. Mechanická konstrukce a nastavení táhel mají zásadní vliv na přenos výchylky hřídele servomotoru na příslušný akční člen, a tím i na akční zásah touto výchylkou vyvolaný. Jedná se o problematiku velmi podstatnou z hlediska návrhu řídicí jednotky servomotorů.

Táhla tvoří mechanické spojení mezi servomotory a mechanickými akčními členy, a jejich správná konstrukce a nastavení jsou nezbytné pro úspěšné řízení vrtulníku. Většinou se realizují buď jako bowdeny nebo na krátké vzdálenosti se vyrábí z ocelových tyček. Táhla na vrtulníku tvoří mnohdy složité mechanické řetězce, protože servomotor a akční člen vzhledem k jejich umístění často není možno spojit jednoduchým přímým táhlem. Táhla tvoří v naprosté většině případů spojnice mezi rameny servomotorů a obdobnými rameny akčních členů. Pomocí délek těchto ramen lze mechaniky nastavovat citlivost spoje, ale chybným nastavením je také možné zavést si do řídicího systému zbytečné nelinearity. Příklad jednoduchého táhlového spoje je na obrázku 3.4.

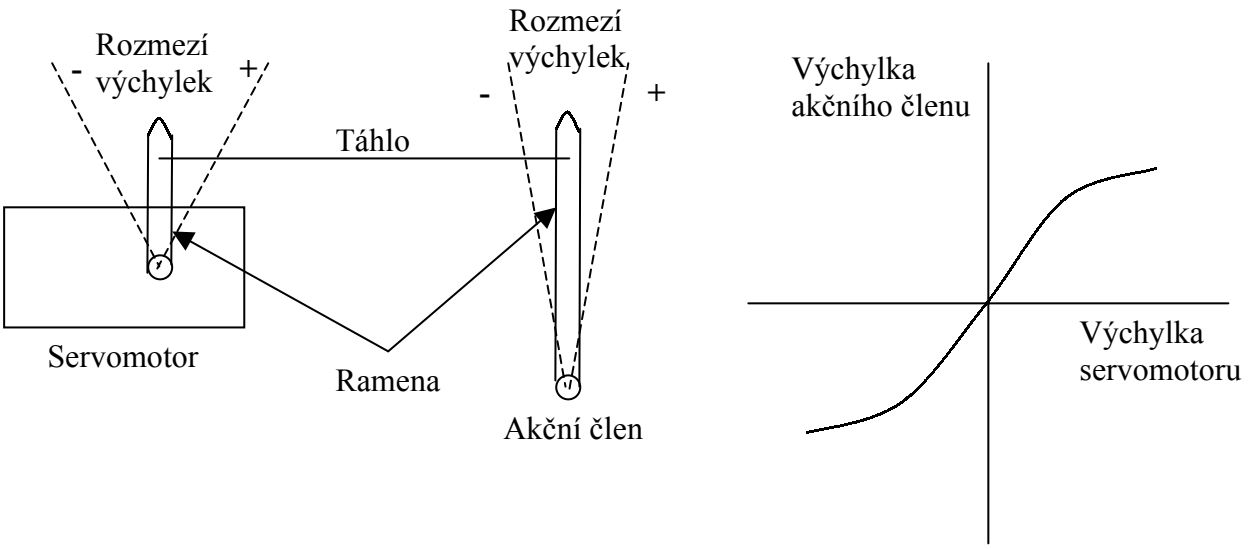
Z obrázku 3.4 vidíme, že pokud jsou délky obou ramen stejné, je závislost výchylky ramene akčního členu na výchylce ramene servomotoru lineární. Pokud prodloužíme rameno servomotoru, dosáhneme tím vyšší citlivosti spoje, ovšem za cenu nelineární závislosti obou výchylek. Pokud naopak prodloužíme rameno u akčního členu, citlivost se nelineárně zmenší, viz. obrázky 3.5 a 3.6.

---

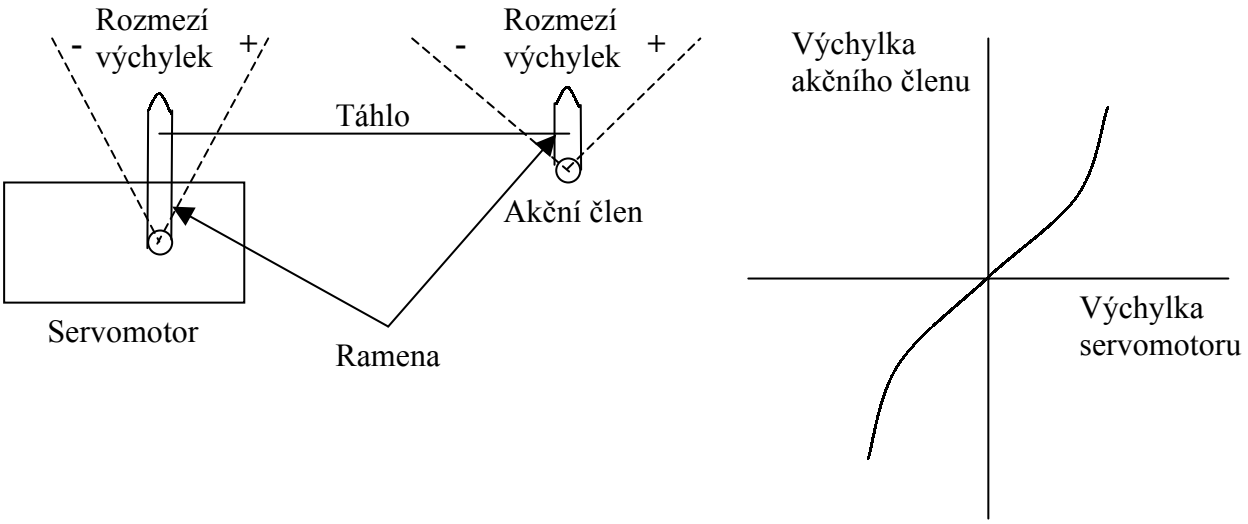
natočení hřídele servomotoru. Nulová šířka pulsu, tj. stálé kladné napětí na vstupu značí plnou levou výchylku, naopak puls široký přes celou periodu, tj. vstup trvale v logické 0 znamená plnou pravou výchylku.



Obr. 3.4 – Příklad jednoduchého táhlového spoje



Obr. 3.5 – Příklad nelineárního snížení citlivosti spoje

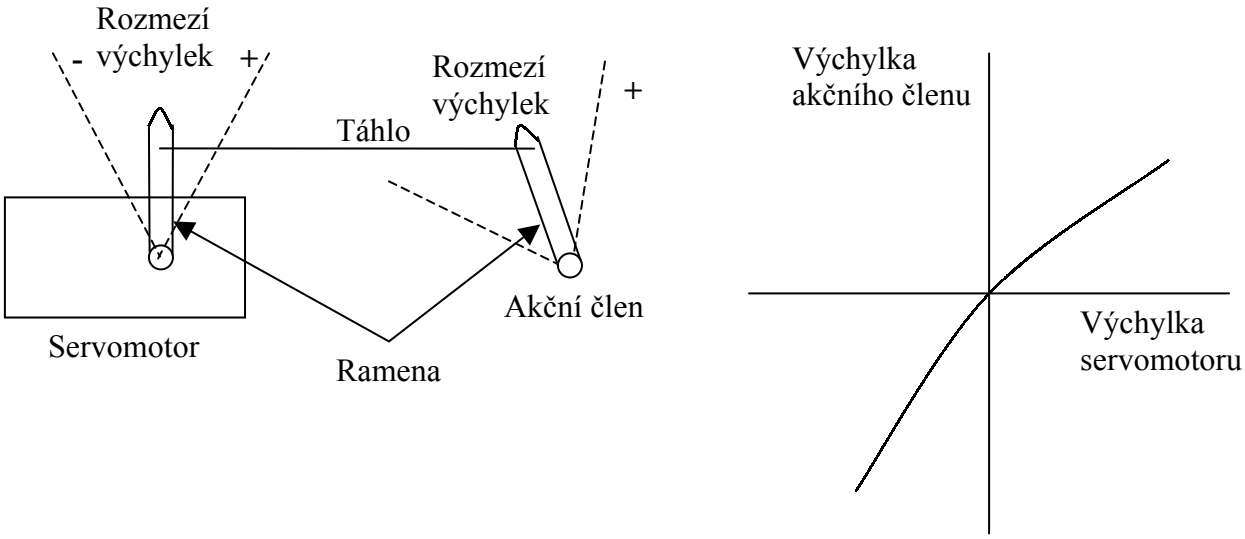


Obr. 3.6 – Příklad nelineárního zvýšení citlivosti spoje

Poznámka autora: Tyto nelineární, tzv. exponenciální výchylky záměrně používají zejména akrobatičtí piloti. Například prodloužením ramene servomotoru se dosáhne možnosti velmi razantních zásahů do řízení při maximálních výchylkách, ovšem v okolí klidových poloh není řízení nadměrně citlivé, což je pro lidského pilota žádoucí. Z hlediska automatického řízení jsou ovšem nelineární výchylky nevýhodné. V některých případech lze ale záměrně zavedených nelinearit v táhlovém spoji využít k mechanické kompenzaci nelineárního chování některých akčních členů.

Nežádoucí nelinearity můžeme ale do táhlového spoje vnést i jiným způsobem. Například tak, že obě ramena spolu nebudou rovnoběžná, jak je znázorněno na obrázku 3.7. Takovýto druh nelinearity v řízení je naprosto nežádoucí a snažíme se mu proto za každou cenu vyhnout. Z tohoto důvodu se vždy snažíme montovat táhla tak, aby v klidové poloze byla k oběma ramenům kolmá.

Abychom minimalizovali síly v řízení, působící na servomotor a jeho uchycení, snažíme se maximálně prodloužit délku obou ramen a tím i páku, na kterou kroutivý moment servomotoru působí. Proto, pokud máme na výběr a konstrukce modelu nám to umožňuje, používáme vždy úchyt táhla zcela na vnějším konci obou ramen. Toto pravidlo je zvlášť důležité u servomotorů kolektivního a cyklického řízení, protože na ně působí v provozu největší síly.



Obr. 3.7 – Nevhodné seřízení ramen táhlového spoje

Poznámka autora: Tato podkapitola byla zpracována s využitím poznatků uvedených v [5].

3.4.7.3 Bloková struktura jednotky řízení servomotorů

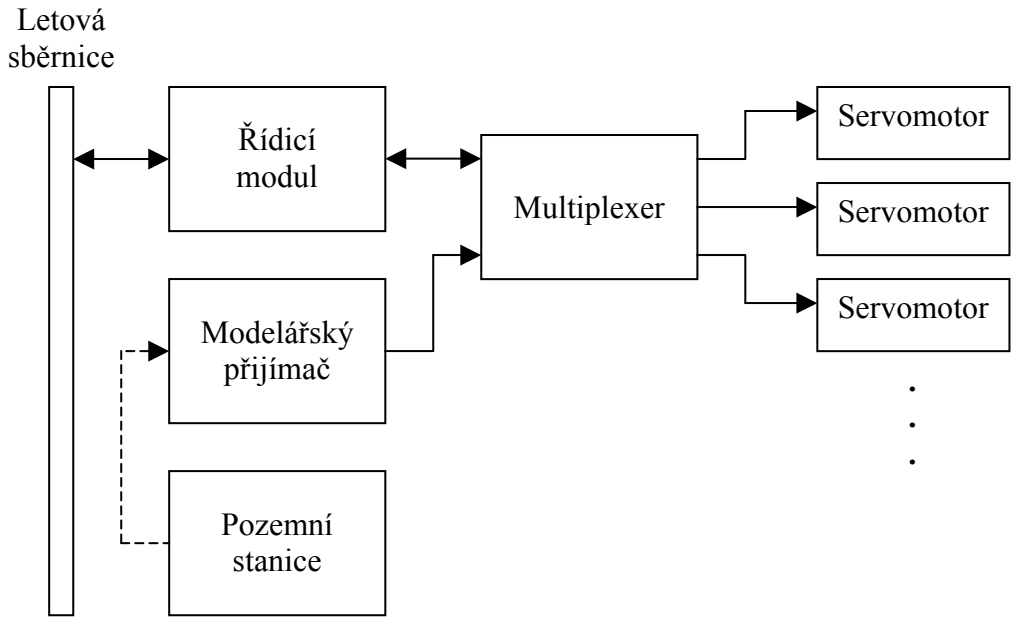
V této části budeme popisovanou jednotku dekomponovat do blokové struktury a popíšeme funkci jednotlivých bloků.

Vyjděme z požadavků, popsaných v části 3.4.7.1. Jednotka má přijímat povely z letové sběrnice a podle nich řídit servomotory, a současně má přepínat mezi automatickým a manuálním řízením, a to na povel přišedší z modelářského přijímače a mající formu PWM signálu. Z toho můžeme usoudit na základní blokovou strukturu jednotky, jenž je zobrazena na obrázku 3.8. Jednotka se nám rozpadá na 2 základní bloky; na „řídicí modul," který přijímá povely ze sběrnice a „multiplexer," který vstupy servomotorů připojuje buď na výstupy řídicího modulu nebo na výstupy modelářského přijímače, a to podle stavu „řídicího kanálu"

přijímače, tj. kanálu, kterým se ze země přepíná způsob řízení. Jsou-li vstupy servomotorů připojeny na modelářský přijímač, musí multiplexer umožnit řídicímu modulu jejich čtení.

„Řídicí modul“ je postaven na bázi mikroprocesoru HITACHI H8S2638F. Jedná se o velmi výkonný mikroprocesor, který má širokou paletu hardwarových periferií, z nichž jsou pro naše účely podstatné především dvoukanálový budič sběrnice CANBUS, až 16 samostatně nastavitelných PWM výstupů a interní watchdog.

„Multiplexer“ bude naprogramován do nějakého vhodného hradlového pole (FPGA).



Obr. 3.8 – Blokové schéma jednotky řízení servomotorů

3.4.8 Pozemní stanice

Pozemní stanice slouží ke komunikaci s vrtulníkem, zobrazování telemetrických údajů, vývoji softwaru pro vrtulník, obsluze pozemního přijímače GPS atd. Švýcarští kolegové z ETH ve svém projektu použili dva laptopy, z nichž jeden byl použit jako řídicí stanice pro vrtulník (posílal k vrtulníku údaje z pozemního přijímače GPS a povely, naopak přijímal a on-line zobrazoval vybrané telemetrické údaje) a druhý byl použit pro vývoj řídicího softwaru a nahrávání jeho nových verzí do řídicího počítače vrtulníku. Třetí částí byla samozřejmě modelářská vysílačka pro manuální řízení vrtulníku.

3.4.9 Použité modelářské vybavení

V této části vyjmenujeme a velmi stručně popíšeme hlavní komponenty modelářského vybavení použitého na našem vrtulníku.

Námi požitý model Robbe Futura 1.8 SE je poměrně moderní cvičný model vrtulníku s dvoutaktním spalovacím motorem o zdvihovém objemu 15cm^3. Průměrem rotoru 1,8m se řadí mezi větší modely. Unese užitečnou zátěž o hmotnosti asi 3-5kg, podle umístění vzhledem k těžišti.

K řízení je použita moderní RC souprava Graupner mc-20, která se řadí ke světové špičce. Je to počítačem řízená šestikanálová souprava s mnohými speciálními funkcemi pro řízení vrtulníku.

K řízení kolektivu, cykliky a plynu jsou použity servomotory Graupner DS 8231, pro řízení pomocného rotoru potom speciální ultrarychlé digitální servo Graupner DS 8418. Ke stabilizaci ocasu je použito špičkové piezoelektrické počítačem řízené gyro ICG 540. Pro řízení otáček motoru je použit samostatný regulátor Graupner Heli-control.

K napájení jsou použity 2 sady Ni-CD akumulátorů 6V/1600mAh, jedna hlavní a druhá záložní, mezi nimi je zapojena elektronická rozbočka napájení, která automaticky přepne na záložní sadu, dojde-li k vybití sady hlavní.

3.5 Matematické modely a systémový popis vrtulníku

3.5.1 Úvod

V této části stručně shrneme způsoby popisování vrtulníku ze systémového hlediska a způsoby jeho matematického popisování.

Jak již bylo několikrát uvedeno v předchozím textu, vrtulník je nelineární nestabilní systém se šesti stupni volnosti a s velmi malým přirozeným tlumením. Z hlediska automatického řízení má vrtulník na nejnižší úrovni 3 vstupy (úhel náběhu listů hlavního rotoru, úhel náběhu listů pomocného rotoru a otáčky, resp. kroutivý moment motoru) a 6 výstupů (3 souřadnice polohy referenčního bodu a 3 natočení kolem hlavních os).

Z našeho pohledu budou ovšem vstupní veličiny vypadat poněkud jinak, protože použijeme několika nám už známých pomocníků: jedním vstupem budou žádané otáčky motoru (o jejich stabilizaci se bude starat standardní PID regulátor otáček), další bude kolektivní úhel náběhu rotorových listů, další 2 vstupy bude tvořit cyklické řízení (azimut a dodatečný maximální úhel náběhu) a nakonec žádaná rychlost otáčení kolem osy rotoru, o jejíž dodržování se postará gyro. Dohromady tedy budeme mít 5 vstupů.

Nadefinujme pro začátek na vrtulníku vhodný systém, který bude beze zbytku popisovat pro nás zajímavé veličiny.

3.5.2 Systémový popis vrtulníku

3.5.2.1 Souřadné systémy

Nadefinujme nejprve souřadné systémy, ve kterých budeme vrtulník popisovat. Je výhodné zavést tyto systémy dva – tzv. světový souřadný systém spojený se zemí a palubní souřadný systém vrtulníku.

V souladu s [1] definujme světový souřadný systém jako kartézský, s počátkem na zemi v místě přijímače referenční GPS a s osou x orientovanou k severu, osou y na východ a osou z do země.

Palubní souřadný systém je taktéž kartézský, počátek má umístěn v těžišti vrtulníku a osy x,y a z jsou po řadě orientovány dopředu, doprava a dolů.

Transformační matice rotace z palubního do světového souřadného systému je

$$C_p^s = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad (3.1)$$

kde θ je úhel klopení (kolem podélné osy), ϕ je úhel klonění (kolem příčné osy) a ψ je úhel otočení (kolem osy rotoru). Odvození transformace (3.1) je zřejmé, a proto je nebudeme uvádět (lze je provést například pomocí eulerových úhlů).

3.5.2.2 Definice systému na vrtulníku

Nyní na vrtulníku nadefinujeme systém, který bude popisovat pro nás zajímavé veličiny a vztahy mezi nimi. Těmito veličinami jsou p_x, p_y a p_z, což jsou souřadnice těžiště vrtulníku ve světové souřadné soustavě, dále v_x^s, v_y^s a v_z^s což jsou složky rychlosti, kterou se vrtulník pohybuje, měřené ve směru odpovídajících os světové souřadné soustavy (horní index *s* znamená, že tyto veličiny jsou měřeny ve světové souřadné soustavě, naopak horní index *p* znamená palubní souřadnou soustavu). Dále máme těmto rychlostem odpovídající zrychlení a_x^s, a_y^s a a_z^s, potom úhly natočení vrtulníku podle podélné (klonění), příčné (klopení) a rotorové (zatáčení) osy ϕ, θ a ψ, a jim odpovídající úhlové rychlosti p, q a r.

Protože senzory na palubě vrtulníku měří zrychlení nikoli ve směru os světové, ale palubní souřadné soustavy, zavedeme si ještě tři pomocné veličiny a_x^p, a_y^p a a_z^p, což jsou právě zmiňovaná zrychlení měřená ve směru os palubní souřadné soustavy. Zřejmě platí převodní vztah (viz. článek 3.5.2.1)

$$\begin{bmatrix} a_x^s \\ a_y^s \\ a_z^s \end{bmatrix} = C_p^s \begin{bmatrix} a_x^p \\ a_y^p \\ a_z^p \end{bmatrix} \tag{3.2}$$

Kde matice C_p^s je definována vztahem (3.1). Nyní již máme vše připraveno pro sestavení systémových rovnic, popisujících vztahy mezi výše popsanými stavovými veličinami. Systémové rovnice mají následující tvar:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{v}_x^s \\ \dot{v}_y^s \\ \dot{v}_z^s \\ \dot{p}_x^s \\ \dot{p}_y^s \\ \dot{p}_z^s \end{bmatrix} = \begin{bmatrix} T_\omega \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ C_p^s \cdot \begin{bmatrix} a_x^p \\ a_y^p \\ a_z^p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ v_x^s \\ v_y^s \\ v_z^s \end{bmatrix} \tag{3.3}$$

Kde matice rotace T_ω má tvar:

$$T_\omega = \begin{bmatrix} 1 & \dfrac{\sin\phi\sin\theta}{\cos\theta} & \dfrac{\cos\phi\cos\theta}{\cos\theta} \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{3.4}$$

Rozeberme si nyní význam jednotlivých rovnic. Poslední tři rovnice ve vztahu (3.3) zřejmě jen vyjadřují známý fyzikální zákon, že rychlost je časovou derivací polohy. Obdobně zrychlení je časovou derivací rychlosti, což vyjadřují další tři rovnice – ovšem protože ve sloupcovém vektoru na levé straně máme příslušné veličiny ve světových souřadnicích, zatímco jim odpovídající zrychlení na straně pravé jsou v souřadnicích palubních, musíme použít transformaci (3.2). Nakonec ještě musíme ke zrychlení ve směru osy z přičíst tíhové zrychlení g. Konečně první tři rovnice vyjadřují pouze fakt, že úhlové zrychlení je derivací úhlové rychlosti. Ze stejných důvodů jako při transformaci zrychlení musíme požít rotační matici T_ω definovanou ve vztahu (3.4) – příslušná úhlová zrychlení totiž měří senzory pevně umístěné na vrtulníku, a tudíž v palubní souřadné soustavě, zatímco úhly ϕ, θ a ψ jsou měřeny od pevně daných os.

Výše uvedený systém byl převzat z literatury [1] a jak vidno je nelineární. Je použit pro výpočet veličin, které nemůžeme na vrtulníku přímo měřit, ale přesto je potřebujeme pro jeho řízení. Tyto veličiny získáme integrací rovnic (3.3). Pro praktický výpočet se používá Kalmanův filtr.

3.5.2.3 Linearizovaný matematický model vrtulníku ve visu

V literatuře [1] je prezentován jednoduchý lineární matematický model vrtulníku, který je použitelný pro vrtulník nacházející se ve visu nebo pomalém letu. Odchylky od roviny xy jsou v tomto případě velmi malé, a goniometrické funkce lze proto snadno linearizovat. Model taktéž předpokládá konstantní otáčky rotoru, tento předpoklad je ovšem v praxi dobře splněn, viz. článek 6.1.5.4.

Model vypadá takto:

$$\begin{aligned}
\dot{p}_x^p &= v_x^p \\
\dot{v}_x^p &= \alpha_1 \theta \\
\dot{\theta} &= q \\
\dot{q} &= \alpha_2 q + \alpha_3 B_{1s} + \alpha_4 A_{1s} \\
\dot{p}_y^p &= v_y^p \\
\dot{v}_y^p &= \alpha_5 \phi \\
\dot{\phi} &= p \\
\dot{p} &= \alpha_6 p + \alpha_7 B_{1s} + \alpha_8 A_{1s}
\end{aligned} \tag{3.5}$$

Kde A_{1s} je akční zásah boční cykliky a B_{1s} je akční zásah předozadní cykliky. Význam první rovnice je zřejmý – rychlost je časovou derivací polohy (úhlová rychlost je časovou derivací úhlové odchylky) tento zákon vyjadřují také 3., 5. a 7. rovnice v soustavě (3.5). 2. rovnice říká, že dopředná rychlost je pro malý úhel klopení θ na tomto úhlu lineárně závislá. Totéž, ale pro boční rychlost říká rovnice 6. 4. rovnice vyjadřuje závislost změny úhlové rychlosti klopení na aktuální úhlové rychlosti a cyklickém akčním zásahu. Totéž pro klonění vyjadřuje poslední rovnice soustavy (3.5).

Existuje velké množství používaných matematických modelů vrtulníku, různě složitých, lineárních i nelineárních. Složitější modely berou například v úvahu též dynamiku

rotoru (pohyb rotorových listů, viz. článek 6.1.4.1). Drtivá většina těchto modelů bývá ovšem použitelná pouze pro specifické režimy letu.

3.6 Způsoby automatického řízení vrtulníku

3.6.1 Úvod

Existuje celá řada různě složitých regulátorů pro vrtulníky, navržených pomocí klasických i moderních metod. Většina těchto regulátorů bývá, podobně jako je tomu u matematických modelů, určena pro specifické režimy letu. Například pro stabilizaci vrtulníku ve visu lze použít kaskádně zapojené PD regulátory či vícerozměrný PID regulátor (viz. [9]), je možné použít LQ nebo lépe LQG regulátor (viz. [10]) či v poslední době též nějakou formu H_∞ regulátoru. Existují též úspěšné pokusy o nasazení fuzzy regulátorů, emulujících chování živého pilota.

Všechny tyto regulátory mají jedno společné – musí být dostatečně robustní, protože žádný matematický model není dostatečně přesný a nepopisuje zcela věrně všechny vlastnosti skutečného vrtulníku. Rozsah jejich použití je proto omezen rozsahem použitého matematického modelu.

3.6.2 Popis jednotlivých typů regulátorů

3.6.2.1 „Klasické“ regulátory

Pokusy s automatickým řízením vrtulníku pomocí klasických (rozumí se PD nebo PID) regulátorů se datují do 80. let minulého století. Vzhledem rychlé dynamice systému a menším nárokům na přesnost řízení se zpravidla používá PD regulátorů. Stabilizující regulátor postavený na bázi kaskádně zapojených PD regulátorů byl použit například M. F. Weilenmannem v roce 1988 na ETH v Curychu. Regulátory tohoto typu mají zpravidla nastaveny parametry vhodné pouze pro nějaký specifický režim letu, a rozsah jejich použitelnosti je tedy omezen. Příkladem jejich použití může být například stabilizace vrtulníku ve visu.

Protože u těchto regulátorů se dá jen nesnadno zajistit jejich robustnost, nemluvě už vůbec o optimalitě řízení, používá se jich v současné době zpravidla už jen pro dílčí úkoly, například pro udržování konstantních otáček motoru nebo pro řízení pomocného rotoru za účelem stabilizace ocasu.

3.6.2.2 Moderní typy regulátorů

V současné době se pro účely komplexního řízení vrtulníků používá spíše LQ či LQG regulátorů. S úspěchem bylo také použito robustních zpětnovazebních regulátorů postavených na H_∞ technikách.

S výhodou se používá zejména LQG regulátorů. Protože jednotka inerciální navigace nám zajišťuje měření všech stavových veličin systému, jak jsme si je definovali v části 3.5.2.2, můžeme snadno uzavřít stavovou zpětnou vazbu. Další výhodou je fakt, že u LQG regulátoru lze relativně snadno zajistit beznárazové přepínání řízení z manuálního na automatické. Lze to uskutečnit například tak, že v okamžiku přepnutí si systém zapamatuje poslední vzorek polohy všech servomotorů (poslední stav řízení lidského pilota) a k němu příslušející stav systému (tj. hodnoty všech systémových veličin vrtulníku). Poslední stav řízení se potom přičítá k výstupu regulátoru a poslední stav systému se přičítá k jeho vstupu. V neposlední řadě je velkou výhodou tohoto regulátoru optimalita regulačního pochodu a snáze zajistitelná robustnost než u „klasických“ regulátorů.

3.6.2.3 Fuzzy regulátory

Fuzzy regulátory se snaží emulovat chování živého pilota. Jsou vybaveny souborem pravidel ve tvaru „když (jsou splněny nějaké podmínky) tak (následuje vhodná akce)“. Vstupní veličiny jsou na vstupu regulátoru pomocí vhodných funkcí takzvaně „fuzzyfikovány,“ tj. jejich přesné číselné hodnoty jsou převedeny do slovní reprezentace, jako například (velmi hodně, hodně, středně, málo, velmi málo), tak, aby se daly snáze použít v podmínkových částech jednotlivých pravidel. Výsledná akce je vlastně reprezentací výstupních veličin ve „fuzzyfikované“ formě, takže na výstupu musí následovat tzv. „defuzzyfikace,“ tj. převod „fuzzy“ veličin zpět na číselné hodnoty.

Uveďme si jako příklad princip fuzzy řízení pomocného rotoru. Na vstupu regulátoru jsou dvě hodnoty – jednak požadovaná úhlová rychlost zatáčení vrtulníku (včetně směru) a skutečná rychlost zatáčení, změřená jednotkou inerciální navigace. Výstup nastavuje hodnotu úhlu náběhu listů pomocného rotoru.

Nejprve vypočteme rozdíl žádané a skutečné hodnoty úhlové rychlosti zatáčení. Řekněme, že nám vyšlo například (-0,08rad/s) (znaménko určuje směr otáčení, například znaménko – bude doleva). Fuzzifikační funkce tuto hodnotu fuzzifikuje, tj. převede do slovní reprezentace. Tato funkce může vypadat například takto:

Když ($|r| \prec 0{,}001$) tak (vůbec)

Když ($|r| \in \langle 0{,}001;0{,}01)$) tak (velmi málo)

Když ($|r| \in \langle 0{,}01;0{,}1)$) tak (málo)

Když ($|r| \in \langle 0{,}01;1)$) tak (středně)

Když ($|r| \in \langle 1;5)$) tak (hodně)

Když ($|r| \geq 5$) tak (velmi hodně)

Kde r je rozdíl žádané a skutečné hodnoty úhlové rychlosti zatáčení. Výše naznačená funkce přiřadí rychlosti zatáčení -0,08rad/s slovní hodnotu „středně“. Další fuzzifikační funkce určující směr otáčení vypadá například takto:

Když ($\mathrm{sgn}(r) \succ 0$) tak (doprava)

Když ($\mathrm{sgn}(r) \prec 0$) tak (doleva)

Která našemu číslu přiřadí hodnotu „doleva“. Potom se vybere z báze znalostí vhodné pravidlo. To může vypadat například následovně:

Když ((doleva) ∧ (středně)) tak ((doprava) ∧ (středně))

Následuje defuzzifikace, která funguje obdobně jako fuzzifikace. Tak se určí číselná hodnota akčního zásahu, která jde na výstup regulátoru.

Výhodou fuzzy regulátorů je možnost strojového učení pomocí různých metod umělé inteligence, především pak metod s učitelem, např. „teaching by showing“. Podrobnější popis těchto metod ovšem přesahuje rámec této práce (lze je najít například v [11]). Další jejich výhodou je jejich průhlednost a principielně jednoduchá funkce. Jejich nevýhodou je obtížné, ne-li nemožné exaktní dokazování stability a robustnosti těchto regulátorů, nemluvě už vůbec o optimalitě řízení.

3.7 Shrnutí

Na tuto část, pojednávající o automatickém řízení vrtulníku, je třeba nahlížet pouze jako na práci přehledovou, nezacházející do podrobností. Dokonce se ani nedá říct, že by byla kompletní rešerší, protože na skutečně seriózní prozkoumání a porovnání všech možností by bylo potřeba prostudovat mnohem více literárních zdrojů. Tento stav odráží časové možnosti, které jsem měl na zpracování této kapitoly k disposici. Za necelý jeden semestr, po který jsem se touto prací zabýval, se více stihnout nedalo.

Přesto je dle mého přesvědčení tato práce přínosem v tom, že alespoň naznačuje cesty, kudy se lze dále ubírat. Je také třeba vzít v úvahu, že před vlastním hledáním možností jak řídit vrtulník automaticky bylo nutné seznámit se s základními principy a mechanismem vrtulníkového letu. Ačkoli to není oblast přímo spadající do zorného úhlu této práce, bylo nutné tak učinit pro účely lepšího pochopení obtíží s tímto úkolem spojených a k pochopení způsobů matematického popisování vrtulníku. Výsledky této práce jsou shrnuty v příloze 6.1. Odvažuji se proto doufat, že přese všechny své nedostatky bude tato práce použita při budování základů našeho vlastního vrtulníkového projektu.

4. Závěr

Závěrem lze říct, že všech hlavních cílů uvedených v zadání bylo dosaženo. V této práci je kompletně popsána konstrukce automatické krmičky ryb odpovídající požadavkům uvedených v 2.5.1 a koncepční návrh řídicí jednotky modelu vrtulníku, včetně literární rešerše na toto téma. Na obou úkolech je demonstrována konstrukce a funkce vestavěných zařízení (elektronických řídicích jednotek).

Systém automatického krmení ryb byl nejen navržen a realizován, ale prošel úspěšně i náročnými praktickými testy a v průběhu času dospěl do reálně použitelné podoby, o čemž svědčí jeho dlouhodobý bezporuchový chod v průběhu srovnávacího experimentu (viz. 2.5.5) a příznivé hodnocení jeho uživatelů (viz. příloha 6.5). Myslím, že výsledky v těchto testech dosažené ukazují jasně přednosti (spolehlivý provoz, úspora krmiva i lidské práce) tohoto systému a ospravedlňují tak náklady vynaložené na jeho vývoj. V jeho prospěch hovoří také příznivé srovnání s konkurenčními výrobky (viz. 2.2.1.1) – především je to možnost jednoduchého začlenění do existujícího řídicího systému, přesnost odměřování malých krmných dávek a nízká cena. Možnosti budoucího vývoje jsou shrnuty v 2.5.7 – jedná se především o vylepšení mechanické konstrukce zařízení a testování alternativních typů pohonů.

Rešerši o automatickém řízení vrtulníku a návrh jednotky řízení servomotorů nelze sice považovat za kompletní, nicméně jsem přesvědčen, že vzhledem k časovým možnostem jsem i v tomto směru učinil vše co bylo možné a položil tak dobrý základ pro další práci na tomto projektu, jíž se, jak doufám, budu moci i nadále účastnit. Při hodnocení této části je třeba přihlédnout k faktu, že se jedná o nesmírně náročný projekt, nacházející se v současné době v nejrannějším stadiu své existence. Proto nelze od této práce očekávat žádné dalekosáhlé výsledky, ale právě jen přehledné shrnutí a načrtnutí možností a cest budoucího vývoje. Práce do budoucna na tomto projektu je mnoho – v nejbližší době je to především detailnější propracování konstrukce řídicí jednotky, návrh a stavba jednotlivých modulů a zabudování řídící jednotky do modelu vrtulníku.

Rád bych dodal, že práce na obou projektech byla pro mě kromobyčej poučná a umožnila mi v praxi uplatnit rozličné vědomosti, nabyté v uplynulých letech studia. Velmi si také cením toho, že jsem měl možnost pracovat na velkém a zajímavém projektu (projekt IFIBO) a vyzkoušet si tak práci ve velkém kolektivu lidí, zabývajících se různými obory (řídicí technikou a chovem ryb). Spolupráce s lidmi z firmy DataPartner (jmenovitě s ing. Liborem Švehlou) i s odborníky na chov ryb (především s ing. Pavlem Kozákem, Ph.D. a Dr. ing. Jitkou Hamáčkovou) probíhala na velmi dobré profesionální i osobní úrovni. Díky této spolupráci jsem také lépe porozuměl problematice chovu ryb a naopak biologové lépe pochopili možnosti a omezení moderní řídicí techniky. Jsem proto přesvědčen, že tato spolupráce byla oboustranně prospěšná a poučná a bude i nadále pokračovat mimo rámec této diplomové práce.

5. Seznam použité literatury

[1] M. Sanvido, *Control of helicopters.* http://www.eecs.berkeley.edu/~msanvido

[2] N. Wirth, *A computer system for model helicopter flight control.* Technical report #284, http://www.inf.ethz.ch

[3] M. Kottman, *Software for model helicopter flight control.* Technical report #316, http://www.inf.ethz.ch

[4] M. Sanvido, W. Schaufelberger, *Design of a framework for hardware-in-the-loop simulations and its application to a model helicopter.* http://www.control.ethz.ch

[5] C. Mill, *Practical theories.* http://www.w3mh.co.uk

[6] R.Jíra, J.Ticháček a kol., *Aerodynamika a mechanika letu.* Naše vojsko, Praha 1960

[7] http://www.digitallogic.com

[8] http://www.advantech.com

[9] J. John, *Systémy a řízení.* Vydavatelství ČVUT, Praha 1999

[10] V. Havlena, J. Štecha, *Moderní teorie řízení.* Vydavatelství ČVUT, Praha 2000

[11] V. Mařík, O. Štěpánková, J. Lažanský a kol., *Umělá inteligence.* Academia, Praha 1997

[12] P. Kocourek, *Přenos informace.* Vydavatelství ČVUT, Praha 1999

[13] http://www.gryf.cz

[14] S. Ďaďo, M. Kreidl, *Senzory a měřící obvody.* Vydavatelství ČVUT, Praha 1999

[15] J. Lažanský, *Přednášky z předmětu Operační systémy a jejich aplikace.* http://labe.felk.cvut.cz/ftp/vyuka/33OSA

[16] C. Jarvis, K. Kinsella, P. Timpanaro, *PharLap ETS – An Industrial-Strength RTOS.* http://www.vci.com

[17] L. Reisner, P. Štefka, *DisCO – Jeden software, různý hardware.* http://www.datapartner.cz

[18] L. Reisner, P. Štefka, *SoftPLC - regulátory na bázi systému DisCO.* Automatizace, leden 2002, roč. 45, č. 1, str. 22-25

[19] *AT90S2312 datasheet.* http://www.atmel.com

[20] *L293D datasheet.* http://www.st.com

[21] *LTC485 datasheet.* http://www.linear.com

[22] http://www.microcon.cz

[23] J. Janoušek, V. Suchánek, *Základy silnoproudé elektrotechniky.* Vydavatelství ČVUT, Praha 1997

6. Přílohy

6.1 Základy aerodynamiky a mechaniky letu vrtulníku

6.1.1 Úvod

Problematika řízení vrtulníku je velice složitá a pro její pochopení je dobré seznámit alespoň se základy aerodynamiky a mechaniky vrtulníkového letu. V této kapitole se dozvíme odpovědi na otázky typu „proč vlastně vrtulník létá," „proč je vrtulník bytostně nelineární a nestabilní systém," „jak vrtulník udržuje stabilitu a jak se řídí" apod. Protože se jedná pouze o přehledovou kapitolu, zdržíme se všech složitých matematických vztahů a uvedeme pouze nejzákladnější skutečnosti. Pro lepší pochopení a kvantitativní náhled (jak velká je ta která síla, jak prakticky významná je popisovaná vlastnost…) uvedeme též několik příkladů.

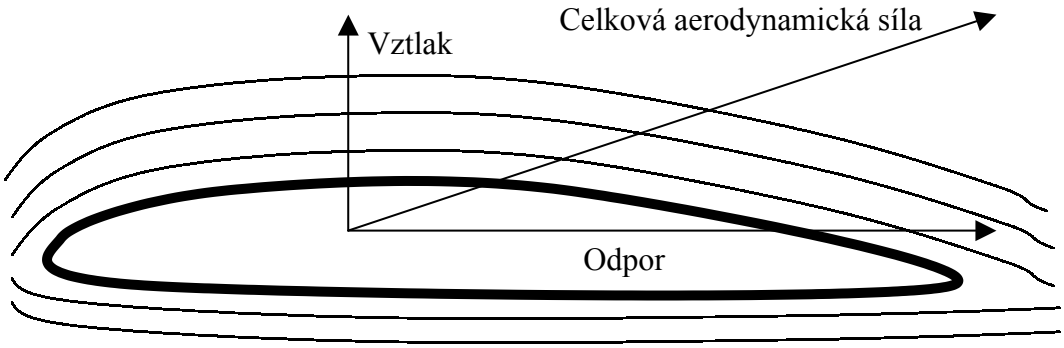
Tato kapitola byla zpracována s využitím poznatků uvedených v [5] a [6].

6.1.2 Základní aerodynamika

6.1.2.1 Základy aerodynamiky pevného křídla

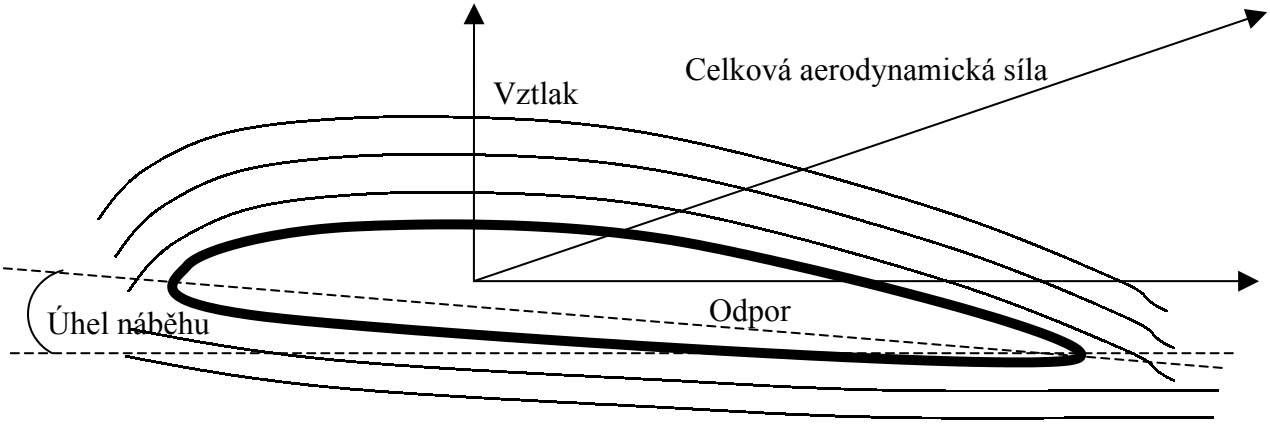
V této části si vysvětlíme základní aerodynamické pojmy. Pro jednoduchost budeme v této části uvažovat obyčejné křídlo běžného letadla, a teprve později budeme zde získané vědomosti aplikovat na rotorový list.

Na začátku si připomeňme, jakým způsobem vlastně generuje vztlak obyčejné křídlo a jaké síly na něj působí. Na každé těleso obtékané proudem vzduchu působí aerodynamická síla. Tuto sílu můžeme v zásadě rozdělit na dvě složky – na aerodynamický **odpor** a **vztlak**. Složku kolmou ke směru proudu nazýváme vztlak, vodorovnou složku pak odpor. Pokud je obtékané těleso vůči směru proudu souměrné, negeneruje žádný vztlak a výsledná aerodynamická síla je tvořena pouze odporem. Je-li těleso nesouměrné, proudí vzduch na jeho spodní a horní straně rozdílnou rychlostí. To vyplývá ze zákona spojitosti proudění – aby se nám na jedné straně tělesa nehromadily částice vzduchu, musí být jejich objem přenesený za jednotku času po obou stranách tělesa stejný. Takže delší stranu tělesa musí vzduch nutně obtékat vyšší rychlostí. Ze známé Bernoulliovy rovnice víme, že čím větší je rychlost tekutiny, tím menší tlak ve směru kolmém k proudu generuje. Proto, když si představíme profil křídla obtékaný proudem vzduchu, vidíme, že na horní straně profilu musí vzduch za jednotku času urazit delší dráhu než na straně spodní. Tím musí po horní straně křídla proudit vyšší rychlostí, čímž ovšem vyvozuje menší tlak. Takže na horní straně křídla nám oproti straně spodní vzniká podtlak, který má snahu křídlo zdvihat vzhůru. Situace je schematicky zobrazena na obrázku 6.1.



Obr. 6.1 – Schematický nákres křídla obtékaného proudem vzduchu

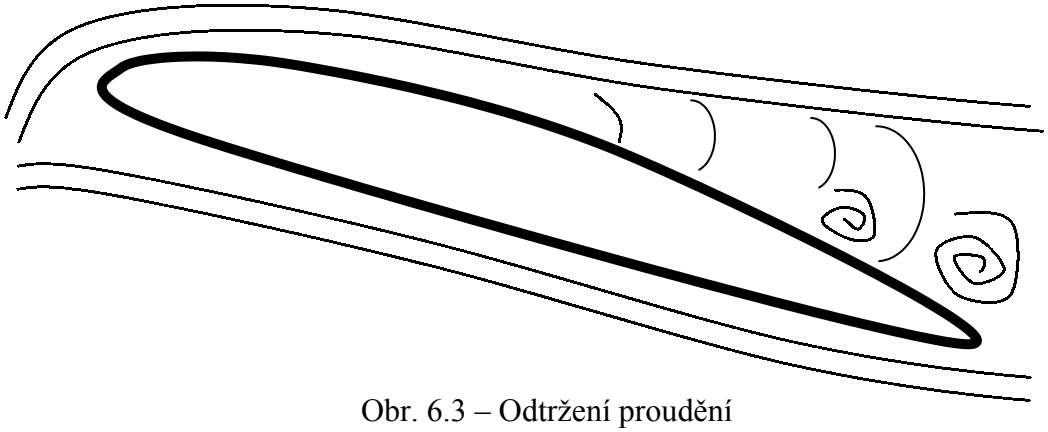
Nyní si představme, jak se celá situace změní, pokud křídlo nebude nastaveno souběžně s proudem vzduchu, ale bude tomuto proudu nastaveno poněkud šikmo. Situaci ilustruje obrázek 6.2.



Obr. 6.2 – Schematický nákres šikmo nastaveného křídla obtékaného proudem vzduchu

Z obrázku 6.2 vidíme, že pokud křídlo nastavíme proudu vzduchu šikmo, zvětšíme tím rozdíl drah částic na horní a spodní straně a tím také zvětšíme generovaný vztlak. Vzroste nám ovšem také aerodynamický odpor, protože křídlo je nyní proudu nastaveno větší plochou. Úhel, o který je křídlo natočeno od pomyslného vodorovného směru, se nazývá **úhel náběhu** (angl. angle of attack).

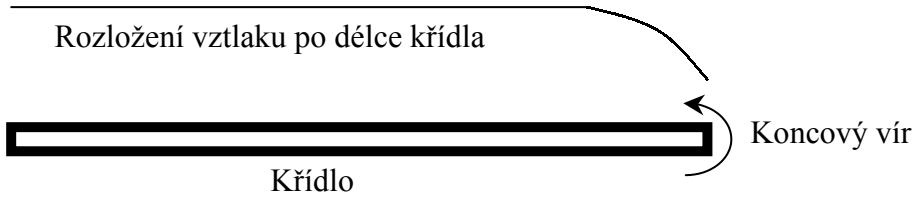
Tento proces funguje tak dlouho, dokud je v rozumných mezích zachována podmínka laminárního obtékání křídla (v praxi proud vzduchu obtékající křídlo samozřejmě nikdy není zcela laminární; zvláště na odtokové hraně křídla často přechází v proudění turbulentní). Pokud je úhel náběhu příliš velký, částice vzduchu na odtokové straně profilu už nesledují laminárně profil křídla, ale přechází do turbulentního proudění. Tento jev se nazývá **odtržení proudění** (angl. stall). Úhel náběhu, při kterém začíná k odtrhávání prudění docházet, se nazývá **kritický úhel náběhu** (angl. critical angle of attack). Odtržení proudění je znázorněno na obrázku 6.3.



Obr. 6.3 – Odtržení proudění

Prozatím jsme hovořili pouze o profilu křídla a o vlivu jeho tvaru a nastavení na generovaný vztlak a odpor. Rozeberme si nyní chování křídla jako celku. Pokud by křídlo bylo nekonečně dlouhé, byl by vztlak jím generovaný po délce rozložen stejnoměrně. Ve skutečnosti má ale křídlo délku (rozpětí) vždy konečnou. Protože na horní straně křídla je menší tlak vzduchu než na straně spodní, je vzduch na koncích křídla nasáván ze spoda nahoru a tím vzniká vír, který samozřejmě účinnost křídla snižuje. Situace je znázorněna na obrázku 6.4.

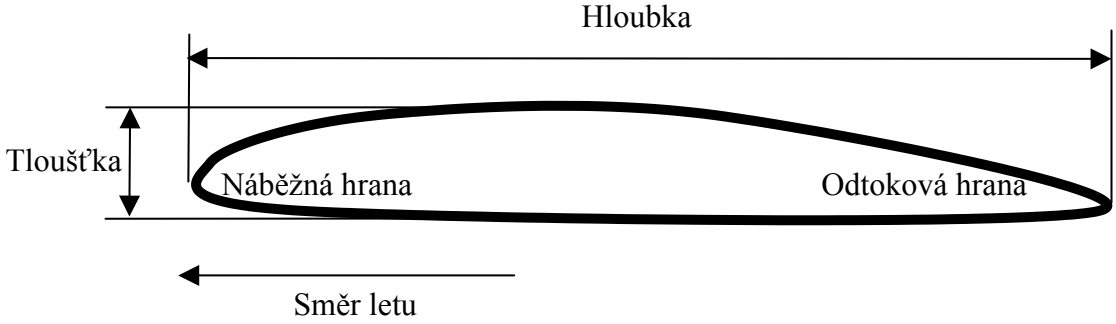
Poznámka autora: K minimalizaci tohoto škodlivého jevu se na konce křídla často montují ***koncová vřetena*** *nebo různě nastavená pomocná křidélka, tzv.* ***winglety****. Jejich detailnější popis ovšem přesahuje rámec této práce.*



Obr 6.4 – Průběh vztlaku po délce křídla

6.1.2.2 Základní charakteristiky profilu

V následujících několika odstavcích se seznámíme s hlavními charakteristikami profilu křídla. Jeho základní rozměry se nazývají **hloubka** a **tloušťka** profilu. Hloubka je délka profilu při bočním pohledu, tloušťka je největší výška profilu. **Náběžná hrana** je ta hrana profilu, která je při normálním směru letu vpředu, tj. rozráží vzduch. **Odtoková hrana** je opačná hrana, kterou vzduch křídlo opouští. Všechny zavedené pojmy jsou ukázány na obrázku 6.5.



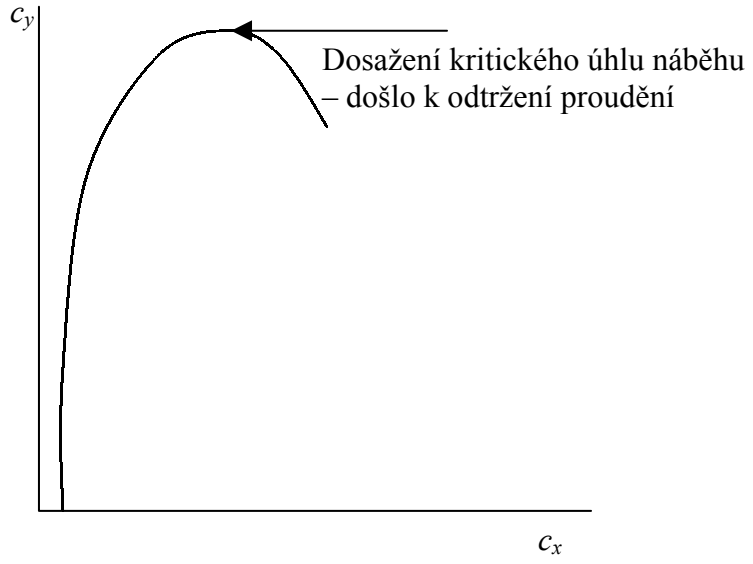
Obr. 6.5 – Hlavní rozměry profilu

Jak již bylo uvedeno v předchozím textu, tvar profilu určuje jeho hlavní aerodynamické vlastnosti, tj. jaký vztlak je schopen generovat a jaký odpor má při dané rychlosti. Číselně tyto dvě vlastnosti profilu určují tzv. **součinitel vztlaku** c_y a **součinitel odporu** c_x. Protože generovaný vztlak a odpor profilu jsou závislé na úhlu náběhu (viz. část 6.1.2.1), jsou na úhlu náběhu samozřejmě závislé i oba součinitele, a tudíž se jedná o závislé veličiny. Jako hlavní charakteristika profilu se proto udává tzv. **polára,** což je graf závislosti c_y na c_x. Příklad průběhu poláry je na obrázku 6.6.

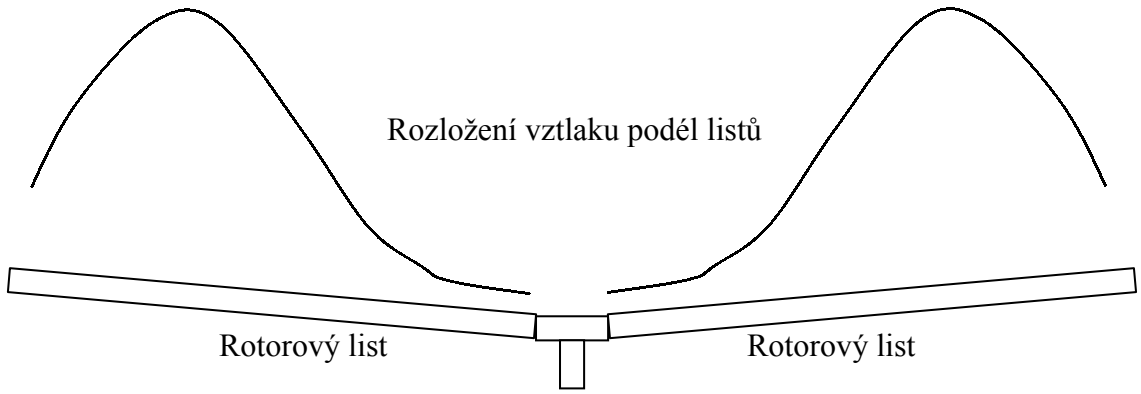
6.1.2.3 Aerodynamika rotorového listu

Rotorový list (angl. rotor blade) si můžeme představit jako křídlo, které rotuje kolem osy umístěné na jednom svém konci. Podstatným rozdílem oproti pevnému křídlu je fakt, že rychlost pohybu profilu ve vzduchu je závislá na vzdálenosti od středu otáčení. Z tohoto důvodu je samozřejmě rozdílný také vztlak, generovaný jednotlivými elementy listu. Vztlak generovaný poblíž osy otáčení je velmi malý, neboť tato část se pohybuje nejpomaleji a její účinnost je ještě snižována koncovým vírem. Se vzdáleností od středu otáčení vztlak roste, ovšem poblíž konce listu začne zase klesat, protože účinnost listu je zde podobně jako poblíž

středu znehodnocena koncovým vírem. Průběh vztlaku po délce rotorového listu ukazuje obrázek 6.7.



Obr. 6.6 – Příklad průběhu poláry profilu



Obr. 6.7 – Rozložení vztlaku podél rotorových listů

6.1.3 Dynamika vrtulníku

6.1.3.1 Základní síly působící na vrtulník

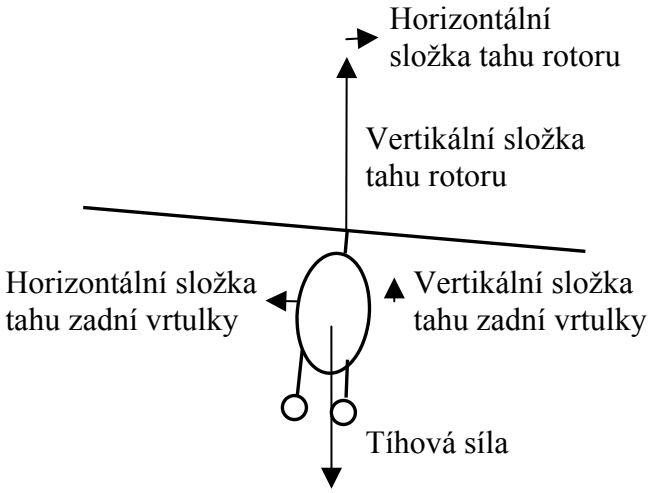
Zamysleme se nyní nad základními silami, působícími na vrtulník. Pro jednoduchost předpokládejme, že se vrtulník nachází ve **visu** (angl. hover), tj. v prostoru se nepohybuje žádným směrem. Na vrtulník, jako na každé hmotné těleso, působí samozřejmě tíhová síla. Ta má své působiště v **těžišti** vrtulníku (angl. centre of gravity) a směřuje kolmo k zemi. Tuto sílu vyrovnává tahová síla rotoru (angl. lift nebo thrust). Na obrázku 6.8, kde jsou síly působící na vrtulník schematicky znázorněny, je vrtulník záměrně nakreslen v náklonu a tahová síla rotoru tak působí poněkud „šikmo," tj. není rovnoběžná se silou tíhovou. Proč tomu tak je, zjistíme později.

Předpokládejme pro začátek, že vrtulník je vzhledem k zemi orientován „rovně," tj. osa otáčení rotoru je kolmá na zem a tahová síla rotoru je rovnoběžná se silou tíhovou. Uvidíme, kam nás tato úvaha dovede. Protože v tomto případě se obě síly jednoduše vyruší, zůstává ve vertikálním směru vrtulník v klidu (tj. nestoupá ani neklesá). Silový moment

otáčejícího se rotoru ovšem vyvolává stejně velký reakční moment na trup (ale v opačném směru), takže vrtulník má snahu otáčet se kolem osy rotoru (uvědomme si, že pro pravotočivý rotor, tj. rotor otáčející se v záporném směru, má trup snahu roztočit se doleva, tj. v kladném směru!). Tento moment kompenzuje zadní vrtulka (**ocasní rotor**, angl. tail rotor), která přes páku tvořenou trupem vrtulníku vyvozuje vzhledem k ose otáčení hlavního rotoru stejně velký kompenzační moment. Na první pohled by se mohlo zdát, že v tuto chvíli už jsou všechny síly a momenty vyrovnány a vrtulník setrvává v klidu, ovšem není tomu tak. Nesmíme totiž zapomínat na boční tah zadní vrtulky, který má snahu táhnout vrtulník „do strany".

Tento tah musíme samozřejmě také nějak kompenzovat. Protože pro to nemáme jiný akční člen, nezbývá nám než použít hlavní rotor. Proto celý vrtulník nakloníme podle podélné osy tak, aby se tah hlavního rotoru rozdělil na vertikální složku, která kompenzuje tíhovou sílu, a složku horizontální, která vyrovnává boční tah zadní vrtulky. Tím se ovšem vychýlí osa zadní vrtulky z horizontální roviny, a tahová síla zadní vrtulky se tak také rozdělí na vertikální a horizontální složku. Vertikální složka tohoto tahu má snahu naklánět vrtulník „nosem dolů," tj. klopit ho kolem příčné osy. Tuto sílu musíme vyrovnávat **předozadní cyklikou** (angl. back cyclic) hlavního rotoru (o cyklickém řízení rotorových listů pojednává oddíl 6.1.5.1).

Poznámka autora: V praxi se boční náklon vrtulníku (odchylka osy hlavního rotoru od kolmého směru) pohybuje v řádu jednotek úhlových stupňů, typicky bývá do 5°.



Obr. 6.8 – Hlavní síly působící na vrtulník ve visu

6.1.3.2 Základní dynamika rotorových listů

V této podkapitole rozebereme základní síly působící na rotorové listy. Pro jednoduchost budeme opět předpokládat, že vrtulník je ve visu, tj. nepohybuje se v prostoru žádným směrem. Chování rotorových listů za letu je mnohem složitější a rozebereme ho v následujících kapitolách.

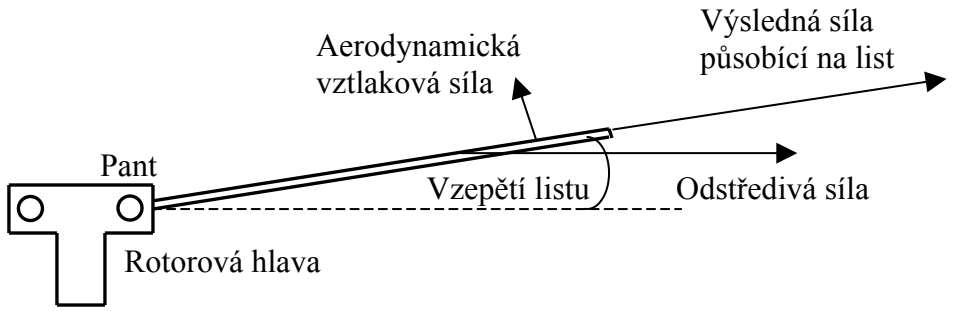
Na rotorové listy ve visu působí dvě hlavní síly – **síla odstředivá**, způsobená otáčením rotoru (angl. centrifugal force), a **aerodynamická vztlaková síla** (angl. lift). Vzhledem k relativně nízké hmotnosti rotorových listů lze proti těmto dvěma silám zanedbat sílu tíhovou, jelikož je řádově menší. Rozložení vztlaku podél rotorového listu není stejné, ale roste směrem k vnějším oblastem, protože ty se oproti vnitřním pohybují vyšší rychlostí a tudíž ho generují více než vnitřní. Na vnějších okrajích listů ovšem vztlak zase klesá, protože účinnost profilu je zde snižována vírem (o principu generování vztlaku rotorovými listy pojednává část 6.1.2.3). Proto působiště aerodynamické vztlakové síly (angl. centre of lift) je posunuto směrem k vnějšku listu. Podobná úvaha platí také pro působiště odstředivé síly

(angl. centre of gyration). Jeho poloha je určena především rozložením hmotnosti po délce listu. Hmota dále od osy otáčení se pohybuje rychleji, a proto má větší vliv než hmota umístěná blízko osy. Pokud předpokládáme homogenní rozložení hmoty, nachází se působiště odstředivé síly asi v 58% délky listu směrem od osy otáčení. Jeho polohu lze měnit dovažováním rotorového listu.

Aerodynamická vztlaková síla působí ve směru kolmém na rotorový list, a má tak snahu ohýbat ho směrem vzhůru a do středu otáčení (viz. obrázek 6.9.). Pokud by nebyla kompenzována odstředivou silou a mechanika rotoru by to umožnila, listy by byly přiraženy k sobě a postaveny kolmo vzhůru podél osy otáčení. (Rotorové listy jsou k hlavici rotoru připevněny pomocí pantů, ve kterých se v jistých mezích mohou otáčet, viz. obr. 6.9).

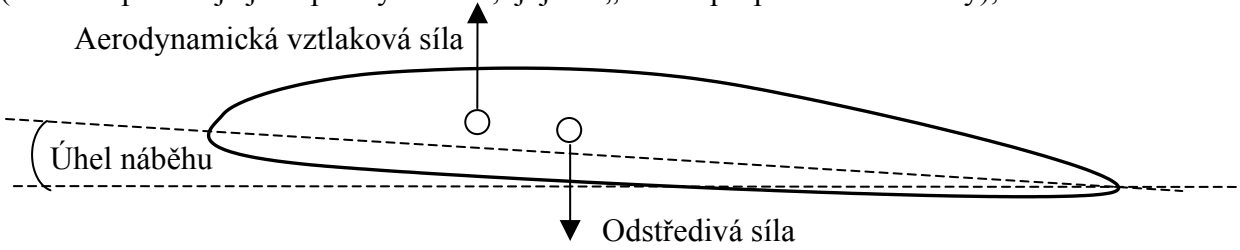
Odstředivá síla působí ve směru kolmém na osu otáčení, a má tak snahu (respektive její část) pokládat rotorové listy do roviny kolmé na osu otáčení. Složením antagonistických složek obou sil zjistíme, že rotorové listu jsou postaveny do V. Toto „V“ je poměrně široce rozevřené, protože odstředivá síla bývá řádově větší než aerodynamická vztlaková síla.

Poznámka autora: Pro lepší představu o velikosti obou sil si uveďme příklad z praxe. Na rotorový list o délce 76cm a hmotnosti 100g při 1700ot./min působí odstředivá síla o velikosti asi 100kg! Naprotitomu aerodynamická vztlaková síla generovaná tímto listem je typicky kolem 1,5kg. Proto ***vzepětí listu*** *(úhel, který rotorový list svírá s rovinou kolmou na osu otáčení, angl. coning angle) je asi 0,8º. Z tohoto příkladu také vidíme, jak obrovské síly působí na rotorovou hlavu a závěsy listů, a jak pevné tyto části musí být.*



Obr. 6.9 – Hlavní síly působící na rotorový list

V předchozích odstavcích jsme analyzovali, kde budou umístěna působiště aerodynamické vztlakové síly a odstředivé síly na rotorovém listu v podélném smyslu, a jakým způsobem se jejich působení na list projeví. Velmi důležité je ovšem také umístění působišť těchto sil v příčném řezu. Ukažme si to na následujícím příkladu. Pokud budeme uvažovat homogenní list, bude působiště odstředivé síly v příčném směru splývat s příčnou polohou těžiště. Naprotitomu působiště aerodynamické vztlakové síly bude v místě největšího rozdílu tlaků na spodní a vrchní straně listu, tj. v místě, kde končí náběžná část profilu. Pro typický list bude působiště aerodynamické vztlakové síly asi v 25% hloubky profilu, a při uvažování homogenního listu bude působiště odstředivé síly asi v 35% hloubky profilu (hloubka profilu je jeho příčný rozměr, tj. jeho „délka“ při pohledu ze strany), viz. obr. 6.10.

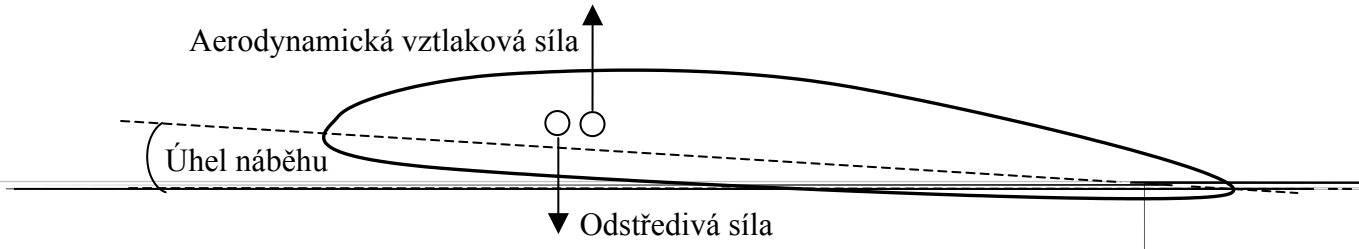


Obr. 6.10 – Umístění působišť sil v příčném řezu

Z obrázku 6.10 je patrné, že obě síly spolu tvoří dvojici, která má snahu rotorový list příčně kroutit. Vidíme, že při tomto rozložení sil zkrut listu zvětšuje jeho úhel náběhu. Zvětšením úhlu náběhu profilu se ale také zvýší jeho součinitel vztlaku (viz sekce 6.1.2.1), tím se zvýší aerodynamická vztlaková síla, tím zkrut a zase úhel náběhu… Máme uzavřenu klasickou kladnou zpětnou vazbu. Inženýrská intuice nám napoví, že tento proces nemůže probíhat do nekonečna. Probíhá až do dosažení kritického úhlu náběhu, při kterém dojde k odtržení proudění. Tím se rychle sníží aerodynamická vztlaková síla a pružnost vrátí rotorový list prudce zpět. Dojde tak ovšem ke snížení úhlu náběhu, a celý proces se periodicky opakuje. Tento jev se nazývá **třepetání** listu (angl. flutter), a patří mezi vůbec nejnebezpečnější jevy v letectví. Třepetáním rotorových listů se vrtulník nejen stává velmi nestabilním, ale ponenáhlu zpravidla dojde k jejich destrukci vinou nadměrného mechanického namáhání. Pád vrtulníku je potom neodvratný.

Poznámka autora: K třepetání nosných a ocasních ploch dochází i u letounů s pevným křídlem, kde bývá způsobeno mechanickými rezonancemi konstrukce křídla, buzenými například obtékáním křídla vzduchem či vibracemi motoru apod. Tento dlouho neznámý jev způsobil celou řadu leteckých katastrof, a byl také jedním s hlavních problémů prvních nadzvukových letadel, kde docházelo hlavně k problémům s třepetáním výškového kormidla, buzeným turbulencemi za křídlem. Tento problém se podařilo vyřešit až zavedením plovoucích ocasních ploch.

Problémů s třepetáním se zbavíme jedině tak, že nějakým způsobem posuneme těžiště odstředivé síly před působiště síly vztlakové. Tím vlastně zavedeme stabilizující zápornou zpětnou vazbu; Zvětšená aerodynamická vztlaková síla bude mít tentokrát snahu úhel náběhu zmenšovat (viz. obr. 6.11), a tím se automaticky sama reguluje.



Obr. 6.11 – Umístění působišť sil v příčném řezu

Tím způsobem, kterým můžeme působiště odstředivé síly posunout, je dovážení listu na náběžné hraně. Zvýšením hmotnosti v náběžné části listu můžeme tedy stabilizovat nebezpečné třepetání. S působištěm aerodynamické vztlakové síly dost dobře hýbat nelze, protože jeho poloha je dána profilem listu.

Poznámka autora: V modelářské praxi se dovažování rotorových listů (v případě velmi obvyklých dřevěných listů) řeší tak, že náběžná část listu se vyrobí z tvrdého a těžkého dřeva (např. smrk) a odtoková část z lehkého, zpravidla z balsy. Tím se tak říkajíc zabijí dvě mouchy jednou ranou, protože list se touto konstrukcí také patřičně zpevní.

6.1.4 Aerodynamika vrtulníku

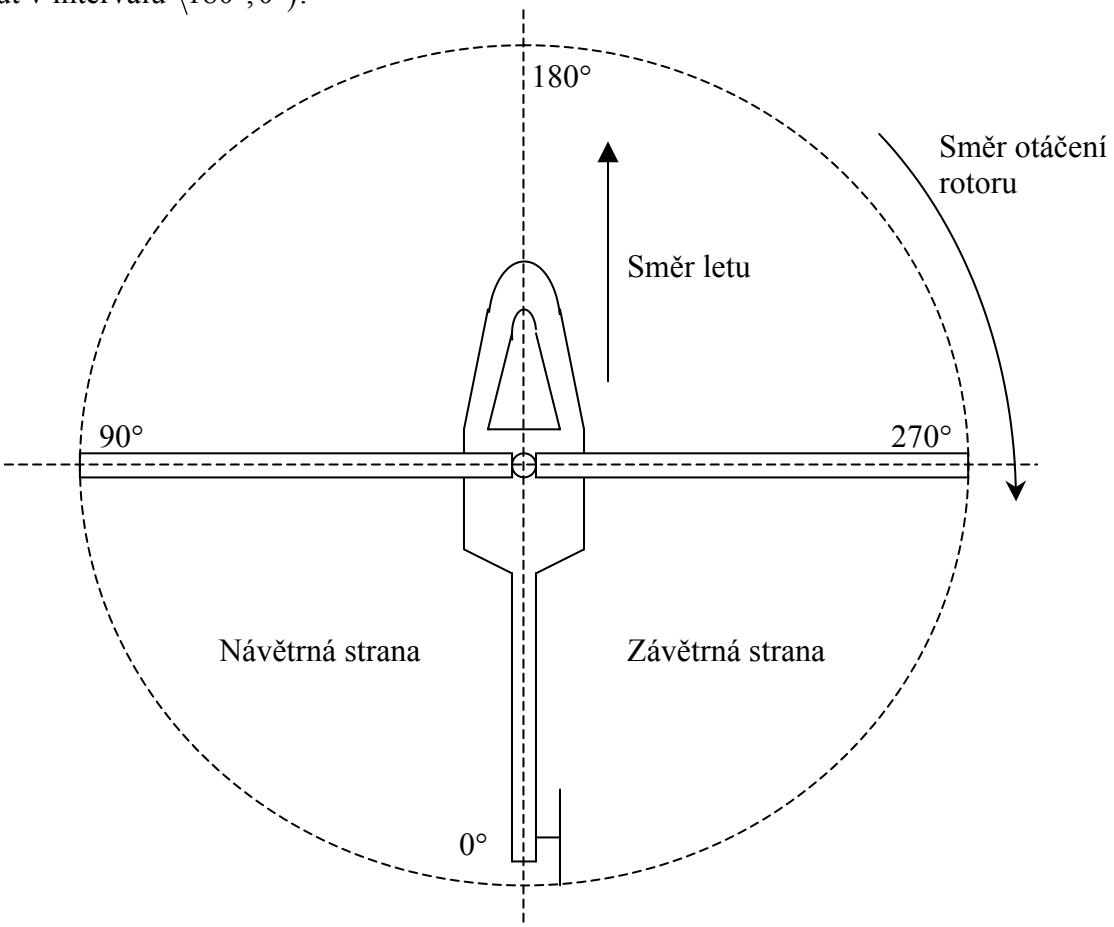
6.1.4.1 Aerodynamika rotorových listů v dopředném letu

Až doposud jsme ve všech našich rozborech uvažovali vrtulník ve visu, tj. „stojící“ ve vzduchu na místě. Nyní se zamyslíme na tím, jak se celá situace změní, pokud vrtulník přejde z visu do dopředného letu. Pro začátek probereme chování rotorových listů.

Dříve než se pustíme do dalšího výkladu, zavedeme si několik nových pojmů. **Azimut** (angl. azimuth) nám bude sloužit k popisu úhlové polohy rotorových listů. Pokud si představíme, že na vrtulník hledíme shora, bude mít rotorový list azimut 0°, pokud bude mířit podél trupu vzad. Azimut 90° bude mít list tehdy, bude-li orientován kolmo k trupu na levé straně. Bude-li rovnoběžný s osou trupu a v předu, bude jeho azimut 180° atd., viz. obr. 6.12.

Návětrná strana (angl. advancing side) vrtulníku je ta strana, kde se list pohybuje proti směru proudu vzduchu, vyvolaném dopředným letem vrtulníku. Tj. pro pravoběžný rotor (rotor otáčející se v záporném směru) bude návětrná strana levá strana vrtulníku (při pohledu shora). Rotorový list se bude nacházet na návětrné straně, bude-li jeho azimut v intervalu $\langle 0°, 180° \rangle$.

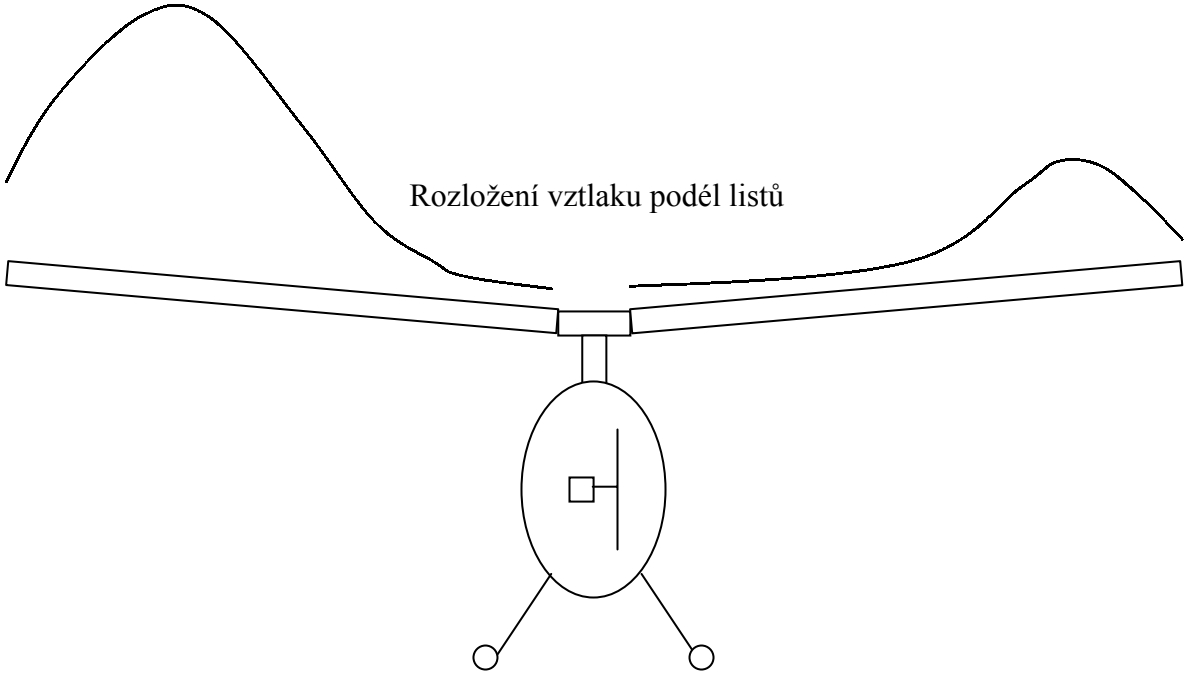
Analogicky **závětrná strana** (angl. retreating side) bude ta strana vrtulníku, kde se bude list pohybovat po směru proudu vzduchu. Pro pravoběžný rotor se tedy bude jednat o pravou stranu vrtulníku. List se bude nacházet na závětrné straně tehdy, pokud bude jeho azimut v intervalu $\langle 180°, 0° \rangle$.



Obr. 6.12 – Definice azimutu a dalších pojmů

Pokud by se vrtulník ve vzduchu pohyboval vpřed a rotorové listy by byly pevně připevněny k rotorové hlavě, rovnováha vztlaku generovaného oběma rotorovými listy by byla porušena. Jinými slovy – každý list by generoval jiný vztlak než ten druhý (uvažujeme pouze dvoulistý rotor). Je to způsobeno tím, že list na návětrné straně vrtulníku má vyšší vzdušnou rychlost (je obtékán rychleji), protože jeho skutečná rychlost se ještě vektorově sčítá s rychlostí vrtulníku. Generuje tedy větší vztlak než list na závětrné straně, jehož vzdušná rychlost je naopak o rychlost vrtulníku snižována. Může se dokonce stát, že při dostatečně rychlém letu je vnitřní část listu na závětrné straně obtékána v opačném směru než

by měla být (od odtokové hrany k náběžné). Přibližné rozložení vztlaku na obou listech při dopředném letu je znázorněno na obrázku 6.13 (vrtulník letí směrem „od nás“).



Obr. 6.13 – Rozložení vztlaku podél rotorových listů při dopředném letu

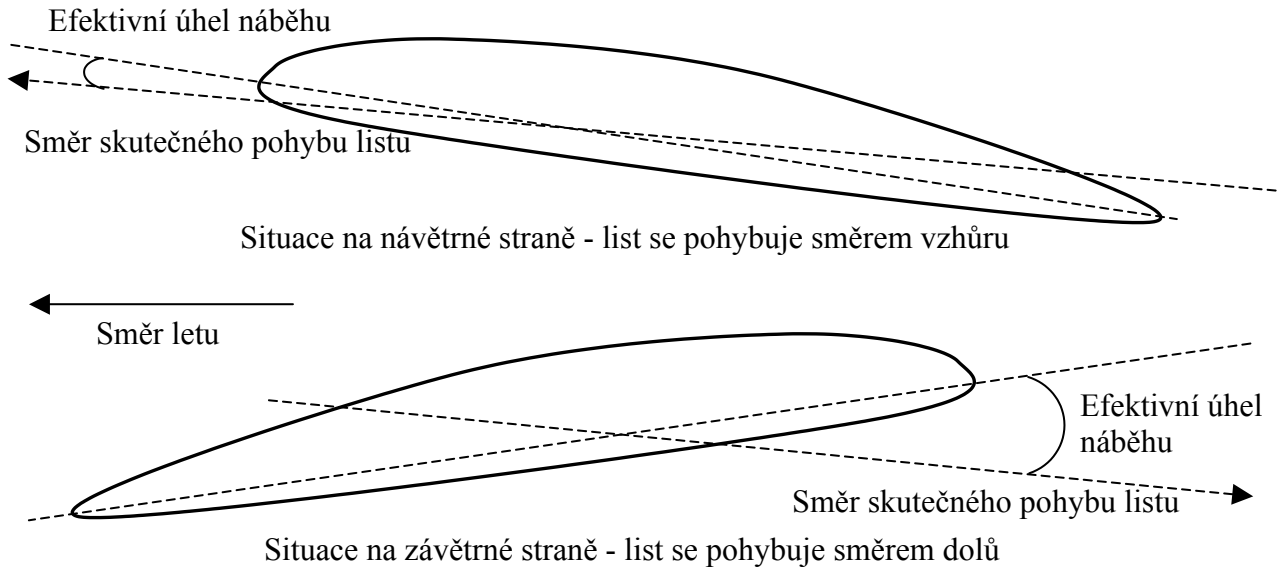
Je zřejmé, že tato nerovnováha vyvolává silový moment, který má snahu vrtulník otáčet. Ve skutečnosti by díky gyroskopickému efektu tento moment zdvihal nos vrtulníku, tj. klopil kolem příčné osy vzad (pozor, kdykoli uvažujeme síly působící na rotor, nesmíme zapomínat na gyroskopický efekt způsobený jeho otáčením! Ten způsobuje, že síla začíná efektivně působit až zhruba o čtvrt otáčky dál než dosahuje svého maxima. Proto, ačkoli maximální vztlak list generuje když má azimut 90°, maxima silového působení na trup je dosaženo až o čtvrtotáčku dál, tedy v azimutu 180°).

Řešení tohoto problému se skrývá v pohyblivém uchycení rotorových listů k rotorové hlavě (například pomocí pantů), které jim umožňuje pohybovat se nahoru a dolů. Jedná se o stejný systém, který listům umožňuje zaujmout polohu do V ve visu (viz. článek 6.1.3.2).

Zvýšený vztlak, generovaný rotorovým listem na návětrné straně, zvětší jeho vzepětí. Naopak na závětrné straně, kde je vztlak menší, bude menší také vzepětí listu. Tímto způsobem se budou oba listy pohybovat periodicky nahoru a dolů (jejich vzepětí se bude periodicky zvětšovat a zmenšovat) v závislosti na tom, v jakém azimutu se budou právě nacházet (pozor – nezapomínejme na gyroskopický efekt! Vlivem fázového posunu způsobeného otáčením rotoru bude list dosahovat maximálního vzepětí až čtvrtotáčku po amplitudě vztlaku, tedy v azimutu 180°. Nejmenší vzepětí bude mít naopak v azimutu 0°).

Podívejme se nyní, jaký bude mít tento pohyb vliv na vztlak generovaný oběma listy. Nezapomínejme, že vlivem periodického stoupání a klesání se listy už nepohybují v rovině kolmé na osu otáčení rotoru! Proto vlivem stoupání listu na návětrné straně se bude zmenšovat jeho efektivní úhel náběhu, a tím i jím generovaný vztlak. Na závětrné straně se bude efektivní úhel náběhu naopak zvětšovat, a tím také vztlak generovaný tímto listem. Tím se vztlaky generované oběma listy zase vyrovnají. Celý princip zvětšování, respektive zmenšování úhlu náběhu při klesání, respektive stoupání rotorového listu je lépe zřejmý z obrázku 6.14. Tento mechanismus byl objeven už ve dvacátých letech minulého (20.) století

donem Juanem de la Ciervou při jeho výzkumu vírníků (angl. autogyro. Vírník je v principu obyčejné letadlo, které má místo křídla rotor, roztáčený nikoli motorem, ale náporem vzduchu za letu. Jedná se vlastně o přímého vývojového předchůdce vrtulníku.).



Obr. 6.14 – Princip změny efektivního úhlu náběhu listů vlivem periodických změn vzepětí

Poznámka autora: Na tomto místě si dovolím menší historickou odbočku. Čtenář který se nezajímá o historii ji možná shledá poněkud nudnou, a proto mu vřele doporučuji ji přeskočit.

Juan de la Cierva byl španělský šlechtic, který pro vývoj vrtulníků vykonal totéž, co Luis Blériot či bratři Wrightové pro letadla s pevným křídlem. Narodil se roku 1895, a již jako šestnáctiletý! sestrojil první funkční španělské letadlo těžší než vzduch. Po první světové válce založil továrnu na výrobu letadel. Její první stroj byl těžký třímotorový letoun a jeho zalétávacím pilotem byl velitel madridského letiště. Velitel byl možná schopný úředník, ale nezkušený pilot, a proto první let tohoto stroje skončil jak musel – katastrofou. Hned po stratu letadlo po přetažení ztratilo rychlost a zřítilo se k zemi. Pilot naštěstí vyvázl nezraněn, ale letoun byl zcela zničen. Tato příhoda přivedla Juana na myšlenku sestrojit letadlo, kterému by ztráta rychlosti nevadila. Jako pětadvacetiletý už zalétává svůj první úspěšný vírník – typ c4 (ačkoli byl označen c4, byl to již jeho 16. vírník v pořadí – Juan totiž od každého typu postavil několik kusů s různými obměnami). V roce 1925 se stěhuje do Anglie a zakládá zde firmu na výrobu vírníků. Ve 20. a 30. letech vyrobila tato firma několik set dobře létajících vírníků, které dodala do celého světa. Jednalo se o velmi snadno ovladatelné a spolehlivé stroje – až do začátku druhé světové války došlo pouze k jediné smrtelné nehodě vírníku! Po roce 1945 už jsou vírníky nahrazovány prvními vrtulníky. Toho už se ale Juan de la Cierva bohužel nedožil. Ironií osudu, poté, co přežil všechny lety svých pokusných strojů, zahynul jako cestující při nehodě obyčejného dopravního letadla v roce 1936.

K předchozímu výkladu je nutné podotknout, že skutečnou situaci v zájmu lepšího pochopení značně zjednodušuje. Ve skutečnosti je problematika periodického pohybu rotorových listů (angl. flapping) mnohem složitější. V předchozím textu například uvažujeme ideální stav, kdy maximálního vzepětí list dosahuje přesně v azimutu 180° a minimálního v azimutu 0°. To si můžeme představit také tak, že celá rovina otáčení rotoru se jakoby nakloní o několik stupňů vzad. To sice způsobí snahu vrtulníku zvedat nos (klopení vrtulníku, angl. pitch), ale tento efekt můžeme snadno eliminovat předozadní cyklikou (princip

cyklického řízení rotorových listů bude objasněn až v článku 6.1.5.1). Ve skutečnosti bude ovšem amplituda vzepětí dosahována poněkud dále, přibližně v azimutu 200°. Nejnižší bod se naopak posune na azimut přibližně 20°. To je způsobeno tím, že vzepětí listu, jak již bylo řečeno, ovlivňuje úhel náběhu a tím i generovaný vztlak. Pokud bude list v azimutu 180°, bude nastaven podélně na proud vzduchu způsobený dopředným letem, ale díky velkému vzepětí bude generovat větší vztlak než v azimutu 0°, kdy je také tomuto proudu nastaven podélně, ale má vzepětí menší. Tato nerovnováha způsobí výše zmíněný fázový posun amplitud. Nyní ovšem není rovina otáčení rotoru nakloněna pouze vzad, ale také do strany. To způsobuje, že vrtulník má také snahu naklánět se kolem podélné osy (tento typ náklonu se nazývá klonění, angl. roll). Problém je v tom, že tato snaha se projevuje jen při dopředném letu. Takže když máme vrtulník dokonale seřízený a stabilní ve visu, má při dopředném letu snahu bočně se naklánět (klonit).

Dalším problémem je fakt, že jsme uvažovali listy, které se mohou pohybovat zcela bez omezení. To ovšem v praxi nebývá možné. Pokud bychom totiž dovolili listům pohybovat se v pantech úplně volně, byl by vrtulník velmi nestabilní, protože mezi listy a trupem by nebyla dostatečně pevná vazba. Proto bývá pohyb listů v pantech nejen omezen co do maximálního úhlu vzepětí, ale také tlumen gumovými tlumiči. Tyto tlumiče a omezení amplitudy mají snahu zvýšit frekvenci kmitání rotorových listů (v ideálním případě, jak vyplývá z předchozího textu, je tato frekvence stejná jako frekvence otáčení rotoru). To způsobuje, že rotorový list dosahuje maximálního vzepětí naopak dříve než v ideálním případě, např. už v azimutu 160°. Tento efekt působí boční klonění, ovšem v opačném směru, než posun popsaný v předchozím odstavci.

Pokud máme štěstí, oba výše popsané efekty se vyruší. Mnohdy ale způsobují nestabilitu vrtulníku při dopředném letu, ačkoli pro vis je vrtulník seřízen dobře. Tento problém se pak většinou řeší nastavováním výše popsaných tlumičů.

6.1.5 Mechanika letu vrtulníku

6.1.5.1 Kolektivní a cyklické řízení rotorových listů

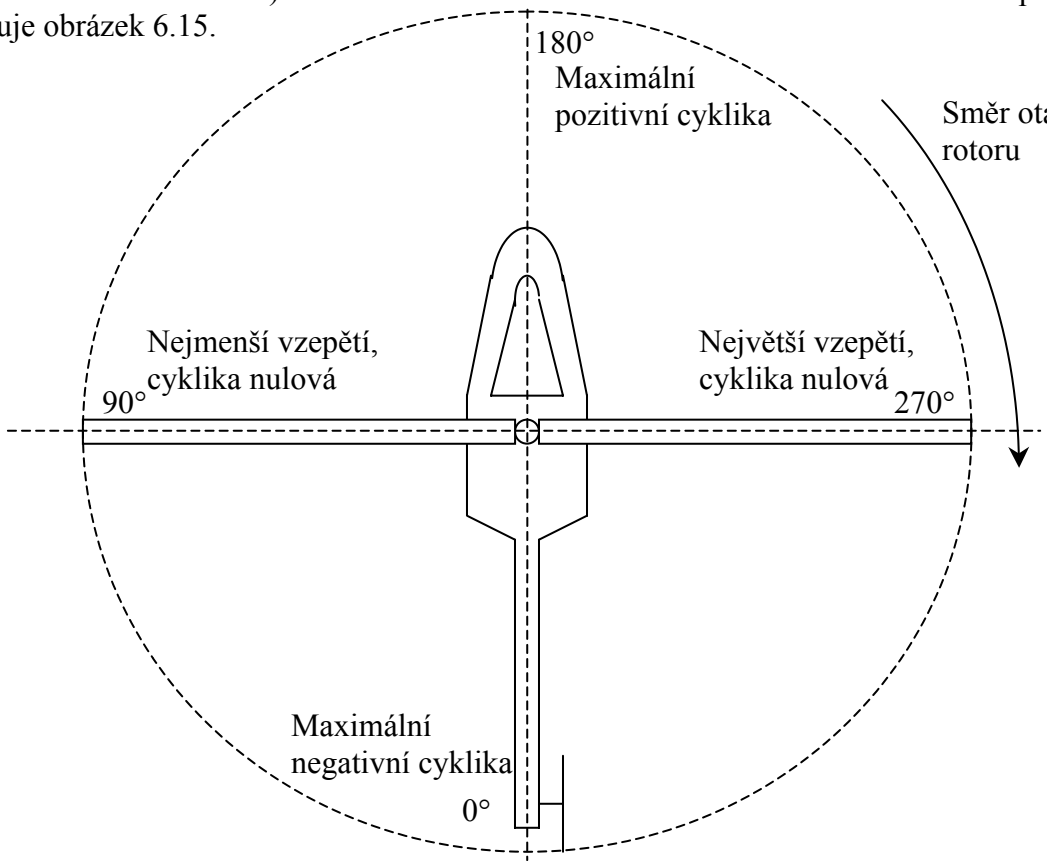
V této podkapitole si popíšeme princip řízení rotorových listů. V zásadě můžeme toto řízení rozdělit do dvou víceméně nezávislých částí – na **kolektivní** a **cyklické** řízení (angl. collective, resp. cyclic control).

Pomocí kolektivního řízení ovládáme velikost a směr tahové síly generované rotorem. Protože rotor má z důvodů popsaných kapitole 6.1.5.4 konstantní otáčky, měníme velikost a směr jeho tahu řízením úhlu náběhu rotorových listů (Rotorové listy jsou za tímto účelem upevněny k rotorové hlavě tak, aby bylo možno jejich úhel náběhu řídit – kromě toho, že vlastní listy jsou uchyceny v pantech, jsou tyto panty k rotorové hlavě připojeny otočnými klouby, které umožňují natáčet rotorové listy kolem jejich podélné osy. Tyto klouby se anglicky nazývají feathering hinges či feathering shafts. Úhel náběhu listu je potom řízen servomotorem.). Změnou úhlu náběhu lisů regulujeme velikost jimi generovaného vztlaku. Pokud je nastavíme do záporného úhlu náběhu, bude rotor generovat tah opačným směrem (bude vrtulník „tlačit do země," pokud bude v normální poloze. Opačný tah má význam pro let na zádech a různé akrobatické obraty.). Jak již název napovídá, řídíme tímto způsobem oba rotorové list stejně (kolektivně). Princip kolektivního řízení je zřejmý, a proto ho nebudeme dále rozebírat.

Cyklické řízení je poněkud komplikovanější. Jeho účelem je naklánět vrtulník jak kolem příčné (předozadní cyklika), tak kolem podélné osy (boční cyklika). Je založeno na periodických změnách úhlu náběhu listů v závislosti na jejich fázi (fází listu rozumíme jeho

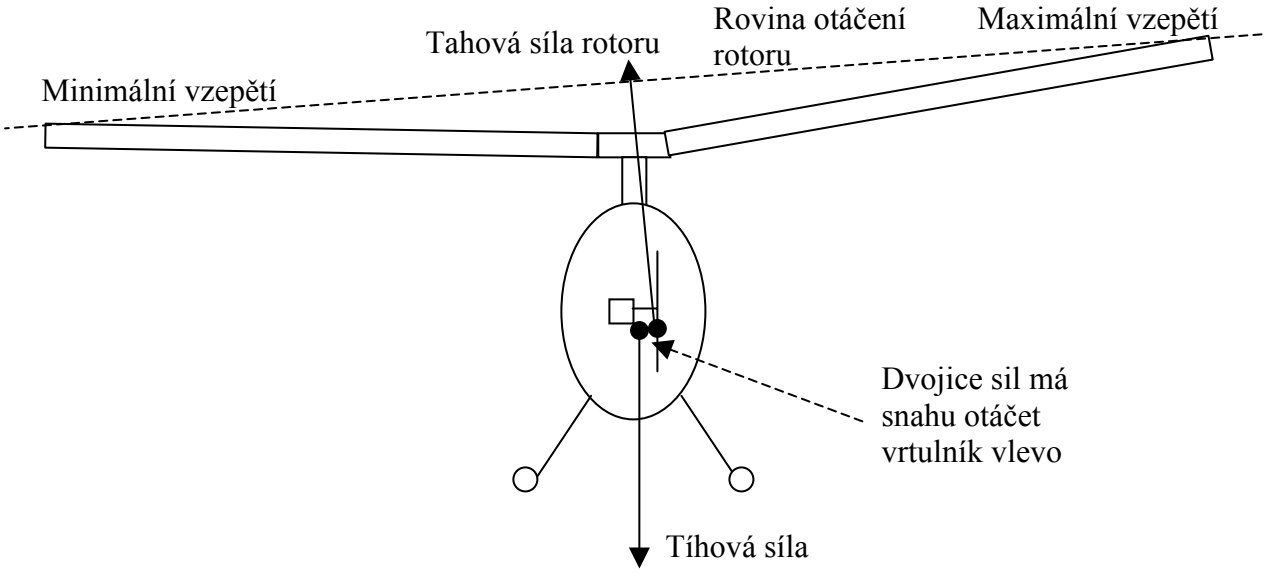
polohu, tj. azimut) . Jak bylo ukázáno v části 6.1.4.1, úhel náběhu a vzepětí listu jsou do jisté míry závislé veličiny a jejich periodické změny mohou naklánět vrtulník kolem podélné i příčné osy. Toho se využívá pro řízení náklonu v obou směrech – jednoduše ve „vhodném“ azimutu nastavíme list na větší úhel náběhu a tento úhel náběhu budeme proporcionálně zmenšovat v závislosti na jeho fázi, až o půlotáčku dál dosáhneme minima. Od tohoto minima začneme úhel náběhu zase postupně zvětšovat, až po dokončení celé periody (otáčky) se list dostane zase do stejného azimutu kde jsme začali a bude mít zase maximální úhel náběhu. Takto budeme plynule periodicky měnit úhly náběhu obou listů. Tímto způsobem periodicky měníme velikost vztlaku generovaného listem podle jeho fáze, a tím vyvoláváme náklon vrtulníku žádaným směrem. Ze zřejmých důvodů se toto nazývá cyklickým řízením rotorových listů.

Nejlépe si princip cyklického řízení ukážeme na příkladu. Pro lepší pochopení příklad opět poněkud zidealizujeme a pomineme některé jeho složitější aspekty. Řekněme, že chceme vrtulník naklonit podél podélné osy doleva (levé klonění, angl. left roll). Toho dosáhneme tak, že v azimutu 180° nastavíme list na maximální úhel náběhu (největší pozitivní cyklický akční zásah, zkráceně největší pozitivní cyklika – angl. maximum positive cyclic). Tento úhel náběhu budeme postupně zmenšovat, jak se bude list otáčet. V azimutu 270° bude náš akční zásah nulový a list bude mít úhel náběhu určený pouze kolektivním řízením (tj. takový, jako kdybychom cyklické řízení neaplikovali. Uvědomme si, že kolektivní řízení určuje jakýsi „klidový“ úhel náběhu listu, a cyklické řízení se ke kolektivnímu „přičítá“.) Od azimutu 270° budeme úhel náběhu nadále zmenšovat, až v azimutu 0° bude minimální. V tomto azimutu bude největší negativní cyklický akční zásah (největší negativní cyklika, angl. maximu negative cyclic). Analogicky, od azimutu 0° až do 180° budeme úhel náběhu zase postupně zvětšovat, takže v azimutu 90° bude zase cyklický akční zásah nulový (list bude v „klidové“ posici určené kolektivem) a ve 180° bude úhel náběhu zase maximální. Situaci přehledně ukazuje obrázek 6.15.



Obr. 6.15 – Cyklické řízení pro levé klonění

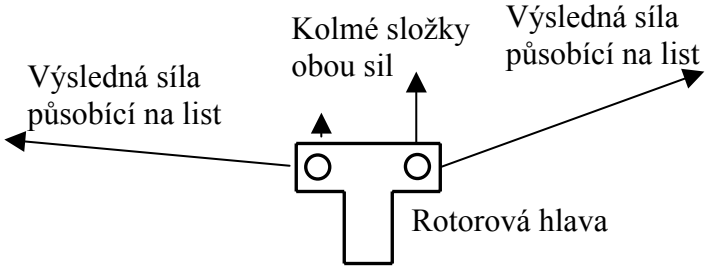
Podívejme se nyní, jak se bude měnit vzepětí listu v závislosti na úhlu náběhu. Maximálního vzepětí bude dosahovat v azimutu 270° (pozor – nezapomínejme na gyroskopický efekt! Maximální úhel náběhu dosahujeme sice už v azimutu 180°, ale maximálního vzepětí bude dosaženo až o čtvrtotáčku dál – viz. část 6.1.4.1). Analogicky, minimální vzepětí nastane v azimutu 90°. Čeho jsme tím vším dosáhli? Díky periodickým změnám vzepětí listů si nyní můžeme rovinu otáčení rotoru představit jakoby nakloněnou doleva. Tím už ovšem působiště tahové síly rotoru neleží v těžišti, ale je posunuto poněkud „do strany“. Tahová síla rotoru nám spolu s tíhovou silou tvoří dvojici, jejíž moment má snahu otáčet vrtulník kolem podélné osy doleva (viz. obr. 6.16). To je ovšem přesně to, čeho jsme na začátku chtěli dosáhnout.



Obr. 6.16 – Působení sil na vrtulník při aplikaci levé cykliky

Není to ovšem jediná síla (respektive moment), který jsme cyklickým řízením vyvolali. Protože rotorové listy se v pantech nemohou pohybovat zcela volně, ale jejich pohyb je tlumen gumovými tlumiči, přenáší se na trup také tato síla. Takže když list zvětšuje svoje vzepětí, přes gumový tlumič se síla tlačící ho vzhůru přenáší také na rotorovou hlavu a tím i na trup. Protože na jedné straně list stoupá a na druhé klesá, máme další dvojici opačných sil, které na vrtulník vyvozují kroutivý moment.

Poslední dvojici sil působících na vrtulník si nejlépe můžeme představit takto: Výsledná síla působící na rotorový list je složením odstředivé síly způsobené otáčením listu a aerodynamické vztlakové síly, generované listem. Tato síla působí na list podélně (viz. obr. 6.9). Dvojice těchto sil, působících na oba listy, vyvolává přes páku tvořenou rotorovou hlavou další moment, viz. obr. 6.17.



Obr. 6.17 – Moment rotorových listů působící na rotorovou hlavu

Je zřejmé, že stejným způsobem lze také naklánět vrtulník kolem příčné osy vpřed a vzad a oba způsoby naklánění lze libovolně kombinovat, jednoduše tím, že vhodně zvolíme azimuty pro maximální pozitivní a negativní cykliku. Ovšem, jak už to bohužel bývá, celá problematika cyklického řízení je mnohem komplikovanější, než by se zdálo z předchozího příkladu. V následujících několika odstavcích probereme hlavní problémy cyklického řízení.

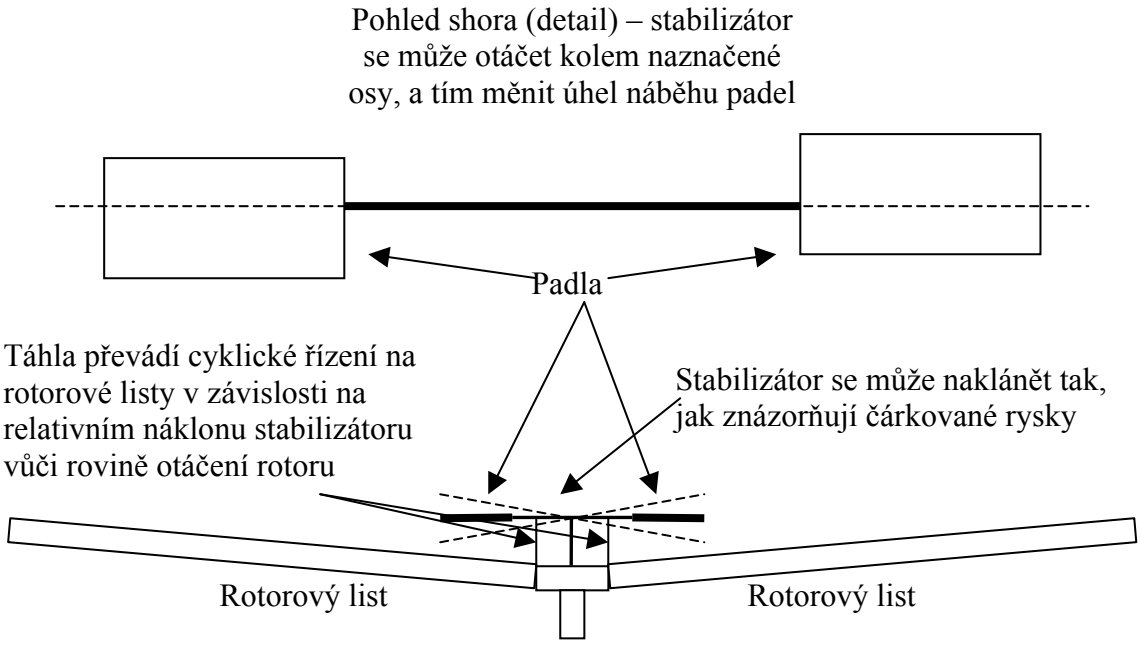
Samozřejmě je dobré vědět, jakým způsobem a jak rychle bude vrtulník reagovat na cyklické řízení. Zamysleme se nejprve nad tím, co se stane, když výše popsané síly začnou působit. Vzhledem ke své setrvačnosti se trup bude otáčet zprvu pomalu. Ovšem tím, že se pootočí, pootočí se samozřejmě také rovina otáčení rotoru – tj. nakloní se ještě víc na tu stanu, na kterou se trup otáčí. Protože tíhová síla působí stále stejným směrem, ale tahová síla rotoru se otáčí společně s trupem, bude se moment této dvojice sil postupně zmenšovat. Účinek tohoto momentu se dá konstrukčně regulovat vzdáleností rotorové hlavy od těžiště vrtulníku. Čím menší tato vzdálenost bude, tím menší bude moment vyvozovaný touto dvojicí. Velikost dalších dvou momentů popsaných výše se nemění. Otáčení vrtulníku se tedy bude postupně zrychlovat (nelineárně), až do dosažení určité úhlové frekvence, dané fyzikálními parametry vrtulníku (až do momentu, kdy bude dosaženo rovnováhy mezi otočným momentem a odporem vzduchu).

Jak rychle ale bude této ustálené rychlosti otáčení dosaženo a jak velká bude? Na to se nedá jednoznačně odpovědět, závisí to samozřejmě na konkrétním typu vrtulníku – především na jeho hmotnosti a konstrukci rotoru. Dá se ovšem říct, že pro modelové vrtulníky obecně platí, že tato rychlost je velmi vysoká a reakční doba velmi krátká. Je to způsobeno jejich relativně nízkou hmotností a značnou tahovou silou rotoru. Uvědomme si, že tato tahová síla musí například ve visu držet celou váhu vrtulníku. Na jeho otočení kolem osy by ale stačila síla řádově menší. Proto, když začneme vrtulník otáčet změnou směru této síly, bude jeho reakce velice rychlá. Dokonce tak rychlá, že pro lidského pilota by takovýto vrtulník byl bez dalších úprav zcela neřiditelný. Pro modelové vrtulníky se ustálená rychlost otáčení typicky pohybuje v řádech až 360° za méně než jednu sekundu. Reakční doba se pohybuje v řádu desítek milisekund.

Další nepříjemnou vlastností je, že vrtulník reaguje na cyklické řízení nelineárně – po akčním zásahu na okamžik téměř nereaguje a potom, pokud je akční zásah stále stejný, začne otáčení zrychlovat až do ustálené hodnoty úhlové frekvence. Dokonce i toto úhlové zrychlení se mění, nejprve roste a potom klesá. Celý systém je tedy silně nelineární a nestabilní – vrtulník rozhodně nemá snahu vracet se při malé odchylce zpět do stabilního stavu, naopak „utíká“ čím dál víc. Je zřejmé, že takovýto vrtulník by se bez dalších úprav dal řídit jedině s pomocí moderní řídicí techniky. Protože ovšem vrtulníky vznikaly už v dobách, kdy ještě počítače byly v plenkách, a bylo nutné stabilizovat je tak, aby je mohl řídit i lidský pilot, našlo se mechanické řešení. Toto řešení představuje takzvaný Bellův nebo vylepšený Bell-Hillerův systém.

6.1.5.2 Bellův systém řízení rotorových listů

Výše uvedené nedostatky cyklického řízení částečně odstraňuje Bellův systém. Podstatnou součástí tohoto systému je takzvaný stabilizátor (angl. flybar). Tento stabilizátor se zpravidla nachází na ose rotoru, umístěn nad hlavními rotorovými listy. Vypadá jako malý pomocný rotor – je to kulatá ocelová tyčka s malými křidélky (padly, angl. paddle) na obou koncích. Stabilizátor je uchycen tak, že se může otáčet kolem své osy (tím se mění úhel náběhu padel) a současně také naklánět (podobně jako když hlavní rotorové listy mění vzepětí). Celý mechanismus je schematicky znázorněn na obrázku 6.18.



Obr. 6.18 – Stabilizátor

Protože padla jsou mnohem menší než rotorové listy, vyvozují také mnohem menší vztlak. Jsou navržena tak, aby rychlost jejich odezvy vyhovovala pilotovi. Rychlost odezvy stabilizátoru se dá nastavovat velikostí padel, jejich hmotností atd. Cyklické řízení se aplikuje pouze na stabilizátor, který je ovšem mechanicky spřažen s hlavními rotorovými listy. Tím se upravuje (zpomaluje) dynamika celého systému. Systém se také stabilizuje. Funkci Bellova systému a stabilizátoru si opět nejlépe ukážeme na příkladu.

Předpokládejme, stejně jako v příkladu na cyklické řízení, že chceme vrtulník naklonit doleva. Cyklické řízení se nyní aplikuje pouze na stabilizátor, a to stejným způsobem jako když bylo použito přímo na rotorové listy. To má za následek, že stabilizátor se nakloní doleva (ze stejných příčin, které byly vysvětleny v článku 6.1.5.1). Tento náklon je mechanicky převeden na hlavní rotorové listy, takže vlastně „přenáší“ cyklické řízení i na ně, ovšem v omezené míře. Aplikace cykliky na hlavní rotorové listy má za následek, že vrtulník se začne naklánět. Jak se ovšem naklání rovina otáčení rotoru, vyrovnává se její relativní sklon vůči stabilizátoru. Proto se akční zásah (daný relativním sklonem stabilizátoru vůči rovině otáčení rotoru) začne zmenšovat. Naopak pokud rotor „zaostává“ za stabilizátorem, akční zásah se automaticky zvětší. Vidíme tedy, že stabilizátor, hoden svého jména, automaticky reguluje akční zásah a tím i výslednou rychlost otáčení vrtulníku tak, aby byla v každém okamžiku přibližně stejná.

Stabilizátor ovšem také zpomaluje rychlost odezvy na cyklické řízení. Uvědomme si, že cyklika se nejprve aplikuje na stabilizátor, ten se po jistém čase nakloní, tím přenese cyklické řízení na rotorové listy, a teprve potom začne vrtulník zprvu pomalu a neochotně zatáčet. Pro mnohé vrtulníky prodlouží tento systém reakční dobu nad únosnou mez, takže jejich odezva je už neúměrně pomalá. Tento problém Bellova řídicího systému řeší vylepšený systém Bell-Hillerův.

6.1.5.3 Bell-Hillerův systém řízení rotorových listů

Bell-Hillerův systém funguje principielně stejně jako výše popsaný systém Bellův, je ovšem poněkud sofistikovanější. V zásadě pracuje tak, že cyklický akční zásah se aplikuje jak na stabilizátor, tak v omezené míře přímo na rotorové listy. Na stabilizátor se aplikuje „plný“

akční zásah, na rotorové listy pouze jeho malá část (například pokud by měl akční zásah změnit úhel náběhu listů o 5°, změní se úhel náběhu padel o plných 5°, ale na hlavní rotorové listy se aplikuje cyklika v rozsahu pouze třeba 1°). Tento způsob zrychluje odezvu vrtulníku na cyklické řízení a současně zachovává výhody Bellova systému.

Na závěr této kapitoly neuškodí malé shrnutí. Zapamatujme si tedy, že kolektivní řízení se používá k řízení velikosti tahu rotoru, cyklické řízení pak k naklánění vrtulníku kolem podélné (klonění) i příčné osy (klopení). Protože odezva vrtulníku na cyklické řízení je příliš rychlá a nelineární, používá se k úpravě dynamiky a stabilizaci vrtulníku stabilizátor, řízený buď Bellovým, nebo častěji Bell-Hillerovým systémem.

6.1.5.4 Vliv otáček motoru na kolektivní řízení

V této části zjistíme, proč je výhodné udržovat konstantní otáčky rotoru a ke změnám velikosti jeho tahu používat kolektivní řízení (změnu úhlu náběhu rotorových listů). Zřejmě by bylo možné měnit tento tah také tak, že úhel náběhu rotorových listů by byl stále stejný a měnili bychom otáčky rotoru, tak, jak to dělají letadla s pevným křídlem s vrtulemi (či spíše dělávala v dřevních dobách, vrtule moderních vrtulových letadel totiž už mají řiditelný úhel náběhu vrtulových listů). Na první pohled by to bylo výhodné uspořádání; otáčky rotoru lze řídit snadno řízením otáček motoru a odpadla by složitá mechanika kolektivního řízení. Podívejme se tedy, proč se tento způsob řízení tahu v praxi příliš často nepoužívá.

Představme si, že máme vrtulník s pevně nastavenými rotorovými listy (zapomeňme na okamžik na potřebu cyklického řízení), který se nachází ve visu a je dokonale stabilizován. Pokud budeme chtít začít stoupat, nezbývá nám, než zvýšit otáčky rotoru. Přidáme tedy plyn a budeme sledovat, co se stane. Nejprve se toho moc dít nebude, protože vzhledem k setrvačnosti a značnému aerodynamickému odporu rotorových listů motoru nějakou dobu potrvá, než zvětšený kroutící moment přinese vyšší otáčky. Otáčky se tedy budou zvětšovat relativně pomalu, rozhodně ne „skokem“. Vrtulník tedy začne pomalu stoupat a jeho vertikální rychlost se bude zvyšovat, až dojde k nové rovnováze. Ale ouha! Tím, že jsme zvýšili kroutící moment motoru, stoupl také reakční moment trupu a vrtulník se začíná roztáčet kolem osy rotoru (a to okamžitě po přidání plynu, ne až po zvýšení otáček rotoru!). Protože máme gyro, zvýší se automaticky úhel náběhu listů zadní vrtulky (neboli pomocného rotoru) a tím se zvětšený reakční moment vyrovná. Vrtulník se ovšem pootočí kolem osy rotoru. Jak začnou otáčky hlavního rotoru stoupat, zvýší se také otáčky zadní vrtulky (ta má pevný převod od motoru, takže její otáčky jsou v konstantním poměru k otáčkám hlavního rotoru) a tím se zvýší i její účinnost. Vrtulník se tedy začne otáčet kolem osy rotoru v opačném směru! Gyro reaguje zmenšením úhlu náběhu listů zadní vrtulky, ale vrtulník se zase pootočí. Problém trvá až do ustálení otáček hlavního rotoru.

Shrňme si tedy obsah předchozího odstavce: když řídíme tahovou sílu rotoru změnou jeho otáček, nevýhodou je zpoždění v reakci na akční zásah (angl. lag), způsobené setrvačností a aerodynamickým odporem listů rotoru. Další nevýhodou je nestabilita ocasu (vrtulník je takzvaně tail-happy), když se po přidání plynu pootočí nejprve na jednu a potom na druhou stranu.

Pokud ovšem udržujeme otáčky rotoru konstantní a jeho tah měníme kolektivním řízením, výše uvedeným problémům se namnoze vyhneme. Otáčky motoru bývají pak po celou dobu letu řízeny samostatným PID regulátorem s konstantní žádanou hodnotou. Vrtulník na změnu úhlu náběhu rotorových listů reaguje velice svižně, a na zvýšení jejich aerodynamického odporu reaguje regulátor přidáním plynu a zvýšením kroutícího momentu, který se na rozdíl od otáček po přidání plynu zvýší velice rychle. Malý chvilkový pokles otáček při zvyšování úhlu náběhu (než zareaguje regulátor) nebývá většinou znatelný.

Nevyhneme se sice problému s úplavem ocasu doleva (pro pravotočivý rotor) vlivem zvýšeného reakčního momentu trupu, ale protože otáčky hlavního rotoru (a tím i zadní vrtulky) se nezmění, k opačnému úplavu už nedojde. Přechod z visu do stoupání je tedy mnohem hladší, ovšem platíme za to komplikovanější mechanikou vrtulníku.

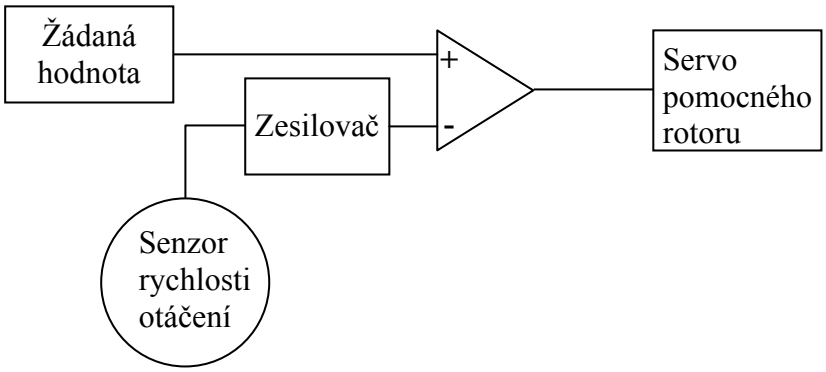
Zajímavým problémem pro vrtulníky s řízeným kolektivem je nastavení optimálních otáček rotoru. Je otázkou, zdali je výhodnější stejný tah docílit pomocí vyšších otáček a menším úhlem náběhu listů, nebo naopak zvětšením úhlu náběhu a poklesem otáček. Obecně se dá říci, že záleží na způsobu letu. Při vyšších otáčkách a menším úhlu náběhu je reakce na kolektivní řízení rychlejší než při opačném poměru, a také nám zbývá větší prostor pro zvyšování tahu (pokud ovšem disponujeme dostatečným výkonem motoru), protože máme větší úhel, o který můžeme náběh zvýšit, než dosáhneme kritického úhlu (viz. kapitola 6.1.2.1). Podstatný je také tvar křivky závislosti kroutivého momentu použitého motoru na otáčkách – je důležité, abychom motor udržovali pokud možno co nejblíže optimálním otáčkám. Tato problematika je poměrně složitá a její podrobnější rozbor přesahuje rámec této práce.

6.1.5.5 Princip řízení ocasní vrtulky (pomocného rotoru)

Nejdůležitějším úkolem pomocného rotoru je vyrovnávat reakční moment působící na trup vrtulníku, vzniklý v důsledku kroutivého momentu motoru působícího na rotor. Problematika týkající se jeho řízení už byla částečně zmíněna v předchozím textu, viz. články 6.1.3.1 a 6.1.5.4.

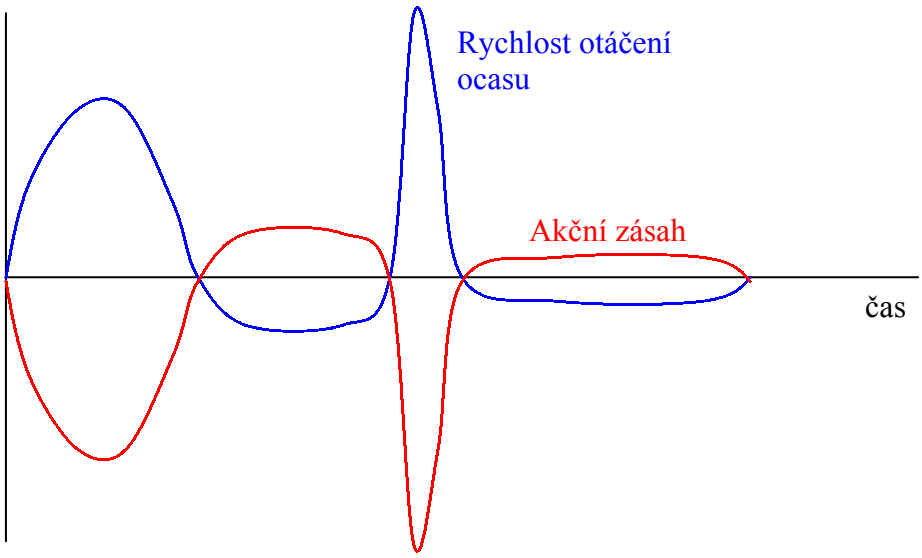
V dřevních dobách modelářského vrtulníkového létání, když se ještě nepoužívala gyra, bylo řízení pomocného rotoru buď zcela ponecháno na umění pilota, nebo bylo založeno na nějaké formě mixování s kolektivem. Jak víme z článku 6.1.5.4, aplikací kolektivního řízení se změní kroutivý moment motoru a tím i reakční moment trupu. Na to je třeba reagovat změnou tahu pomocného rotoru. To buď musel zajistit sám pilot, nebo mu k tomu částečně dopomohla automatika tím, že úhel náběhu listů pomocného rotoru byl nějakým způsobem svázán s kolektivním řízením. Nedostatky tohoto řešení jsou zřejmé – nastavení mixování bylo pevné a nemohlo reagovat na další nepředvídatelné vlivy, jako poryvy větru apod. Dalším problémem je to, že vrtulník je co se otáčení kolem osy rotoru týče velmi nestabilní, protože má velmi malé přirozené tlumení. Proto bylo bez pomoci gyra téměř nemožné udržet ocas v klidu tak, aby se alespoň pomalu neotáčel na tu či onu stranu. Tyto nedostatky byly vyřešeny až s nástupem gyra.

Hlavním úkolem gyra je stabilizovat ocas helikoptéry a poskytovat také jakési umělé tlumení reakce ocasu na pilotní zásah. Princip gyra je zřejmý z blokového schématu na obrázku 6.19.



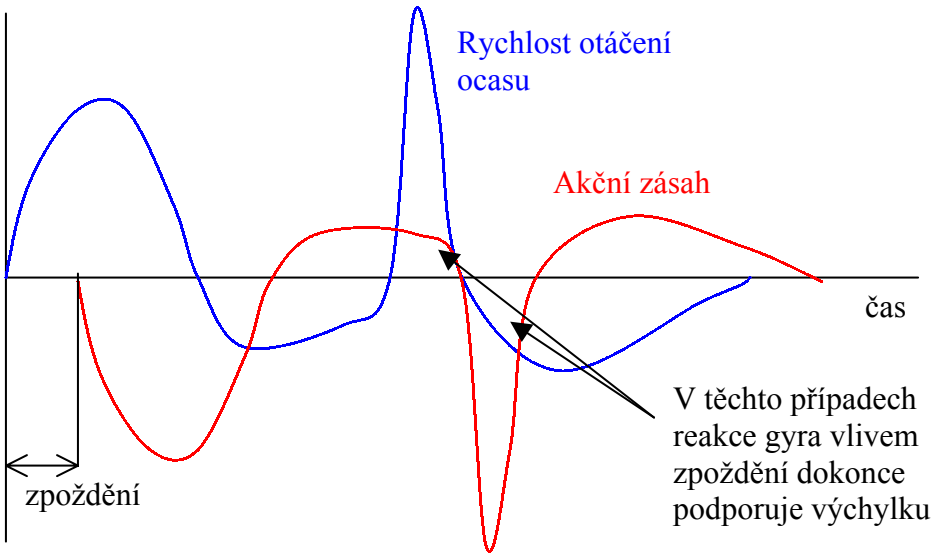
Obr. 6.19 – Blokové schéma gyra

Vidíme, že gyro porovnává údaj o rychlosti otáčení s žádanou hodnotou a podle toho upravuje nastavení úhlu náběhu listů pomocného rotoru. Pokud je žádaná hodnota rychlosti otáčení 0 (chceme, aby vrtulník byl v klidu), chtěli bychom, aby gyro reagovalo na otáčení ocasu tak, jak je znázorněno na obrázku 6.20.



Obr. 6.20 – Ideální reakce gyra

Bohužel takto ideální průběh není ve skutečnosti reálný. Vlivem různých zpoždění je reakce gyra za podnětem v čase značně posunuta. Hlavním zdrojem zpoždění je setrvačnost vrtulníku. Trupu nějakou dobu trvá, něž se zastaví po zvýšení tahu pomocného rotoru (typicky kolem 500ms). Další zpoždění je dáno reakční dobou senzoru a dobou zpracování údaje v gyru (v dnešní době bývají gyra dosti sofistikovaná zařízení s mikroprocesorem a zpracování chvíli trvá) – v součtu kolem 30ms. Dále působí reakční doba serva (mezi 50 až 200ms, podle typu serva) a velké zpoždění může také způsobit příliš dlouhé a pružné táhlo od serva k listům pomocného rotoru. Celkově proto akční zásah značně zaostává, jak je vidět z obrázku 6.21. Při příliš rychlých změnách rychlosti a směru otáčení se dokonce může stát, že gyro otáčení dokonce podporuje. Proto při nadměrně vysokém zesílení zpětné vazby vznikají problémy s oscilacemi ocasu.



Obr. 6.21 – Skutečná reakce gyra

Z tohoto důvodu se snažíme zpoždění reakce gyra eliminovat na nejmenší možnou míru, například úpravou táhel, použitím extrémně rychlého servomotoru apod.

6.1.6 Závěr

Tato kapitola si rozhodně neklade za cíl být vyčerpávající příručkou teorie vrtulníkového letu. Jejím cílem bylo pouze seznámit čtenáře s jeho nejdůležitějšími aspekty a co nejjednodušeji vysvětlit principy mechaniky letu a řízení vrtulníku. Mnoho zde popisovaných jevů bylo záměrně zjednodušeno a mnohé méně podstatné jevy byly zcela zamlčeny. Pokud čtenáři posloužila k získání jednoduchého a praktického náhledu na popisovanou problematiku, pak beze zbytku splnila svůj účel.

6.2 Výpis zdrojového kódu firmwaru automatické krmičky ryb

```
.include "2313def.inc"

;***** Konstanty

.equ StackHr=136  ;vyska zasobniku (SRAM zacina na adrese 96)
.equ Pos=137      ;zacatek posice pistu
.equ ZacData=139  ;zacatek dat prijatych z UART
.equ Tbit=6       ;cislo T-bitu v SREG

;***** Definice pinu

.equ Ena=4        ;enable vystupu na budici motoru L293D
.equ MOut1=0      ;ovladani motoru 1
.equ MOut2=1      ;ovladani motoru 2 (konec 1. vinuti)
.equ MOut3=2      ;ovladani motoru 3
.equ MOut4=3      ;ovladani motoru 4 (konec 2.vinuti)
.equ Doraz=6      ;signalizace dorazu pistu
.equ Tl=2         ;tlacitko
.equ LedZ=3       ;zelena LED
.equ LedC=6       ;cervena LED
.equ RO=0         ;UART data in
.equ DI=1         ;UART data out
.equ DE=4         ;povoleni data out na LTC485
.equ RE=5         ;povoleni data in na LTC485
.equ NC1=5        ;nezapojeny pin (pouzito na ISP programovani FLASH)
.equ NC2=7        ;nezapojeny pin (pouzito na ISP programovani FLASH)

;***** Globalni registrove promenne

.def SZal=R0            ;zalohovani SREG v prubehu osetreni preruseni
.def C0=R1              ;konstanta 0
.def C1=R2              ;konstanta 1
.def C10=R3             ;konstanta 10
.def C255=R4            ;konstanta 255
.def MCount1=R5         ;dolni byte pocitadla kroku motoru
.def MCount2=R6         ;horni byte pocitadla kroku motoru
.def MRef1=R7           ;dolni byte reference pocitadla kroku motoru
.def MRef2=R8           ;horni byte reference pocitadla kroku motoru
.def CTS=R9             ;clear to send-promenna indikujici moznost vysilat
.def NewData=R10        ;flag znamenajici nova data prijata z UART
.def Rev=R11            ;flag znamenajici zmenu smeru otaceni motoru
.def ParTyp=R12         ;typ parity UART
.def Adr1=R13           ;dolni byte adresy zarizeni
.def Adr2=R14           ;horni byte adresy zarizeni
.def Temp=R16           ;viceucelovy pomocny registr
.def TlFree=R17         ;pocitadlo casu od preruseni od tlacitka
.def Count=R18          ;pocitadlo cyklu casovace 0
.def CountP=R19         ;pocitadlo cyklu casovace 1
.def Count1=R20         ;pocitadlo jednicek v bytu pro vypocet parity
.def DataOut=R21        ;registr vysilanych dat
.def DataIn=R22         ;registr prijimanych dat
.def EPData=R23         ;registr dat z/do EEPROM
.def EPAdr=R24          ;registr adresy v EEPROM
.def ByteCount=R25      ;pocitadlo bytu prijatych z UART

;***** Interrupt handlery

.org $0000
```

```
rjmp Start        ;RESET handler

.org INT0addr
rjmp INT0_Han     ;INT0 handler

.org INT1addr
reti              ;nepouzity INT1 handler

.org ICP1addr
reti              ;nepouzity Timer 1 capture handler

.org OC1addr
reti              ;nepouzity Timer 1 compare handler

.org OVF1addr
rjmp Tim1_Ovf     ;Timer 1 overflow handler

.org OVF0addr
rjmp Tim0_Ovf     ;Timer 0 overflow handler

.org URXCaddr
rjmp RX_Complete  ;UART RX complete handler

.org UDREaddr
reti              ;nepouzity UART Data register empty handler

.org UTXCaddr
rjmp TX_Complete  ;UART TX complete handler

.org ACIaddr
reti              ;nepouzity analog comparator handler

;***** Zacatek programu (inicializace)

Start:

cbi PORTB,Ena
sbi PORTB,MOut1
cbi PORTB,MOut2
sbi PORTB,MOut3         ;vychozi nastaveni portu B
cbi PORTB,MOut4
cbi PORTB,LedC
sbi PORTB,NC1
sbi PORTB,NC2

sbi PORTD,Doraz
cbi PORTD,Tl
sbi PORTD,LedZ
sbi PORTD,RO            ;vychozi nastaveni portu D
sbi PORTD,DI
cbi PORTD,DE
cbi PORTD,RE

sbi DDRB,Ena
sbi DDRB,MOut1
sbi DDRB,MOut2
sbi DDRB,MOut3          ;definice vstupu/vystupu portu B
sbi DDRB,MOut4
sbi DDRB,LedC
cbi DDRB,NC1
cbi DDRB,NC2
```

```
cbi DDRD,Doraz
cbi DDRD,Tl
sbi DDRD,LedZ
cbi DDRD,RO             ;definice vstupu/vystupu portu D
sbi DDRD,DI
sbi DDRD,DE
sbi DDRD,RE

ldi ByteCount,255
ldi Count,0
mov C0,Count
mov NewData,C0
mov Rev,C0

ldi Temp,1              ;pocatecni nastaveni registrovych promennych
mov C1,Temp
mov CTS,C1

ldi Temp,10
mov C10,Temp

ldi TlFree,255
mov C255,TlFree

ldi Temp,StackHr
out SPL,Temp            ;inicializace Stack pointeru

ldi ZH,0
ldi ZL,Pos              ;nastaveni Z-registru na pozici pistu

st Z+,C0                ;vynuluj horni i dolni byte pocitadla
st Z,C0                 ;kroku motoru

ldi ZL,ZacData          ;inicializace Z-registru pro adresovani SRAM

ldi EPAdr,0
rcall Cti_EP
mov Adr1,EPData         ;precteni dolniho bytu adresy z EEPROM

inc EPAdr
rcall Cti_EP
mov Adr2,EPData         ;precteni horniho bytu adresy z EEPROM

inc EPAdr
rcall Cti_EP
out UBRR,EPData         ;precteni Baud-rate z EEPROM a nastaveni

inc EPAdr
rcall Cti_EP
mov ParTyp,EPData       ;precteni typu parity z EEPROM a nastaveni

cli                     ;globalni zakaz preruseni

wdr
ldi Temp,15             ;nastaveni interniho watchdog timeru
out WDTCR,Temp

ldi Temp,144
cpse ParTyp,C0          ;nastaveni parametru UART podle typu parity
ldi Temp,148
```

```
out UCR,Temp

out TCCR1A,C0           ;nastaveni casovace 1

ldi Temp,2
out TCCR0,Temp          ;nastaveni periody casovace 0

out TCCR1B,Temp         ;nastaveni periody casovace 1

out MCUCR,Temp          ;nastaveni INT0 na spadovou hranu

ldi Temp,130
out TIMSK,Temp          ;povoleni preruseni od casovace 0 a 1

ldi Temp,64
out GIMSK,Temp          ;povoleni INT0

clt                     ;vynulovani T-bitu (pouzit k indikaci stisku
                        tlacitka)

sei                     ;globalni povoleni preruseni

;***** Hlavni program

            sbis PIND,Doraz         ;pokud je pist na dorazu,
            rjmp Kalib2             ;preskoc autokalibraci

Kalib:      sbis PIND,Doraz         ;opakuj dokud pist nedosahne dorazu
            rjmp Kalib1

            in Temp,SREG            ;precti SREG
            sbrc Temp,Tbit          ;je-li detekovan stisk tlacitka,
            rjmp Kalib2             ;prerus kalibraci

            mov MRef1,C10           ;do dolniho bytu reference dej 10,
            mov MRef2,C0            ;do horniho 0

            rcall Dozadu            ;zavolej rutinu otaceni motorem vzad

            rjmp Kalib

Kalib1:     sbi PORTB,LedC          ;zhasni cervenou LED
            cbi PORTD,LedZ          ;rozsvit zelenou LED

            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Mref1,Z+
            ld Mref2,Z              ;nacti horni byte posice pistu

            com MRef1               ;neguj (bitove) oba registry
            com MRef2

            ldi ZL,Pos

            st Z+,C0                ;vynuluj horni i dolni byte pocitadla
            st Z,C0                 ;kroku motoru

            pop ZL                  ;obnov Z-registr
```

```
            sei                     ;globalni povoleni preruseni

            rcall Dopredu           ;zavolej rutinu otaceni motorem vpred

Kalib2:     cbi PORTD,LedZ          ;rozsvit zelenou LED
            cbi PORTB,LedC          ;rozsvit cervenou LED

            ldi CountP,0

Kalib3:     cpi CountP,15           ;chvilku pockej (15 cyklu casovace 1)
            brlo Kalib3

            sbi PORTB,LedC          ;zhasni obe LED (efekt bliknuti
            sbi PORTD,LedZ          ;po dokonceni kalibrace)

            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Temp,Z+
            ld Temp,Z               ;nacti horni byte posice pistu

            pop ZL                  ;obnov Z-registr
            sei                     ;globalni povoleni preruseni

            clt                     ;vynulovani T-bitu (pouzit k indikaci
                                    ;stisku tlacitka)
            cpi Temp,40
            brlo Main               ;je-li horni byte >=40

            cbi PORTB,LedC          ;rozsvit cervenou LED

Main:       cpse NewData,C0         ;hlavni cyklus programu
            rcall Vykonej_P         ;je-li prijat povel z UART,vykonej ho

            in Temp,SREG            ;precti SREG
            sbrc Temp,Tbit          ;je-li detekovan stisk tlacitka,
            rcall Tl_Zm             ;osetri ho

            rjmp Main

;============================ Vykonej povel ==============================

Vykonej_P:  ldi ZL,ZacData          ;identifikace a vykonani povelu z UART
                                    ;nacti do Z-registru poc. adresu dat
            ld Temp,Z+              ;precti prvni byte

            adiw ZL,2               ;preskoc adresove byty

            cpi Temp,37             ;je-li to %,jdi na konfiguraci
            breq Vykonej_P1

            ld Temp,Z+              ;precti dalsi byte

            cpi Temp,82
            breq Posun_Pist1        ;je-li R,bez na cteni posice pistu

            cpi Temp,80
            breq Posun_Pist2        ;je-li P,bez na posun pistu vpred
```

```
            cpi Temp,83
            breq Posun_Pist3        ;je-li S,bez na posun pistu vzad

            cpi Temp,77
            breq Posli_Jm           ;je-li M,bez na poslani ID

            cpi Temp,70
            breq Posli_FV           ;je-li F,bez na poslani verze firmwaru

            rcall Posli_Busy        ;posli chybovou hlasku,neni-li spravne
            ret                     ;dekodovan povel a skonci

Posun_Pist1:rcall Pist_Pos          ;zavolej rutinu cteni posice pistu
            ret                     ;skonci

Posun_Pist2:rcall Pist_Vpred        ;zavolej rutinu posunu pistu vpred
            ret                     ;skonci

Posun_Pist3:rcall Pist_Vzad         ;zavolej rutinu posunu pistu vzad
            ret                     ;skonci

Posli_Jm:   rcall Posli_ID          ;zavolej rutinu pro poslani jmena
            ret                     ;skonci

Posli_FV:   rcall Posli_FR          ;zavolej rutinu poslani verze firmwaru
            ret                     ;skonci

Vykonej_P1: rcall Konfigurace       ;zavolej obsluznou rutinu rekonfigurace
            ret                     ;skonci

;=============================== Pist vzad ================================

Pist_Vzad:  adiw ZL,5               ;rutina posunu pistu vzad
                                    ;preskoc dalsich 5 bytu
            ld Temp,Z+              ;nacte prvni polovinu horniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            lsl Temp
            lsl Temp                ;logicky posune o 4 bity vlevo
            lsl Temp
            lsl Temp
            mov MRef2,Temp          ;presune ji do horniho bytu reference

            ld Temp,Z+              ;nacte druhou polovinu horniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            add MRef2,Temp          ;pricte ji k prvni polovine (horni)

            ld Temp,Z+              ;nacte prvni polovinu dolniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            lsl Temp
            lsl Temp                ;logicky posune o 4 bity vlevo
            lsl Temp
            lsl Temp
            mov MRef1,Temp          ;presune ji do dolniho bytu reference

            ld Temp,Z               ;nacte druhou polovinu dolniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            add MRef1,Temp          ;pricte ji k prvni polovine (horni)

            cbi PORTB,LedC          ;rozsvit cervenou LED

            ldi DataOut,33
```

```
            rcall Posli_Byte

            mov DataOut,Adr1
            rcall Posli_Byte
                                  ;vysli potvrzeni o vykonani povelu
            mov DataOut,Adr2
            rcall Posli_Byte

            ldi DataOut,13
            rcall Posli_Byte

            rcall Dozadu          ;zavolej rutinu otaceni motoru vzad

            sbi PORTB,LedC        ;zhasni cervenou LED

            sbic PIND,Doraz       ;pokud pist neni na dorazu,
            rjmp Pist_Vzad1       ;bez na konec

            cli                   ;globalni zakaz preruseni
            push ZL               ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            st Z+,C0              ;vynuluj horni i dolni byte pocitadla
            st Z,C0               ;kroku motoru

            pop ZL                ;obnov Z-registr
            sei                   ;globalni povoleni preruseni

            ret                   ;skonci

Pist_Vzad1: cli                   ;globalni zakaz preruseni
            push ZL               ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Temp,Z+
            ld Temp,Z             ;nacteni horn. bytu posice pistu z SRAM

            pop ZL                ;obnov Z-registr
            sei                   ;globalni povoleni preruseni

            cpi Temp,40
            brlo Pist_Vzad2       ;je-li poslednich 15% objemu krmiva,

            cbi PORTB,LedC        ;rozsvit cervenou LED

Pist_Vzad2: ret                   ;ukonci podprogram

;============================ Pist vpred ================================

Pist_Vpred: adiw ZL,5             ;rutina posunu pistu vpred
                                  ;preskoc dalsich 5 bytu
            ld Temp,Z+            ;nacte prvni polovinu horniho bytu
            rcall Uprav_ZHex      ;prevede ji z hexakodu do binaru
            lsl Temp
            lsl Temp              ;logicky posune o 4 bity vlevo
            lsl Temp
            lsl Temp
            mov MRef2,Temp        ;presune ji do horniho bytu reference
```

```
            ld Temp,Z+              ;nacte druhou polovinu horniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            add MRef2,Temp          ;pricte ji k prvni polovine (horni)

            ld Temp,Z+              ;nacte prvni polovinu dolniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            lsl Temp
            lsl Temp                ;logicky posune o 4 bity vlevo
            lsl Temp
            lsl Temp
            mov MRef1,Temp          ;presune ji do dolniho bytu reference

            ld Temp,Z               ;nacte druhou polovinu dolniho bytu
            rcall Uprav_ZHex        ;prevede ji z hexakodu do binaru
            add MRef1,Temp          ;pricte ji k prvni polovine (horni)

            cbi PORTD,LedZ          ;rozsvit zelenou LED

            ldi DataOut,33
            rcall Posli_Byte

            mov DataOut,Adr1
            rcall Posli_Byte
                                    ;posli potvrzeni o vykonani povelu
            mov DataOut,Adr2
            rcall Posli_Byte

            ldi DataOut,13
            rcall Posli_Byte

            ldi ZL,Pos

            ld Temp,Z+
            add Temp,MRef1          ;nacti aktualni posici pistu a pricti
                                    ;zadanou
            ld Temp,Z
            adc Temp,MRef2

            cpi Temp,48
            brsh Pist_Vpred3        ;je-li posun mimo rozsah,osetri

Pist_Vpred1:rcall Dopredu           ;zavolej rutinu otaceni motoru vpred

            sbi PORTD,LedZ          ;zhasni zelenou led

            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Temp,Z+
            ld Temp,Z               ;nacti horni byte posice pistu

            pop ZL                  ;obnov Z-registr
            sei                     ;globalni povoleni preruseni

            sbi PORTB,LedC          ;zhasni cervenou LED

            cpi Temp,40
            brlo Pist_Vpred2        ;zbyva-li poslednich 15% krmiva,
```

```
            cbi PORTB,LedC          ;rozsvit cervenou LED

Pist_Vpred2:ret                     ;ukonci podprogram

Pist_Vpred3:ldi ZL,Pos              ;osetreni parametru mimo rozsah

            ld Temp,Z+              ;nacti dolni byte posice pistu a
            com Temp                ;spocti zbytek do 255
            mov MRef1,Temp          ;to dej jako novou referenci pohybu

            ld Temp,Z               ;nacti horni byte posice pistu

            cpi Temp,48             ;je-li horni byte >=48 (horni hranice
            brsh Pist_Vpred2        ;rozsahu pohybu pistu),zrus akci

            subi Temp,255           ;odecti od obsahu registru FF
            com Temp                ;prevrat zapornou hodnotu (abs.hodnota)
            subi Temp,207           ;odecti 207 (doplnek od 48 do 255)
            mov MRef2,Temp          ;to zadej jako novou referenci pohybu

            rjmp Pist_Vpred1        ;skoc na pohyb pistem

;============================= Posice pistu ==============================

Pist_Pos:   ldi DataOut,33          ;posle masterovi aktualni posici pistu
            rcall Posli_Byte

            mov DataOut,Adr1        ;vysli hlavicku prenosu (!AA)
            rcall Posli_Byte

            mov DataOut,Adr2
            rcall Posli_Byte

            ldi DataOut,48
            rcall Posli_Byte
            rcall Posli_Byte        ;vysli povinne 4 prvni 0
            rcall Posli_Byte
            rcall Posli_Byte

            ldi ZL,Pos

            ld Temp,Z               ;nacti dolni byte posice pistu
            andi Temp,15            ;demaskuj nizsi 4 bity
            rcall Uprav_DoHex       ;uprav do hexakodu
            push Temp               ;schovej do zasobniku

            ld Temp,Z+              ;znovu nacti dolni byte posice pistu
            lsr Temp
            lsr Temp                ;logicky posun o 4 bity vpravo
            lsr Temp
            lsr Temp
            rcall Uprav_DoHex       ;uprav do hexakodu
            push Temp               ;schovej do zasobniku

            ld Temp,Z               ;nacti horni byte posice pistu
            andi Temp,15            ;demaskuj nizsi 4 bity
            rcall Uprav_DoHex       ;uprav do hexakodu
            push Temp               ;schovej do zasobniku

            ld Temp,Z               ;znovu nacti horni byte posice pistu
            lsr Temp
```

```
            lsr Temp                ;logicky posun o 4 bity vpravo
            lsr Temp
            lsr Temp
            rcall Uprav_DoHex       ;uprav do hexakodu
            push Temp               ;schovej do zasobniku

            pop DataOut
            rcall Posli_Byte        ;posli horni puli horniho bytu

            pop DataOut
            rcall Posli_Byte        ;posli dolni puli horniho bytu

            pop DataOut
            rcall Posli_Byte        ;posli horni puli dolniho bytu

            pop DataOut
            rcall Posli_Byte        ;posli dolni puli dolniho bytu

            ldi DataOut,13
            rcall Posli_Byte        ;posli CR

            ret                     ;skonci

;====================== Prevod z hexakodu do binaru =======================

Uprav_ZHex: cpi Temp,65             ;prevadi cislo z hexakodu do binaru
            brsh Uprav_ZHex1        ;rozhodne,je-li cislo nebo pismeno

            subi Temp,48            ;pokud cislo,odecti ASCII kod 0
            ret                     ;a skonci

Uprav_ZHex1:subi Temp,55            ;pokud pismeno,odecti 55 tak, aby A
            ret                     ;vyslo 10 atd. a skonci

;====================== Prevod z binaru do hexakodu =======================

Uprav_DoHex:cpi Temp,10             ;prevadi cislo z binaru do hexakodu
            brsh Uprav_DoHex1       ;rozhodne,je-li cislo nebo pismeno

            ldi DataOut,48
            add Temp,DataOut        ;pokud cislo,pricti ASCII kod 0
            ret                     ;a skonci

Uprav_DoHex1:ldi DataOut,55         ;pokud pismeno,pricti 55 tak, aby A
            add Temp,DataOut        ;vyslo 10 atd.
            ret                     ;a skonci

;========================== Posli ID zarizeni ============================

Posli_ID:   ldi DataOut,33          ;posle masterovi jmeno zarizeni
            rcall Posli_Byte

            mov DataOut,Adr1
            rcall Posli_Byte

            mov DataOut,Adr2
            rcall Posli_Byte

            ldi DataOut,75          ;K
            rcall Posli_Byte
```

```
            ldi DataOut,82          ;R
            rcall Posli_Byte

            ldi DataOut,77          ;M
            rcall Posli_Byte

            ldi DataOut,73          ;I
            rcall Posli_Byte

            ldi DataOut,84          ;T
            rcall Posli_Byte

            ldi DataOut,75          ;K
            rcall Posli_Byte

            ldi DataOut,79          ;O
            rcall Posli_Byte

            ldi DataOut,13
            rcall Posli_Byte

            ret                     ;vrat se

;========================= Posli verzi firmwaru =========================

Posli_FR:   ldi DataOut,33          ;posle masterovi verzi firmwaru
            rcall Posli_Byte

            mov DataOut,Adr1
            rcall Posli_Byte

            mov DataOut,Adr2
            rcall Posli_Byte

            ldi DataOut,54          ;verze 6
            rcall Posli_Byte

            ldi DataOut,13
            rcall Posli_Byte

            ret                     ;vrat se

;============================ Stisk tlacitka ==============================

Tl_Zm:      cpi TlFree,255          ;osetreni stisku tlacitka
            brlo Tl_Zm              ;cekej na TlFree (kvuli filtraci sumu)

            sbi PORTB,LedC
            sbi PORTD,LedZ          ;zhasni obe LED

            ldi CountP,0

Tl_Zm1:     sbic PIND,Tl
            rjmp Tl_Zm2
                                    ;kontroluj stisk tlacitka po 20 cyklu
            cpi CountP,20
            brlo Tl_Zm1

            cp Rev,C0
            breq Vpred
                                    ;bylo-li tlacitko drzeno,zavolej
```

```
            cp Rev,C1               ;vpred nebo vzad podle flagu Rev
            breq Vzad

Tl_Zm2:     ldi TlFree,0

Tl_Zm3:     cpi TlFree,255          ;nebylo-li tlacitko drzeno, pockej
            brlo Tl_Zm3             ;na TlFree (kvuli filtraci sumu)

            clt                     ;vynuluj T-bit

            out TCNT1H,C0
            out TCNT1L,C0           ;vynulovani konstanty casovace 1

            ldi CountP,0

Tl_Zm4:     brts Dvouklik           ;detekuj dvouklik (druhy stisk
                                    ;tlacitka v prubehu 4 cyklu)
            cpi CountP,4
            brlo Tl_Zm4

Tl_Zm5:     clt                     ;vynuluj T-bit

            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Temp,Z+
            ld Temp,Z               ;nacti horni byte posice pistu

            pop ZL                  ;obnov Z-registr
            sei                     ;globalni povoleni preruseni

            cpi Temp,40
            brlo Tl_Zm6             ;je-li horni byte >=40

            cbi PORTB,LedC          ;rozsvit cervenou LED

            ret                     ;a skonci

Tl_Zm6:     sbi PORTB,LedC          ;zhasni cervenou LED

            ret                     ;a skonci

Vpred:      cbi PORTD,LedZ          ;pohyb vpred-rozsvit zelenou LED

            mov MRef1,C10           ;do dolniho bytu reference dej 10,
            mov MRef2,C0            ;do horniho 0

            rcall Dopredu           ;zavolej rutinu otaceni motorem vpred

            sbis PIND,Tl            ;opakuj dokud je stisknuto tlacitko
            rjmp Vpred

            sbi PORTD,LedZ          ;zhasni zelenou LED

            mov Rev,C1              ;zmen flag Rev

            rjmp Tl_Zm5             ;bez na konec

Vzad:       sbis PIND,Doraz         ;pohyb vzad-pokud pist dosahl dorazu,
```

```
            rjmp Vzad1              ;skonci

            cbi PORTB,LedC          ;rozsvit cervenou LED

            mov MRef1,C10           ;do dolniho bytu reference dej 10,
            mov MRef2,C0            ;do horniho 0

            rcall Dozadu            ;zavolej rutinu otaceni motorem vzad

            sbis PIND,Tl            ;opakuj dokud je stisknuto tlacitko
            rjmp Vzad

Vzad1:      sbi PORTB,LedC          ;zhasni cervenou LED

            mov Rev,C0              ;zmen flag Rev

            sbic PIND,Doraz         ;pokud pist neni na dorazu,
            rjmp Tl_Zm5             ;bez na konec

            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            st Z+,C0                ;vynuluj horni i dolni byte pocitadla
            st Z,C0                 ;kroku motoru

            pop ZL                  ;obnov Z-registr
            sei                     ;globalni povoleni preruseni

            rjmp Tl_Zm5             ;bez na konec

Dvouklik:   cbi PORTB,LedC          ;je-li detekovan dvouklik,rozsvit
                                    ;cervenou LED
            out TCNT1H,C0
            out TCNT1L,C0           ;vynulovani konstanty casovace 1

            ldi CountP,0

Dvouklik1:  cpi Countp,2            ;pockej (kvuli filtraci sumu)
            brlo Dvouklik1

            clt                     ;vynuluj T-bit

Dvouklik2:  sbis PIND,Doraz         ;opakuj dokud pist nedosahne dorazu
            rjmp Dvouklik3

            in Temp,SREG            ;precti SREG
            sbrc Temp,Tbit          ;je-li detekovan stisk tlacitka,
            rjmp Tl_Zm5             ;skonci

            mov MRef1,C10           ;do dolniho bytu reference dej 10,
            mov MRef2,C0            ;do horniho 0

            rcall Dozadu            ;zavolej rutinu otaceni motorem vzad

            rjmp Dvouklik2

Dvouklik3:  cbi PORTD,LedZ          ;rozsvit zelenou LED

            cli                     ;globalni zakaz preruseni
```

```
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            st Z+,C0                ;vynuluj horni i dolni byte pocitadla
            st Z,C0                 ;kroku motoru

            pop ZL                  ;obnov Z-registr
            sei                     ;globalni povoleni preruseni

            ldi CountP,0

Dvouklik4:  cpi CountP,15           ;chvilku pockej (15 cyklu casovace 1)
            brlo Dvouklik4

            sbi PORTB,LedC          ;zhasni obe LED (efekt bliknuti
            sbi PORTD,LedZ          ;po dokonceni kalibrace)

            rjmp Tl_Zm5             ;bez na konec

;============================ Konfigurace ================================

Konfigurace:ld EPData,Z+            ;rutina nastaveni nove konfigurace
            ldi EPAdr,0             ;nacti horni byte nove adresy a zapis
            rcall Zapis_EP          ;ho do EEPROM

            ld EPData,Z+
            inc EPAdr               ;nacti dolni byte nove adresy a zapis
            rcall Zapis_EP          ;ho do EEPROM

            adiw ZL,3               ;preskoc neplatne byty
            ld EPData,Z+            ;nacti kod Baud-rate

            rcall Prepocti_BR       ;vypocti konstantu z kodu

            inc EPAdr               ;novou Baud-rate zapis do
            rcall Zapis_EP          ;EEPROM

            ld EPData,Z+            ;precti neplatny byte
            ld EPData,Z             ;precti kod parity

            lsr EPData
            lsr EPData
            andi EPData,3

            inc EPAdr               ;zapis novy typ parity do
            rcall Zapis_EP          ;EEPROM

            ldi DataOut,33
            rcall Posli_Byte

            mov DataOut,Adr1
            rcall Posli_Byte
                                    ;vysli potvrzeni provedeni povelu,
            mov DataOut,Adr2        ;naposledy podle stareho nastaveni
            rcall Posli_Byte

            ldi DataOut,13
            rcall Posli_Byte

            ldi EPAdr,0
```

```
            rcall Cti_EP

            mov Adr1,EPData         ;nastav novy dolni byte adresy

            inc EPAdr
            rcall Cti_EP

            mov Adr2,EPData         ;nastav novy horni byte adresy

            inc EPAdr
            rcall Cti_EP

            out UBRR,EPData         ;nastav novou Baud-rate

            inc EPAdr
            rcall Cti_EP

            mov ParTyp,EPData       ;nastav novy typ parity

            ldi Temp,144
            cpse ParTyp,C0          ;znovunastaveni parametru UART
            ldi Temp,148
            out UCR,Temp

            cbi PORTB,LedC
            cbi PORTD,LedZ          ;rozsvit obe LED

            ldi CountP,0

Konfigurace1:cpi CountP,15          ;chvilku pockej (15 cyklu)-efekt
            brlo Konfigurace1       ;bliknuti

            sbi PORTB,LedC
            sbi PORTD,LedZ          ;zhasni obe LED

            ret                     ;skonci

;=========== Vypocet kalibracni konstanty UART z kodu BAUD-rate ===========

Prepocti_BR:cpi EPData,53           ;urci kalibracni konstantu UART
            breq PBR5               ;z kodu

            cpi EPData,54
            breq PBR6

            cpi EPData,55
            breq PBR7

            cpi EPData,56
            breq PBR8

            ldi EPData,4            ;nastav konstantu pro 115200
            ret                     ;skonci

PBR5:       ldi EPData,129          ;nastav konstantu pro 4800
            ret                     ;skonci

PBR6:       ldi EPData,64           ;nastav konstantu pro 9600
            ret                     ;skonci

PBR7:       ldi EPData,32           ;nastav konstantu pro 19200
```

```
            ret                     ;skonci

PBR8:       ldi EPData,15           ;nastav konstantu pro 38400
            ret                     ;skonci

;========================== Zapis do EEPROM ==============================

Zapis_EP:   cli                     ;rutina pro zapis dat do EEPROM
            wdr                     ;globalni zakaz preruseni
                                    ;watchdog reset
Zapis_EP1:  sbic EECR,EEWE          ;pooluj bit EEWE (cekej dokud neni 0)
            rjmp Zapis_EP1

            out EEAR,EPAdr          ;nastav adresu
            out EEDR,EPData         ;nastav data

            ldi Temp,4
            out EECR,Temp           ;nastav bit EEMWE

            ldi Temp,6
            out EECR,Temp           ;nastav bit EEWE

            sei                     ;globalni povoleni preruseni

            ret                     ;konec

;============================ Cteni z EEPROM ==============================

Cti_EP:     cli                     ;rutina pro cteni dat z EEPROM
            wdr                     ;globalni zakaz preruseni
                                    ;watchdog reset
Cti_EP1:    sbic EECR,EEWE          ;pooluj bit EEWE (cekej dokud neni 0)
            rjmp Cti_EP1

            out EEAR,EPAdr          ;nastav adresu

            ldi Temp,1
            out EECR,Temp           ;nastav bit EERE

            in EPData,EEDR          ;precti data

            sei                     ;globalni povoleni preruseni

            ret                     ;konec

;============================= Posli byte ================================

Posli_Byte: ldi Count,0             ;posle byte po seriove lince

            cpi DataOut,33          ;je-li !, cekej na zacatku
            breq Posli_byte1

            cpi DataOut,63          ;je-li ?,cekej na zacatku
            brne Posli_Byte2

Posli_Byte1:cpi Count,10            ;pockej 10 cyklu casovace 0
            brlo Posli_Byte1

            sbi PORTD,DE            ;povol vysilac RS-485

Posli_Byte2:mov CTS,C0              ;vynuluj flag CTS
```

```
            ldi Count1,0            ;vynuluj pocitadlo jednicek

            ldi Temp,9

            push DataOut            ;schovej DataOut do zasobniku

Posli_Byte3:ror DataOut

            brcc Posli_Byte4        ;v cyklu pomoci rotace pres carry
            inc Count1              ;spocitej jednicky ve vysilanem bytu

Posli_Byte4:dec Temp

            cpi Temp,0
            brne Posli_Byte3

            pop DataOut             ;obnov DataOut ze zasobniku

            cp ParTyp,C0
            brne Posli_Byte9        ;je-li parita,bez ji osetrit

            ldi Temp,72
            out UCR,Temp            ;neni-li parita,nastav podle toho UART

Posli_Byte5:out UDR,DataOut         ;posli data

Posli_Byte6:cp CTS,C0               ;pooluj CTS (znamena ukonceni prenosu)
            breq Posli_Byte6

            cpi DataOut,13          ;neni-li konec prenosu,vrat se a
            brne Posli_Byte8        ;nemen nastaveni UART

            ldi Count,0

Posli_Byte7:cpi Count,2             ;pockej 2 cykly casovace
            brlo Posli_Byte7

            cbi PORTD,DE            ;nastav vysilac RS-485 na Hi-Z

            ldi Temp,144
            cpse ParTyp,C0          ;znovunastaveni parametru UART
            ldi Temp,148
            out UCR,Temp

            mov NewData,C0          ;vynuluj flag novych dat
            cbi PORTD,RE            ;povol prijimac RS-485

Posli_Byte8:ret                     ;skonci

Posli_Byte9:ldi Temp,2

            cpse ParTyp,Temp        ;je-li licha parita,
            com Count1              ;bitove neguj pocitadlo jednicek

            andi Count1,1           ;demaskuj 0. bit pocitadla jednicek

            ldi Temp,76             ;nastav datovy ramec na 9 platnych bitu
            or Temp,Count1          ;nastav 9. bit podle parity
            out UCR,Temp            ;nastav UART

            rjmp Posli_Byte5        ;bez vysilat
```

```
;============================== Posli Busy ================================

Posli_Busy: ldi DataOut,63          ;poslani stavu BUSY
            rcall Posli_Byte

            mov DataOut,Adr1
            rcall Posli_Byte        ;posli chybovou hlasku

            mov DataOut,Adr2
            rcall Posli_Byte

            ldi DataOut,13
            rcall Posli_Byte

            ret                     ;vrat se

;============================== Dopredu ================================

Dopredu:    mov MCount1,C0          ;rutina pro otaceni motorem vpred
            mov MCount2,C0          ;vynuluj horni i dolni byte pocitadla

            sbi PORTB,Ena           ;povol vystupy budice L293D

Dopredu1:   cp MCount1,MRef1        ;dosahl-li dolni byte reference,odskoc
            breq Dopredu3           ;zkontrolovat horni byte

Dopredu2:   cp MCount1,C255         ;dosahl-li dolni byte FF,odskoc ho
            breq Dopredu4           ;vynulovat a inkrementovat horni byte

            cpse NewData,C0         ;pokud prisla nova data po UART,
            rcall Posli_Busy        ;odpovez ze jsi busy

            rcall Wait_M

            cbi PORTB,MOut3
            sbi PORTB,MOUT4

            rcall Wait_M

            cbi PORTB,MOut1
            sbi PORTB,MOUT2

            rcall Wait_M            ;jeden kompletni cyklus kroku motoru

            sbi PORTB,MOut3
            cbi PORTB,MOUT4

            rcall Wait_M

            sbi PORTB,MOut1
            cbi PORTB,MOUT2

            inc MCount1             ;inkrementuj dolni byte pocitadla
            rjmp Dopredu1           ;vrat se na zacatek

Dopredu3:   cp MCount2,MRef2        ;zkontroluj,odpovida-li horni byte
            brlo Dopredu2           ;referenci-pokud ne,bez na zacatek

            cbi PORTB,Ena           ;pokud ano,nastav vystupy L293D na Hi-Z
```

```
            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Temp,Z
            add Temp,MRef1
            st Z+,Temp
                                    ;aktualizuj polohu pistu v SRAM
            ld Temp,Z
            adc Temp,MRef2
            st Z,Temp

            pop ZL                  ;obnov Z-registr ze zasobniku
            sei                     ;globalni povoleni preruseni

            ret                     ;skonci

Dopredu4:   mov MCount1,C0          ;vynuluj dolni byte pocitadla
            inc MCount2             ;inkrementuj horni byte pocitdla

            rjmp Dopredu1           ;vrat se na zacatek

;=============================== Dozadu ==================================

Dozadu:     mov MCount1,C0          ;rutina pro otaceni motorem vzad
            mov MCount2,C0          ;vynuluj horni i dolni byte pocitadla

            sbi PORTB,Ena           ;povol vystupy budice L293D

Dozadu1:    cp MCount1,MRef1        ;dosahl-li dolni byte reference,
            breq Dozadu3            ;skoc zkontrolovat horni byte

Dozadu2:    cp MCount1,C255         ;dosahl-li dolni byte FF,odskoc ho
            breq Dozadu5            ;vynulovat a inkrementovat horni byte

            cpse NewData,C0         ;pokud prisla nova data po UART,
            rcall Posli_Busy        ;odpovez ze jsi busy

            sbis PIND,Doraz         ;pokud pist dosahl dorazu,
            rjmp Dozadu4            ;skonci

            rcall Wait_M

            cbi PORTB,MOut1
            sbi PORTB,MOUT2

            rcall Wait_M

            cbi PORTB,MOut3
            sbi PORTB,MOUT4

            rcall Wait_M            ;jeden kompletni cyklus kroku motoru

            sbi PORTB,MOut1
            cbi PORTB,MOUT2

            rcall Wait_M

            sbi PORTB,MOut3
            cbi PORTB,MOUT4
```

```
            inc MCount1             ;inkrementuj dolni byte pocitadla
            rjmp Dozadu1            ;a vrat se na zacatek

Dozadu3:    cp MCount2,MRef2        ;zkontroluj,odpovida-li horni byte
            brlo Dozadu2            ;referenci-pokud ne,bez na zacatek

Dozadu4:    cbi PORTB,Ena           ;pokud ano,nastav vystupy L293D na Hi-Z

            cli                     ;globalni zakaz preruseni
            push ZL                 ;schovej Z-registr do zasobniku

            ldi ZL,Pos

            ld Temp,Z
            sub Temp,MRef1
            st Z+,Temp
                                    ;aktualizuj polohu pistu v SRAM
            ld Temp,Z
            sbc Temp,MRef2
            st Z,Temp

            pop ZL                  ;obnov Z-registr ze zasobniku
            sei                     ;globalni povoleni preruseni

            ret                     ;skonci

Dozadu5:    mov MCount1,C0          ;vynuluj dolni byte pocitadla
            inc MCount2             ;inkrementuj horni byte pocitadla

            rjmp Dozadu1            ;vrat se na zacatek

;===================== Cekaci rutina pro rizeni motoru ====================

Wait_M:     ldi Count,0

Wait_M1:    cpi Count,10            ;pocka 10 cyklu casovace
            brlo Wait_M1

            ret

;=========== INT0 handler (preruseni oznamujici stisk tlacitka) ===========

INT0_Han:   in SZal,SREG            ;schovej SREG

            cpi TlFree,255          ;je-li pocitadlo mensi nez 255,skonci
            brlo RX_Complete2       ;(kvuli odfiltrovani sumu)

            ldi TlFree,0            ;vynuluj pocitadlo

            out SREG,SZal           ;obnov SREG
            set                     ;nastav T-bit

            reti                    ;skonci

;================== Handler dokonceni serioveho vysilani ==================

TX_Complete:mov CTS,C1              ;nastav flag dokonceni vysilani
            reti                    ;skonci
```

```
;================= Handler prijeti byte po seriove lince ==================

RX_Complete:in SZal,SREG            ;schovej stav SREG
            in DataIn,UDR           ;precti prijaty byte

            cpse ByteCount,C255     ;pokud je pocitadlo ruzne od 255
            inc ByteCount           ;inktementuj pocitadlo prijatych bytu

            cpi DataIn,13
            brlo RX_Complete2       ;kdyz je kod mimo rozsah,skonci

            cpi DataIn,91
            brsh RX_Complete2       ;kdyz je kod mimo rozsah,skonci

            cpi DataIn,36           ;kdyz je $,nuluj pozici v pameti
            breq RX_Complete4

            cpi DataIn,37           ;kdyz je %,nuluj pozici v pameti
            breq RX_Complete4

            cpi DataIn,64           ;kdyz je @,nuluj pozici v pameti
            breq RX_Complete4

            cpi ByteCount,255       ;kdyz je ByteCount 255, ignoruj
            breq RX_Complete2       ;prijem a skonci

            cpi ByteCount,2         ;kdyz je 2.byte,zkontroluj adresu
            breq RX_Complete5

            cpi ByteCount,3         ;kdyz je 3.byte,zkontroluj adresu
            breq RX_Complete6

RX_Complete1:cpi ZL,224
            brsh RX_Complete2       ;kdyz dosla pamet,skonci

            st Z+,DataIn            ;uloz data

            cpi DataIn,13
            breq RX_Complete3       ;je-li CR,nastav flag

RX_Complete2:out SREG,SZal          ;obnov SREG
            reti                    ;skonci

RX_Complete3:     out UCR,C0        ;zakaz prijimac i vysilac na 2313
            sbi PORTD,RE            ;prijimac RS-485 na Hi-Z
            ldi ByteCount,255       ;ignoruj dalsi prenos

            mov NewData,C1          ;nastav flag prijeti novych dat

            out SREG,SZal           ;obnov SREG
            reti                    ;skonci

RX_Complete4:ldi ZL,ZacData         ;Nastav ukazatel dat na zacatek
            ldi ByteCount,1         ;Nastav pocitadlo prijatych bytu na 1
            rjmp RX_Complete1       ;pokracuj

RX_Complete5:cp DataIn,Adr1         ;sedi-li adresa,
            breq RX_Complete1       ;pokracuj a uloz byte

            ldi ByteCount,255       ;jinak ignoruj prenos
            rjmp RX_Complete2       ;a skonci
```

```
RX_Complete6:cp DataIn,Adr2         ;sedi-li adresa,
            breq RX_Complete1       ;pokracuj a uloz byte

            ldi ByteCount,255       ;jinak ignoruj prenos
            rjmp RX_Complete2       ;a skonci

;====================== Handler preteceni casovace 0 ======================

Tim0_Ovf:   in SZal,SREG            ;schovej stav SREG
            wdr                     ;vynuluj watchdog

            inc Count               ;inkrementuj pocitadlo casovace 0

            cpi TlFree,255          ;je-li pocitadlo stisku tlacitka FF,
            breq Tim0_Ovf1          ;skonci

            inc TlFree              ;inkrementuj pocitadlo stisku tlacitka

Tim0_Ovf1:  out SREG,SZal           ;obnov SREG
            reti                    ;skonci

;====================== Handler preteceni casovace 1 ======================

Tim1_Ovf:   in SZal,SREG            ;schovej stav SREG

            inc CountP              ;inkrementuj pocitadlo casovace 1

            out SREG,SZal           ;obnov SREG
            reti                    ;skonci
```

6.3 Uživatelský manuál k automatické krmičce ryb

6.4 Výsledky experimentů automatického krmení ryb

6.5 Hodnocení automatické krmičky od odborníka na chov ryb

6.6 Obsah přiloženého CD-ROM

Přiložené CD-ROM obsahuje tuto práci ve formátu pdf včetně příloh (v adresáři „diplomka“), dále zdrojové kódy všech programů (adresář „zdrojaky“) a nakonec všechny datasheety součástek, zmiňovaných v textu (adresář „datasheety“).

Adresář „zdrojaky“ obsahuje následující podadresáře:

„krmicka“ – obsahuje zdrojový kód firmwaru krmičky v assembleru

„programator“ – obsahuje zdrojový kód programu k nahrávání firmwaru do krmičky v Borland Pascal

„ridici“ – obsahuje zdrojový kód provizorního řídícího programu pro 3 krmičky, který byl použit při testech (psáno v Borland C++).

„utilita“ – obsahuje pomocný program určený k testování krmičky, psaný v Borland C++.

Adresář „datasheety“ obsahuje následující podadresáře:

„Atmel“ – obsahuje datasheet procesoru AT90S2313

„Hitachi“ – obsahuje datasheet procesoru H8S2638F

„Linear“ – obsahuje datasheet budiče LTC485

„Thompson“ – obsahuje datasheet budiče L293D

„Advantech“ – obsahuje datasheety zařízení řady ADAM a PCA-6145L

„Gryf“ – obsahuje datasheety chemických senzorů Gryf

„Microspace“ – obsahuje datasheet průmyslového PC104 Microspace MSM586SEV