VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POLOAUTOMATICKÉ RC ŘÍZENÍ MODELŮ

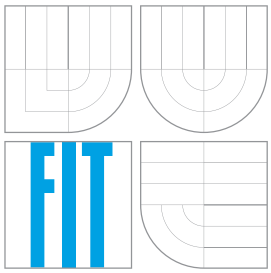
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN FILO

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POLOAUTOMATICKÉ RC ŘÍZENÍ MODELŮ

SEMI-AUTOMATIC CONTROL OF RC MODELS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN FILO

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Dr. Ing. PAVEL ZEMČÍK

BRNO 2010

Abstrakt

Bakalářská práce se zabývá problematikou automatizace ovládání RC modelů letadel. Řeší stabilizaci modelu ve fázi letu z jakékoliv prostorové orientace, přikládá sadu funkcí pro ovládání chování modelů. Přidává také popis jednoduché "podomácku vyrobené"gyroskopické jednotky určené pro testování chování modelu na zemi. Práce využívá předem připravené sady funkcí pro velmi základní ovládání desky s mikrokontrolerem MSP430F169.

Abstract

The Bachelor thesis describes problems connected with automatization of RC airplane models. It solves problem with stabilization during the flight from any space orientation. It attaches a set of functions to control the model behavior. There is also a brief description of easy "Do it yourself"gyrocopic unit aimed to testing the behavior of the model on the ground. The thesis uses provided set of functions for very basic control of board with microcontroller MSP430F169.

Klíčová slova

RC modely, automatizace, stabilizace, PWM, RS232, gyroskopická jednotka

Keywords

RC models, automatization, stabilization, PWM, RS232, gyroscopic unit

Citace

Martin Filo: Poloautomatické RC řízení modelů, bakalářská práce, Brno, FIT VUT v Brně, 2010

Poloautomatické RC řízení modelů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Dr. Ing. Pavla Zemčíka. Technickou pomoc při řešení hardwarových problémů mi zajistil David Král. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Filo
16. května 2010

Poděkování

Chtěl bych poděkovat svému vedoucímu panu doc. Dr. Ing. Pavlu Zemčíkovi za ochotu větší, než bych si zasloužil a Davidu Královi za nezměrnou pomoc při řešení problému s absencí gyroskopické jednotky.

Obsah

Kapitola 1

Úvod

Od konce 19. století, kdy bratři Wrightové zkonstruovali první opravdové letadlo, prodělal letecký průmysl vlivem technického rozvoje obrovský krok kupředu. Důkazem může být první přelet kanálu La Manche či první člověk ve vesmíru. Tento rozvoj byl nemalou měrou ovlivněn pokroky v oblasti výpočetní techniky.

V dnešní době není létání výsada jen nejbohatších lidí, ale umění pilotovat tyto stroje stále patří mezi schopnosti hrstky populace. Vynálezy radiových vln, prvního letadla, parního stroje a dalších přispěly ke vzniku nového odvětví vědy a zábavy, leteckému modelaření. Pořídit si funkční, radiovými vlnami řízený, model přijde naprostého laika jen na několik tisíc korun.

Firmy do světa chrlí značné množství nových technologií čímž se otevírají neustále nové možnosti vytvořit něco ojedinělého. Z pohledu letecké modelařiny lze vytvářet nové komponenty pro modely od stabilizačních systémů, letu dle GPS či dle předem naplánované trasy až po zcela autonomní modely, závislé pouze na stavu paliva v nádrži nebo síle akumulátoru. Je tedy možné vyrobit téměř dokonalou zmenšeninu dnešních velkých letadel.

Bakalářská práce se zabývá problematikou řízení radiově ovládaného modelu letadla pomocí mikrokontroleru z rodiny modelů typu MSP430 firmy Texas Instruments, které se hojně využívají ve vestavěných systémech kvůli své nízké spotřebě a vysokému výkonu. Čtenář postupně získá informace o fyzikálních vlastnostech letadla a základní komponenty pro ovládání směru. Následně bude informován o stavu současných technologií zabývajících se zautomatizováním ovládání modelů. Čtenáři, mající znalosti v oblasti letectví a modelařiny, mohou úvodní kapitolu přeskočit až na detailní popis implementace práce, kde je popsáno, jakým způsobem byl projekt řešen, testován, jakých výsledků bylo dosaženo a co bylo potřeba vyrobit.

Kapitola 2

Analýza současného stavu

Podíváme-li se na historii lidstva, zjistíme, že si lidé průběžně podmanili všechny živly. Jedním z posledních byl vzduch. Stavba létajícího stroje byla námětem mnoha bájí a legend.

Prvními průkopníky v oblasti letectví byli bezesporu Číňané. Ti objevili draka a nepřímo tak položili základy ke zkoumání objektů schopných letu. Prvním člověkem, který se zabýval konstrukcí létajícího stroje byl Leonardo da Vinci. Tento stroj byl inspirován letem ptáků a využíval mávání křídel k letu, začalo se mu tedy říkat Ornitoptéra, ale ke stavbě nikdy nedošlo. K premiéře letu člověka přispěli bratři Montgolfierové, kteří sestrojili první horkovzdušný balón. Let na dlouhé vzdálenosti umožnil až vynález německého inženýra Otto Lilenthala, který navrhl kluzák. Na přelomu 19. a 20. století uskutečnil svůj první let hrabě Ferdinand von Zeppelin se svojí vzducholodí. Díky všem těmto vynálezům byl dán základ pro dnešní podobu leteckého průmyslu.

Navzdory tomu má letecké modelářství poměrně mladou historii. O modelářských začátcích poměrně stručně informuje článek [8]. První záznam o radiově řízeném letu letadla se datuje na 30. 6. 1957, který provedl H. D. Talpin s rádiem řízeného modelu pohaněného elektromotorem[1]. V 70. letech minulého století došlo k zásadnímu rozvoji letecké modelařiny, protože byl na trh uveden rádiový ovladač s možností ovládání dvou směrů zároveň jedním prstem. Díky velmi pokročilému technickému rozvoji jsou dnešní ovládací soupravy na zcela jiné úrovni a umožňují ovládat téměř cokoli, co se v modelu letadla nachází.

2.1 Letadlo

Co to je letadlo? Norma ČSN 31 0001[6] uvádí: "*Zařízení způsobilé létat v atmosféře nezávisle na zemském povrchu, nést na palubě osoby nebo jiný náklad, je schopné bezpečného vzletu a přistání a je alespoň částečně řiditelné.*"Je to tedy stoj využívající aerodynamické a aerostatické síly ke svému pohybu. V bakalářské práci bude dále pojem *letadlo* vymezovat pouze letadla atmosferická. V článku na webu Českého rozhlasu[4] je možné nalézt jednoduše vysvětlený princip, na jakém jsou letadla postavena.

Aerodynamická síla

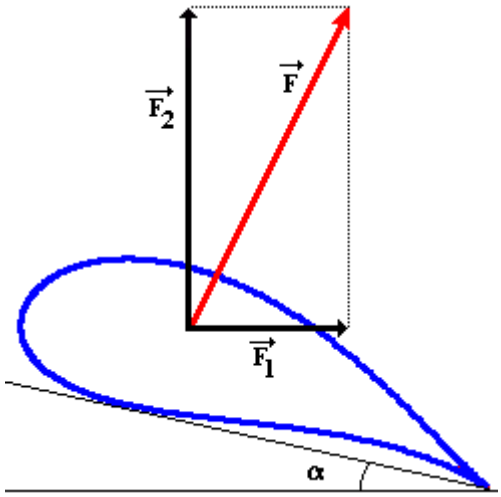
Aerodynamická síla[12], působící na letadlo, je složena ze sil vztlakové a odporové. Tato síla působí vždy, je-li plyn nebo objekt v plynu, v relativním pohybu. Díky této síle je umožněno letadlům těžším než vzduch létat.

[1] Viz http://www.kolmanl.info/index.php?show=LHISTORIE.

Aerostatická síla

Aerostatická síla je obdoba hydrostatické síly s tím rozdílem, že se nachází v prostoru vyplněném plyny. Velikost této síly je dána povrchem tělesa. Čím větší má těleso povrch, tím větší aerostatická síla na něj působí. Působí-li na objekty větší aerostatická síla než síla tíhová, objekty se v plynné soustavě vznášejí. Tohoto jevu využívají například neřiditelné balóny či řiditelné vzducholodě. Pro pochopení aerostatické síly můžeme nahlédnout i do článků o Archimédově zákoně[2].

Proto, aby letadla těžší než vzduch mohla létat, je třeba pohonná jednotka, která bude při pohybu letadla překonávat dynamický odpor a tím zajistí letadlu jeho let. Proto je potřeba konstruovat křídla tak, aby kladla co nejmenší odpor, ale zároveň aby umožňovala co možná největší vztlak.



Obrázek 2.1: Rozbor křídla, [5].

Na obrázku 2.1 je možné vidět odporovou sílu `F1`, působící proti pohybu letadla, vztlakovou sílu `F2`, která umožňuje let a výslednou dynamickou sílu `F`. Tažná síla motoru napomáhá k překonání odporové síly, z čehož plyne, že se konstruktéři snaží vytvořit křídlo s co možná nejmenším součinitelem odporu a největším součinitelem vztlaku. Úhel α se nazývá úhel náběhu, který hraje roli při zjišťování velikosti odporové síly. Křídlo má tzv. proudnicový tvar. Je to takový tvar, kdy vzduch proudící po horní straně křídla musí urazit větší vzdálenost než po straně dolní. To má za následek, že vzduch po horní straně křídla proudí rychleji, čímž roste aerodynamická síla působící na spodní stranu. Této síle říkáme aerodynamický vztlak. Podrobné konstrukční informace ke křídlu a zbytku letadla lze nalézt na webu Ústavu letectví [9].

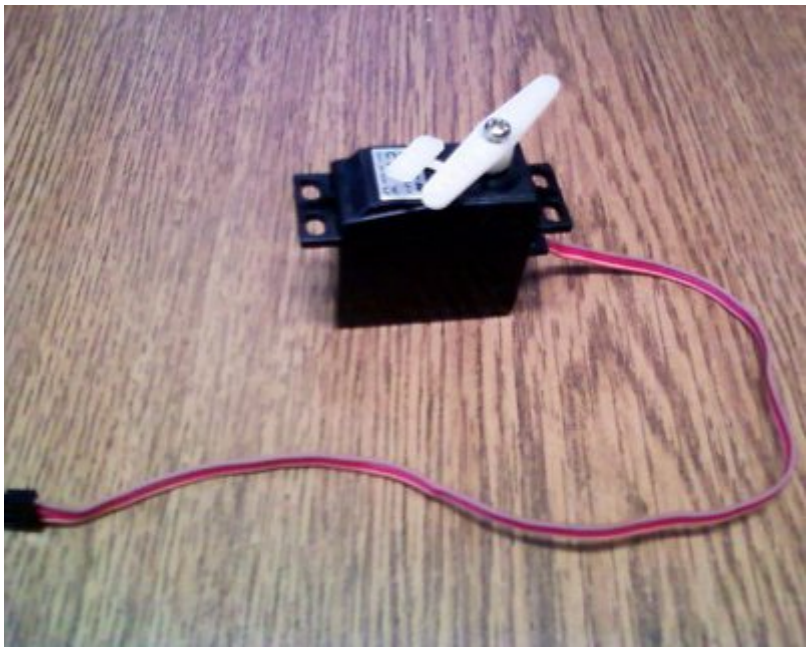
Aby se mohl řídit náklon letadla, jsou na odtokové hraně křídel umístěna křidélka. Jejich vychýlením dojde k náklonu letadla. Při změně směru vlevo nebo vpravo se křidélka natáčejí vždy na opačnou stranu vůči sobě. Při současném pootočení křidélek a směrovky na danou stranu dojde k zatočení letové dráhy.

V ocasní části letadla je umístěna výškovka a svislá ocasní plocha. Výškovka je zadní vodorovné křídlo složené ze dvou částí: stabilizátoru a pohyblivého výškového kormidla. Dojde-li k vychýlení výškového kormidla, letadlo změní dráhu letu kolem příčné osy. Směr vychýlení určuje, kterým směrem se letadlo bude ubírat dále. Svislá ocasní plocha, neboli

směrovka je složena také ze dvou částí: kýlovky a pohyblivého směrového kormidla. Směr vychýlení směrovky zajistí změnu dráhy letu kolem svislé osy. Tato osa je také příčná, ale aby nemohlo dojít k záměně, bude na ni nadále odkazováno jako na osu svislou.

2.2 Servo

Servo je malá mechanická část modelu, která nastavuje svoji polohu na základě délky řídícího signálu. Úhel otáčení serv je cca 90°. Jako vstupní signály se používají +5V, zem a řídící signál. Protože servo používá ke svému nastavení pulzně šířkovou modulaci (dále PWM) s frekvencí 50Hz, je pro řízení potřeba přesně sledovat délky signálů do serv posílaných, protože bezpečně pracují s řídícím signálem délky cca 1 - 2ms. Vycentrovanou polohu, tj. servo v pozici "uprostřed", doháhneme odesláním pulzu délky 1,5ms. Avšak toto pravidlo lze porušit, protože serva lze řídit i s pulzy v rozsahu 0,5 - 2,5ms. Pokud však k tomuto kroku přistoupíme, je nutné si tento fakt ověřit s dokumentací k použitému modelu serva, aby nedošlo k jeho nenávratnému poškození.



Obrázek 2.2: Servo

Připojení serva k přijímači nebo jakémukoliv jinému zařízení je poměrně jednoduché. Musíme mít na paměti, že tmavší z krajních kabelů (obvykle černý nebo hnědý) značí zem, prostřední je vždy +5V a druhý krajní je řídící signál.

2.3 Mikročip MSP430

Mikročip použitý v řídící jednotce je typu MSP430F169[11]. Je předurčen k řízení systémů, kde je jedno z hlavních kritérií co možná nejmenší spotřeba. Mezi hlavní výhody patří:

- flash paměť o velikosti 55KB

- šest 8bitových vstupně/výstupních portů
- dva USART moduly pro rozhraní SPI, UART nebo I^2C
- jeden Analogově digitální a jeden digitálně analogový převodník

Mikročip má integrován spousty dalších komponent, které je možné najít v dokumentaci [10]. Aby bylo možné pracovat s tolika moduly, jednotlivé piny se sdílejí mezi komponentami. Na jednu stranu je to výhoda pro výrobce, ale programátor musí celou dobu pečlivě hlídat, jaký výstup se na jednotlivých pinech nachází.

2.4 Možnosti automatizace - systémy třetích stran

S příchodem výkonných mikročipů s nízkou spotřebou se i v leteckém modelářství projevuje snaha o řízení počítačem. Cenově přijatelná je nyní i technologie GPS, tudíž se otevírají dveře také pro navádění pomocí družic. Výroba kitů pro automatizované ovládaní spadá jak do sféry komerční, kdy je možné nalézt na trhu několik řešení, tak do sféry modelových nadšenců. Výhody komerčních řešení jsou především v odladěnosti řídící aplikace a vyladění firmwaru, na jehož tvorbě se podílejí profesionálové. Značnou nevýhodou tohoto řešení je vyšší cena a nedostupnost na českém trhu, kdy je třeba spoléhat na dovoz ze zahraničí. Na druhou stranu výhody amatérských řešení jsou v dostupnosti součástek a návodů, jak si kit připravit doma. To značně sníží pořizovací náklady spojené s výrobou vlastního zařízení. Naopak nevýhoda těchto řešení spočívá v takřka nulové podpoře k zařízení, nutnosti ladit software na bázi bádání ve zdrojových kódech a jejich úpravách.

2.4.1 Kestrel Autopilot

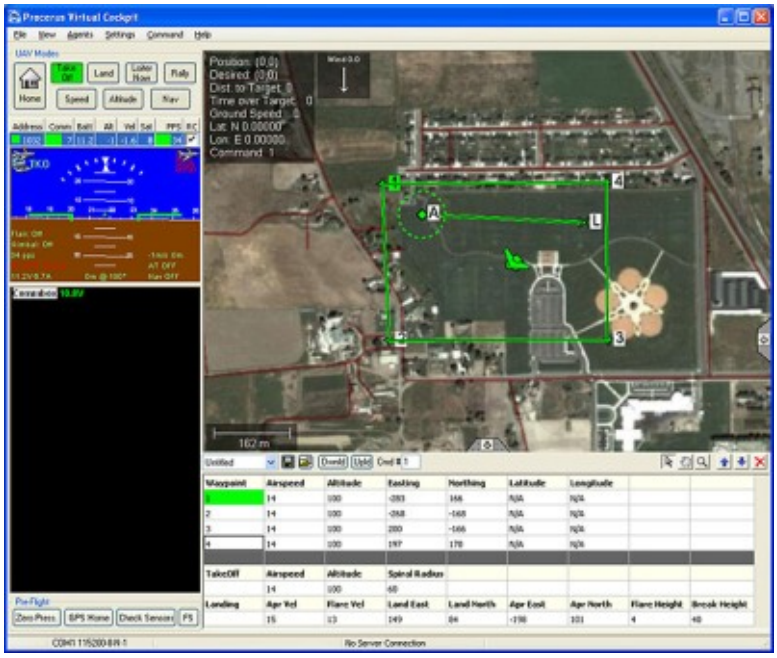
Autopilot[7] je prezentován jako nejlehčí a nejmenší. S vahou 16,7 gramů se výrobci podařilo zabudovat všechny potřebné komponenty, které by zajistily zcela autonomní let modelu. Je možné nalézt rychloměr, výškoměr, upravený GPS modul určený nejen pro let dle předem zadaných bodů, ale také pro další pomocné výpočty.

Výhody tohoto řešení jsou, dle výrobce, propracovaný systém navádění modelu a jednoduché připojení dílčích komponent. Pomocí kontrolních bodů, které se zadají do programu ve formě GPS souřadnic, je možné donutit model opsat předem určenou dráhu, automatický start i přistání. Ovládání zajišťuje dodaný software Virtual Cockpit^{TM}, který umožňuje jak zcela autonomní let, tak dynamicky reaguje na příkazy dané člověkem i ve fázi letu.

Virtual Cockpit^{TM}je řídící jednotka, která se stará o řízení modelu ze země. V této aplikaci je možné předplánovat let, upravovat telemetrii za letu či v reálném čase zobrazovat průběh letu. Pro úpravy letu může posloužit i gamepad. Aplikace bezdrátově komunikuje s modelem oběma směry, kdy časový rozdíl aktuálnosti dat lze počítat v řádu milisekund. Toto řešení skýtá další výhody v možnosti řízení několika modelů současně. Tímto lze například dosáhnout následování předem nastaveného bodu, i za předpokladu, že je tento bod v pohybu.

K řídícímu softwaru Virtual Cockpit^{TM}lze připojit také OnPoint^{TM}Targeting aplikaci. Tato aplikace přidává podporu pro sledování pohyblivého bodu, vyhledání předem daného bodu pomocí rozpoznávání obrazu, nahrávání obrazu a mnoho dalších funkcí.

Následování objektů je řešeno pomocí rozpoznávání obrazu, kdy člověk, sedící u řídící aplikace, sleduje video přenos v reálném čase a pomocí ukazatele myši zvolí objekt k následování. Toto dává velký prostor pro využití nejen v armádní oblasti, ale také lze vytvořit



Obrázek 2.3: Procerus Virtual Cockpit^{TM}.

jednoduchý bezpečnostní či průzkumný systém. Přesnost tohoto sledování lze vyčíslit na 3-5m.

Mód pro stabilizování video přenosu zajišťuje větší přehled o aktuálním dění kolem modelu tak, že se video přenos pozastaví a člověk je schopen snadno předávat řídící příkazy, kdy není limitován okamžitou reakcí na obraz. Může tedy na chvilku prostudovat danou lokaci a mít tak lepší přehled nad situací. Tento mód lze přirovnat ke získání obrazu podobného helikoptéře, která momentálně drží pozici. Následně OnPoint^{TM}Targeting systém umožňuje navedení modelu na bod, který byl označen ve stabilizovaném obraze. To přináší velké výhody, kdy není třeba znát aktuální polohu letadla a aplikace sama zajistí, aby se model navrátil.

Průzkumný mód umožňuje sledování daného cíle z větší vzálenosti tak, že uživatel zadá bod, fixní nebo pohyblivý, a řídící jednotka sama vypočíta potřebnou letovou hladinu a trajektorii, aby záběr z kamery umístěné v modelu byl vždy nebo co možná největší možnou dobu v centru snímače. Aby byla trajektorie nejlepší možná, je vypočítána z mapových podkladů a výskové mapy terénu, které jsou dodány v aplikaci.

Aby byl vývoj a implementace řešení pro uživatele přívětivější, je k dispozici Test Platform. Jde o malé létající křídlo specielně určené pro testování vlastních řídících algoritmů a také slouží jako jednoduchá výuková platforma pro méně zdatné uživatele v oboru automatizace. Na křídlo lze také umístit malou kulovou kameru pro testování průzkumného módu.

Jak je patrné, tento systém nabízí mnoho funkcí. Jednotlivé součásti jsou určeny pro specifický účel a není proto nutné vlastnit veškeré komponenty k zajištění funkčnosti autopilota. Protože jde o komerční řešení určené pro širokou škálu zákazníků, není možné si software upravovat, avšak vývojové oddělení je vstřícné pro další případné potřeby zákazníka. Celkově se toto řešení podepsalo také na ceně, která za komponenty Kestrel^{TM}Autopilot

a Virtual Cockpit^{TM}s pozemní komunikační jednotkou Commbox^{TM}činí 8495$[2]. Tato cena je pro běžného pracujícího modeláře téměř nedostupná, avšak za řešení, které nabízí je očekávatelná.

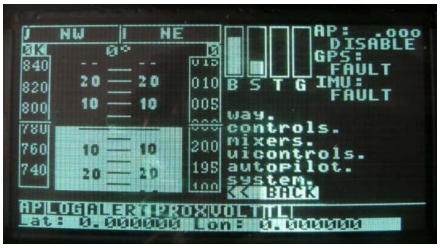
2.4.2 Freespace EZStar

Freespace EZStart je projekt[1] modelářského nadšence, který si ve volném čase vytvořil zcela autonomní model letadla s veškerými řídícími a komunikačními jednotkami.

EZStar[3] patří svou konstrukcí do kategorie modelů pro modeláře začátečníky, takže je ve vzduchu mnohem lépe ovladatelný, avšak na úkor akrobacie. Můžeme ho částečně zařadit i do kategorie větroňů díky velikosti nosných ploch křídel. Protože je model vyroben z poměrně odolného materiálu, vydrží i tvrdší přistání. Všechny tyto vlastnosti jej předurčují ke stavbě jednoduchého autopilota, který by zařídil zcela samostatný let.

Aby bylo možné z takového modelu vytvořit zcela samořízený stoj, byl vyměněn přijímač a nahrazen za transciever The Freespace RF modem[4], jehož autorem je právě Terence J. Bordelon. Oproti klasickým levným přijímačům, které zvládnou pouze předat řídící signál na výstupy, je tento vybaven možností komunikace s pozemní jednotkou. Dosah komunikace činí něco přes 100km s přenosovou rychlostí 500kbps. Tato přenosová rychlost teoreticky umožňuje přenos živého videa, otázkou zůstává, zda-li je kvalita dostačující pro let z vyšších letových hladin. Modem obsahuje 5 PWM výstupů pro ovládání serv, možnost automaticky změnit komunikační kanál, je-li zjištěno rušení a možnost připojení USB zařízení, jako minikamery, flash disky apod.

Samotným jádrem systému je Inertial Measurement Unit[5], tedy vnitřní měřící jednotka, která získává telemetrii a následně předává výšku a pozici. Je to také "mozek"letadla, proto kopírování předem stanovené trasy, start i přistání je implementováno právě v této jednotce. Pokud rozebereme řídící jednotku na jednotlivé součásti, můžeme zjistit, že obsahuje několik gyroskopických jednotek, magnetometr, akcelerometr a GPS modul.



Obrázek 2.4: Ground Control Station.

Pro sledování aktuální telemetrie ze země je potřeba vlastnit řídící centrum, v tomto projektu Ground Control Station. Výhoda pozemní jednotky tkví v kompletní samostatnosti, tj. není vázaná na PC, ačkoli ji lze s počítačem propojit. Tato skutečnost, byť na první pohled nepodstatná, má jednu velkou výhodu. Když dojde k "zatuhnutí"počítače, letadlo se nezřítí. Za předpokladu, že "zatuhne"řídící stanice, je potřeba zajistit, aby byla stanice schopna sama sebe restartovat a uvést do provozuschopného stavu za méně než 100ms.

[2] Ke dni 27.4.2010, což činí cca 162 578Kč

[3] Rádiem řízené letadlo, viz: http://www.hobby-lobby.com/easystar_7008_prd1.htm.

[4] http://bordelon.net/modem.html

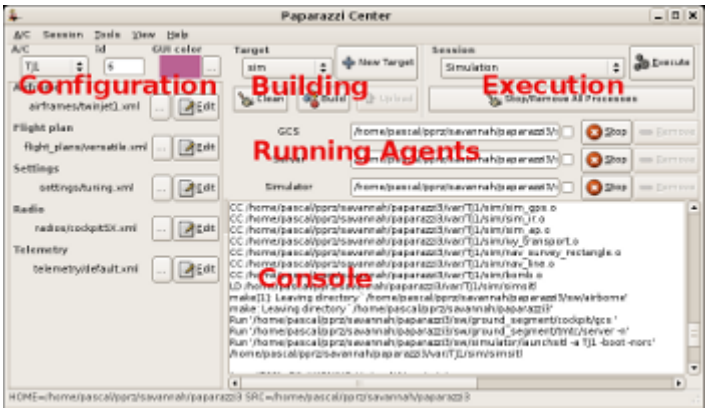
[5] http://bordelon.net/freespaceimu.html

Plánování letu je pořeba provést před vlastním startem, pozemní stanice není schopna měnit letové body bez připojeného PC. Lze na ni vypínat/zapínat autopilota, předprogramované módy či drobné úpravy letových vlastností.

Tento systém umožňuje zautomatizovat modely na pár fukncí, jako následovat letové body, stabilizovat model, umožnit jeho automatický návrat na výchozí pozici. Další nezměrnou výhodou je použití běžně dostupných součástek. Prozatím není tento systém k dispozici na trhu, ale autor je ochoten o systému poskytnout informace pro případné stavitele. Řešení je tedy ukázkou, že lze sestavit autonomní letadlo, avšak nevýhodou je nutnost kompletního naprogramování řídících algoritmů.

2.4.3 Paparazzi Project

Automatizováním modelů letadel se zabývají i další univerzity. Na francouzské univerzitě ENAC je ve vývoji open-source projekt[3] zaměřen na výrobu otevřeného autopilota. Projekt se zaměřuje na všechny aspekty automatizace, počínaje řídícím hardwarem v letadle, přes komunikační stanice, konče jednoduchým ovládacím softwarem. Narozdíl od *Freespace EZStar* 2.4.2 se pozemní stanice spojuje s PC, ve kterém se nachazí veškeré řídící algoritmy. Telemetrie získaná ve vzduchu je okamžitě poslána na pozemní stanici do tzv. komunikačního agenta. Ten ji následně posílá dál pomocí síťového protokolu. Je tedy možné připojit jedenu pracovní stanici, lokální síť nebo internet. Tato data je možné následně zobrazovat buď jako zpravy, čili je možné sledovat kompletní telemetrii v reálném čase nebo zachytávat serverovou částí, která je schopna data logovat, vyhodnocovat a předávat dále pozemní řídící stanici. Všechny tyto moduly jsou řiditelné z grafického výstupu Paparazzi Center.



Obrázek 2.5: Paparazzi Center.

Díky tomu, že je projekt vyvíjen několik let, není nutné vlastnit modely, které nejsou tolik náchylné k vnějším vlivům. Teoreticky je možné létat s jakýmkoliv typem modelu, což dokazuje i galerie uživatelů[6].

Samotná řídící jednotka je založena na mikrokontrolerech Atmel AVR nebo Philips ARM7 LPC. Plošné spoje obsahují jeden nebo dva mikrokontrolery a veškerou potřebnou elektroniku pro řízení serv, RC přijímač, modem, senzory, atd.

[6] http://paparazzi.enac.fr/wiki/Gallery

Základem senzorů určujících polohu modelu jsou infračervené tepelné senzory uložené do rohů čtvercové desky a navzájem kolmé. To by mělo zajistit určení polohy vůči zemi a obloze. Problém může nastat, když letadlo vletí do většího mraku nebo mlhy, kde se nedá určit aktuální poloha, protože okolí má stejnou teplotu. Komunikaci s pozemní stanicí zajišťuje radiomodem, který je hlavní komunikační jednotkou. Avšak pro záchranu stále zůstává aktivní kanál z RC vysílače.

Výsledkem několikaleté usilovné práce se podařilo vytvořit open-source autopilota, který by měl být uživatelsky přívětivý a jednoduchý na ovládání. Protože jsou k dispozici značně rozsáhlé popisy jednotlivých komponent a jsou k dispozici i veškeré návrhy obvodů, činí toto řešení poměrně snadno použitelným. Systém disponuje funkcemi jako vylepšenou stabilitu modelu, autonomní navigaci pomocí kontrolních bodů, segmentovou či kruhovou navigaci, udržování letové hladiny aj. Dalšími funkcemi jsou například přenos kompletní telemetrie do pozemní stanice a zpětné řízení pomocí pozemní stanice. Veškerý software je napsán v jazyce C a XML. Nevýhodou tohoto řešení může být vázání na operační systém GNU/Linux a distribuce Debian, pro kterou je software laděn. Autoři ale dávají k dispozici i LiveCD se všemi potřebnými nástroji.

2.4.4 Shrnutí

V předchozí sekci bylo naznačeno několik řešení třetích stran. Je jisté, že komerční řešení *Kestrel Autopilot* 2.4.1 patří k jednomu z nejvyspělejších a nejkomplexnějších řešení. Pro složitější projekty, kdy je potřeba zajistit složité úkoly jako sledování pohyblivého cíle, průzkum terénu či podobné je systém více než vhodný, ale zákazník si musí nachystat značný finanční obnos.

Naproti tomu řešení tvořené jediným člověkem *Freespace EZStar* 2.4.2 přináší velmi dobře vyladěného autopilota pro základní úkony a promítne se ve velmi jednoduchém ovládání, avšak je zapotřebí dráhu letu určit již před startem, protože do něj nelze následně zasahovat. Velikou nevýhodu může činit takřka nulová podpora autora jak z pohledu hardwaru, tak z pohledu softwaru, ačkoli je možné se s autorem spojit a tyto možnosti prodiskutovat.

Jako zlatá střední cesta se může jevit open-source projekt *Paparazzi* 2.4.3 vyvíjený na francouzské univerzitě letectví. Jejich řešení nabízí kompletní podporu pro software i hardware ze strany autorů. Drobnou vadou může být závislost na počítači a operačním systému GNU/Linux. Pro vytváření trajektorie letu musí člověk disponovat znalostmi programování v jazyce C a XML.

Kapitola 3

Automatizace řízení a použité nástroje

Aby bylo možné s modelem létat, musíme počítat se spoustou vnějších vlivů. Mezi největší potíže, se kterými se můžeme setkat, lze zařadit překážky v letové dráze (např. stromy, budova vedle letiště, vedení vysokého napětí, aj.), konstantní vítr, náhlé poryvy větru, turbulence, špatný úsudek pilota, aj. Spousty z nich lze eliminovat pomocí řídící jednotky, která by zajišťovala správné rozhodnutí na základě předem stanovených pravidel. Avšak aby byla schopna jednotka takto reagovat, musí mít k dispozici přesné údaje o poloze a telemetrii modelu. Mezi absolutně nezbytné údaje, které musí mít jednotka k dispozici pro vyhodnocení situace, patří aktuální poloha modelu v prostoru, tj. celková polohová orientace vůči zemi, rychlost, zrychlení, zeměpisná poloha, letová hladina. Díky těmto informacím lze sestavit zcela autonomní letadlo, které provede předem zadané úkony nebo se bude, na základě rozhodujících algoritmů, samo řídit.

Musí být zajištěna rychlost zpracování vstupních signálů, aby byl autopilot schopen co nejrychleji reagovat na nepředvídatelnou situaci. Se zpracováním vstupních signálů úzce souvisí datový přenos mezi pozemní stanicí a řídící jednotkou, který je narušen vnějšími signály. Toto rušení se v praxi nejčastěji odstaňuje pomocí *Kalmanova filtru*. V současné době je ale projekt teprve v začátcích a problémy spojené s bezdrátovou komunikací budou řešeny až v dalších fázích vývoje.

3.1 Dostupné nástroje pro vývoj

MSP-FET430UIF

Pro vývoj byl použit emulátor MSP-FET430UIF[1], který nabízí firma Texas Instruments. Je to jednotka připojitelná k počítači pomocí rozhraní USB na jejímž výstupu je rozhraní JTAG, určené pro programování mikrokontroleru a pohodlné ladění programů metodou step-by-step, brakepointů apod.

Prvním problémem, který je třeba vyřešit, je nainstalování ovladačů. Firma Texas Instruments žádné ovladače nedodává. Ačkoli není jasné proč, je třeba sáhnout po nástrojích třetích stran. Např. s vývojovým prostředím IAR Embedded Workbench 3.1 jsou ovladače dodány.

[1] http://focus.ti.com/docs/toolsw/folders/print/msp-fet430uif.html

mspgcc

Open-source komunita vytváří nástroj, založený na kompilátoru gcc, pro vývoj na platformu MSP430. Jejím obsahem jsou překladače z různých programovacích jazyků, proxy server, který je nutný pro spojení s emulátorem, vlastní rozhraní pro emulátor, debugger apod. Výhodou je, že výsledný kód není nijak omezen, to je však kompenzováno ne zcela pohodlným ovládáním, nutností změnit firmware emulátoru na dodávaný a, pro nepřátele příkazové řádky ve Windows, nutnosti práce s ní. Bohužel problémy s během na stanici s nainstalovaným systémem Windows 7 mě donutili porozhlédnout se jinde.

Code Composer Studio

Momentálně je k dispozici nástroj pro programování přímo od výrobce, Code Composer Studio v4.x[2]. Fakulta Informačních Technologií disponuje pouze verzí 2.0 z roku 2006, která, bohužel, s emulátorem nekomunikuje. Důvodem je pravděpodobně příliš nový firmware. Toto studio je k dispozici v několika verzích, od free, která je omezena na velikost výsledného kódu až po placenou verzi s cenou od 445$. Nesmyslnost v cenové politice je až zarážející.

Bohužel Code Composer Studio ve své volné verzi nedisponuje potřebnými ovladači pro emulátor a také se vyskytují chyby při objednávce stažení volně dostupného studia.

IAR Embedded Workbench

Asi nejlepším nástrojem pro vývoj a ladění programů pro mikrokontrolery z rodiny MSP430 je IAR Embedded Workbench[3]. Při instalaci software jsou dodávány veškeré potřebné ovladače. Avšak cenová politika je opět podobná té od Texas Instruments. Základní nástroj pro vývoj, omezený na 4KB, případně 8KB[4] zdrojového kódu je zdarma, za neomezenou verzi je však nutné si zaplatit.

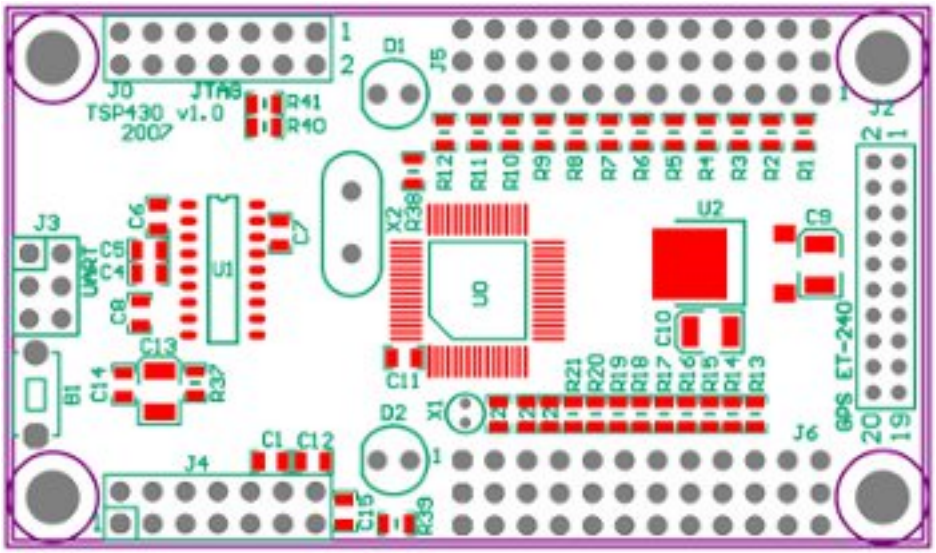
Práce s tímto prostředím je poměrně jednoduchá. Jediným problémem v začátku je nutnost nastavit výsledný mikrokontroler, pro který má být program optimalizován. Ač disponuje autodetekcí připojeného zařízení, je potřeba jej zvolit v možnostech projektu ručně. Spouštění a ladění progamu probíhá, když má člověk zrovna štěstí, bez problémů. Velmi často se vyskytne chyba při komunikaci mezi prostředím a C-SPY jednotkou, která je obsažena v emulátoru a která umožňuje ladění právě pomocí breakpointů, sledování hodnot proměnných apod. Pokud k této situaci dojde, stačí emulátor odpojit z USB portu a zase jej připojit. Dalším problémem pak často bývá neschopnost detekce cílového zařízení. Opětovné odpojení a připojení emulátoru problém vyřeší. Bohužel tyto chyby se vyskytovaly v cca jednom ze tří pokusů o spuštění programu. Také nezřídka dochází k pádu celé aplikace a proto je nutné ji neustále spouštět. Na druhou stranu to nemusí být úplně chyba výrobce, protože jsem prostředí nezkoušel na rozdílných stanicích, je-li chování stejné. Možné omezení také může nastat v omezeném množství breakpointů, které sonda C-SPY při ladění zvládá, ale na běžné ladění je jejich počet zcela dostatečný.

Celkově toto prostředí nabízí nepřeberné množství užitečných funkcí a podporovaných mikrokontrolerů, přičemž si zachovává jednoduchost použití. Z tohoto důvodu jsem jej zvolil pro řešení bakalářské práce.

[2] http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html
[3] http://www.iar.com/website1/1.0.1.0/675/1/
[4] http://focus.ti.com/docs/toolsw/folders/print/iar-kickstart.html

3.1.1 Deska s řídící jednotkou

Na obrázku 3.1 můžeme vidět horní stranu plošného spoje se zapojením všech komponent. Protože na spodní straně je pouze několik rezistorů, není třeba ji přikládat. Pro programování je určeno rozhraní JTAG. Ačkoli jde o standard definovaný normou IEEE 1149.1[5], jeho zapojení na plošných spojích si každý výrobce vytváří po svém[6]. Proto není možné použít rozhraní jiného výrobce. Dodaný emulátor pracuje s mikrokontrolerem poměrně spolehlivě, přestože se občasným chybám nelze vyhnout.



Obrázek 3.1: Schéme řídící jednotky

Konektory `J5` jsou určeny pro získání PWM signálů z různých vstupů, nejčastěji přijímače. Olivier Jossoud, který pracoval na tomto projektu přede mnou, zkoušel připojit i gyroskopickou jednotku, ale z důvodů špatné přesnosti a nízkého kmitočtu (50Hz, signál přijde jednou za 20ms) nelze model řídit pomocí takto zapojené jednotky. Z celkových dvanácti vstupů je pro měření PWM vstupů určeno pouze prvních devět, přičemž vstup 2 není zapojen. Připojení napájení je stejné jako u serv, z čelního pohledu na konektor je spodní řídící signál, uprostřed +5V, nahoře zem (dále GND).

Oproti tomu konektory `J6` jsou zapojeny stejně, ale slouží pro generování výstupních PWM signálů a připojení serv, pohonné jednotky a dalších zařízení. Zde je na výstup určeno pouze 6 kanálů, kde je generován PWM signál.

Konektory `J3`, v programu označován také jako *UART0*, slouží k připojení sériové linky. V původním programu sloužilo jako testovací rozhraní pro jednosměrnou komunikaci mezi počítačem a výstupem na serva. Protože tato komunikace je bezpředmětná, byla v rámci bakalářské práce kompletně přepsána pro připojení testovací gyroskopicé jednotky.

Konektor pro připojení GPS modulu je obdobou sériového portu `J3`. Přestože MSP430F169 má zabudovanou podporu pro dva sériové porty, je druhý připojen právě do GPS modulu, v programu označovaného jako *UART1*. GPS modul jsem, bohužel, neměl k dispozici, proto nedošlo k žádné úpravě.

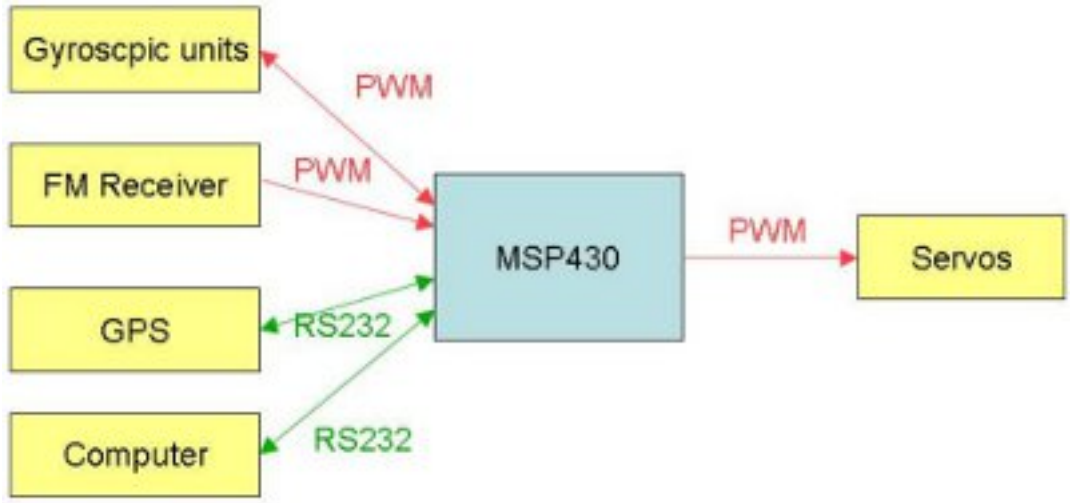
[5] http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-1990_desc.html

[6] http://www.jtagtest.com/jtag-standards

Rozhraní J4 je analogový vstup/výstup a na desce není vyveden konektor.

Filosofie připojení jednotlivých komponent

V původním návrhu3.2 bylo určeno, že se jednotlivé komponenty, ze kterých je třeba brát vstupy, připojí přes PWM, mimo dat z GPS modulu. Avšak, jak již bylo zmíněno výše, gyroskopická jednotka nepracuje přesně a je k dispozici málo údajů pro rozhodnutí, v jaké poloze se model právě nachází.



Obrázek 3.2: Návrh komunikace mezi jednotlivými moduly před zahájením bakalářské práce - autor: Olivier Jossoud.

Protože je sériové rozhraní poměrně rychlé a pracuje asynchronně vůči mikrokontroleru, je při obdržení dat vyvoláno přerušení. Původní modul *UART0*, který byl určen pouze pro testování výstupu na serva, doznal změn, aby bylo možné pomocí sériového rozhraní připojit jakékoliv zařízení.

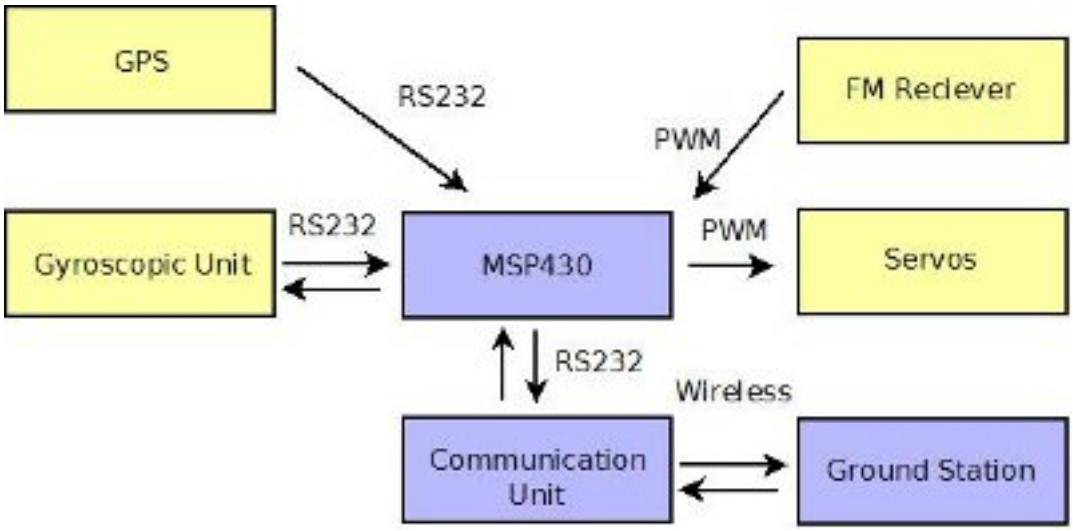
Generování PWM signálů na výstup zajišťuje DMA modul obsažený v mikrokontroleru. Tento modul podporuje přímý zápis bitů na jednotlivé nožičky mikrokontroleru, proto je generování PWM signálů řešeno tímto způsobem elegantněji, než za použití vnitřních hodin, které jsou ale sdílené i s blokem pro měření vstupních PWM signálů. Za použití DMA můžeme adresovat celý jeden bajt, což poslouží ke generování osmi na sobě nezávislých PWM signálů.

Připojování jednotlivých zařízení pomocí PWM mi nepřijde jako rozumný nápad, nejde-li o řídící signály z přijímače. Původní návrh také nepočítá s pozemní stanicí, která by byla záchytným bodem a jedním z nejdůležitějších prvků celého projektu. Projekt se nyní může vydat dvěma směry:

1. Vytvořit zcela autonomní model, který nebude komunikovat s pozemní stanicí a bude se řídit předem danou trasou, či vlastní inteligencí.

2. Připravit pozemní stanici a navrhnout komunikační moduly a protokol pro obousměrnou komunikaci, kdy bude možné model řídit ze země.

Výhody druhé možnosti tkví především v možnosti dynamicky měnit dráhu letu, přizpůsobovat model novým podmínkám či zachytávat obraz v reálném čase, kdy odpadá nutnost připojit do ovládácí jednotky letadla paměťové médium. Proto by, oproti původnímu návrhu, mohl projekt doznat drobných změn, které by se v již hotové programové části daly naimplementovat poměrně snadno.



Obrázek 3.3: Návrh komunikace mezi jednotlivými moduly.

Obrázek 3.3 znázorňuje možné připojení jednotlivých komponent po zakomponování komunikačního modulu. V tomto řešení by se modulům *UART0* a *UART1* změnila funkce tak, že jeden by se staral pouze o připojení komunikační jednotky a druhý měl na starosti všechny připojené komponenty. Protože je sériové rozhraní poměrně rychlé, vztaženo k rychlosti reakce serv, může být tato linka nasdílena mezi několik zařízení, aniž by byla nějak omezena funkce řídící jednotky a ohrožena bezpečnost modelu.

3.2 Programové bloky

Pro vytváření základních algoritmů pro ovládání modelů bylo navázáno na již existující řešení, které vytvořil francouzský student Olivier Jossoud pod dohledem doc. Dr. Ing. Pavla Zemčíka. Tento student vytvořil kostru pro automatizaci ovládání RC modelů na dodaném kitu. Již existující dokumentace popisuje, bohužel, řídící funkce pouze zevrubně a některé algoritmy se při studiu zdrojového kódu mohou zdát chaotické až nečitelné, mimo jiné díky chybějícím komentářům.

V původním stavu zůstaly moduly *UART1, capturePWM, flash, clock* a *calcCMD*. Naopak byl přidán modul *flightcontrol*, ve kterém jsou obsaženy všechny funkce pro zautomatizování některých činností modelu. Krom posledně zmíněného je potřeba před použitím daného modulu spustit inicializační funkce, které nastaví podporu pro daný modul v mikrokontroleru.

Pro bakalářskou práci byly použity moduly *UART0, DMA* a *flightcontrol*, které disponují všemi potřebnými funkcemi k dosažení alespoň částečné automatizace. Dále byl, pro potřeby testovací gyroskopické jednotky, změněn komunikační protokol po sériovém rozhraní.

Kapitola 4

Implementace

Cílem implementace bylo vytvoření předletových rutin, kdy je ověřena správnost chodu jednotlivých mechanických částí modelu, vytvoření stabilizačního zařízení, které by model, z jakékoliv pozice v prostoru, dokázalo uvést do vodorovné pozice kabinou směrem k nebi, rutin pro zajištění letu na dané letové hladině a s tím spojené pomocné funkce a komunikace mezi komponentami.

Programování mikrokontrolerů s sebou nese několik specifických problémů, jedním z nich bylo zprovoznění komunikace mezi vývojovým prostředím a mikrokontrolerem. Momentálně lze doporučit postarší operační systém Windows XP, kde se nenaráží na neustálé problémy s kompatibilitou ovladačů. Také je potřeba počítat s dosti omezenou možností hledání řešení problémů na internetu.

Aby bylo možné vytvořit funkci pro stabilizaci modelu, bylo potřeba použít gyroskopickou jednotku. Poněvadž v osobním vlastnictví žádnou takovou jednotku nemám a pořízení nové, která by měla digitální výstup, je několikatisícová položka, byl jsem nucen vyřešit problém pomocí dostupných zařízení.

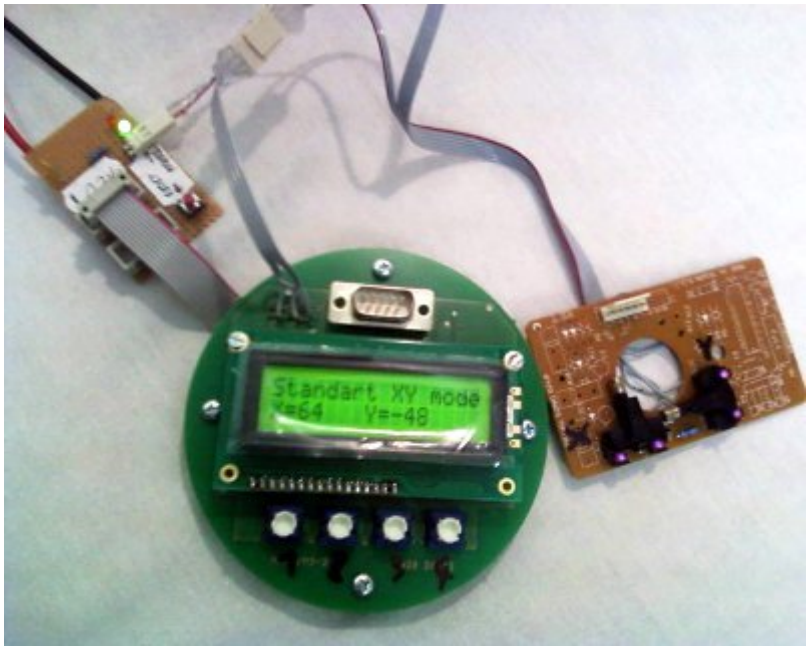
4.1 Testovací digitální gyroskop

Sestavit z běžně dostupných věcí gyroskop, který by se dal zabudovat do modelu letadla je úkol téměř nadlidský. Zbývaly mi dvě možnosti:

1. Vytvořit softwarový simulátor gyroskopu.
2. Vytvořit vlastní jednotku.

Vydal jsem se druhou cestou a za použití staré kuličkové myši, ovládací jednotky pro měření průtoku kapalin v potrubí a rozhraní RS232 byla vyrobena gyroskopická jednotka 4.1. V pravém slova smyslu se o gyroskopickou jednotku nejedná, protože nikde nerotuje setrvačník v inerciální poloze. Využívá se pouze snímačů pohybu mechanické části myši po vyjmutí kuličky. Z plošného spoje v myši nakonec zbyly pouze snímače pohybu a jeden rezistor, poněvadž zbytek elektronického vybavení nebyl potřeba.

Samotná deska však nedostačuje pro ladění programu, mimo jiné proto, že není okamžitě vidět, jak velký je úhel na výstupu. Proto bylo zařízení připojeno na jednotku s LCD displayem, která zobrazuje aktuální velikost náklonu a umužňuje tak přesně sledovat aktuálně nastavený úhel a jeho případné změny. Rozsah otáčení v každém směru je $< -359; +359 >$ stupňů. Při překročení této hranice je model v původním stavu a proto je čítač vynulován.



Obrázek 4.1: Testovací gyroskopická jednotka.

Tato pomocná jednotka je založena na mikrokontroleru ATMEGA8-16 a disponuje rozhraním RS232. Důležitou vlastností tohoto rozhraní je, že je překříženo již na plošném spoji, proto je potřeba komunikovat s ostatními zařízeními pomocí sériového kabelu s poměrem 1:1. Taktéž disponuje čtyřmi tlačítky, ze kterých jsou naprogramovány pouze dvě. Tlačítko č. 1 přepíná jednotku mezi testovacím a programovým módem. Rozdíl v nich je ten, že v testovacím módu je rozlišen pohyb snímačů pouze jako střídání 1 a 0. Jsou také rozlišeny všechny čtyři pohybové snímače na desce myši. V programovém je pak zobrazován aktuální úhel náklonu v osách X a Y. Tlačítko č. 4 slouží k ručnímu vynulování úhlů.

Pro napájení je potřeba, stejně jako u serv, +5V. Jediný rozdíl je v tom, že je potřeba použít nezávislý zdroj napětí. Je to způsobeno napájením displaye. Pokud je napájen spoj s řídící jednotkou MSP430F169 a testovacího gyroskopu naráz a je vyslán řídící signál na servo, dojde ke krátkému poklesu napětí a tím je, bohužel, gyroskop zresetován. I přes tuto nepříjemnou vlastnost jsou veškeré podmínky pro testování aplikace splněny.

Pro zachovaní jednoduchosti komunikace, gyroskop nepoužívá žádné kontrolní součty při přenosu dat a neposílá žádné potvrzení o příjmu příkazu. V opačném směru zase nečeká žádné potvrzení příjmu zaslaných dat či kontrolu jejich správnosti. Pro reálné použití je proto toto zařízení naprosto nepoužitelné, avšak pro ladící účely je více než postačující.

V tabulce 4.1 je přehled všech příkazů, které testovací gyroskop podporuje. Z důvodů zachování srozumitelnosti jsou poznámky psány v anglickém jazyce. Přenosová rychlost 9600 baudů není podmíněna, šlo by naprogramovat gyroskop pro vyšší rychlosti, ale tato je pro testovací účely zcela dostačující. V bakalářské práci se ze všech dostupných příkazů používají pouze příkazy pro čtení hodnot.

Minimálním rozlišovacím krokem je 8°. Jemnější rozlišení není potřeba například proto, že je potřeba počítat s tím, že letící model nedokáže držet stabilní polohu a dochází k drobným výchylkám vlivem vzdušných proudů.

příkaz	hex	odpověď	poznámka
101	65	integer X ve stupních	Read X
102	66	integer Y ve stupních	Read Y
103	67	integer X, Y ve stupních	Read All
111	6F		Clear X (X=0)
112	70		Clear Y (Y=0)
113	71		Clear All (X=0, Y=0)
121 + integer data	79 + data		Set X
122 + integer data	7A + data		Set Y
201	C9		Prog mode
202	CA		Test mode
222	DE	240 (OK hex=F0)	Status
		250 (error hex=FA)	Error

Tabulka 4.1: Komunikace přes rozhraní RS232 při rychlosti 9600 baud.

4.2 Volba vhodného modelu

Pro vývoj a testování bylo potřeba připojit vhodný RC model letadla. K dispozici jsem měl nedokončený model hornoplošníku typu Cessna s křidélky a zcela funkční cvičný model WildHawk bez křidélek. Pro jednoduchost připojení jsem veškeré testování prováděl na modelu WildHawk, avšak projekt pro poloautomatizaci a následně možné plné automatizaci bude tvořen právě pro hornoplošník.

Celkově není projekt určen pro jeden typ letadla, právě naopak, měl by být schopen ovládat jakýkoliv létající model. Může se však lišit smysl chodu serv a proto je nutné zkontrolovat všechny rutiny, zda-li nastavují pozici mechanických části modelu správným směrem. Ve většině případů však bude chování totožné. Projekt není vhodný pro použití v helikoptérách.

4.3 Modul flightcontrol

Tento modul je stežejní pro automatizaci řízení RC modelů. Jeho primárním cílem je definovat funkce, které budou obstarávat jak velmi jednoduché, tak mnohdy komplexní algoritmy spojené s předletovou a letovou kontrolou. Pro řízení byl v původním návrhu modul *calcCMD*, avšak ten by měl zajišťovat pouze převod vstupních PWM signálů na výstupní. Pokud by se další vývoj práce přiklonil k verzi s pozemní řídící stanicí, měl by modul *calcCMD* zajišťovat komunikaci mezi jednotlivými bloky programu.

Všechny funkce a komentáře jsou psány v anglickém jazyce, aby bylo možné navázat další práci i s mezinárodními studenty a také proto, že v anglických textech se vývojář lépe orientuje.

4.3.1 Pomocné funkce a konstanty

Aby bylo možné měnit aktuální nastavení poloh serv, otáčky pohonné jednotky či údaje z gyroskopické jednotky, byla vytvořena sada konstant, které jsou k nalezení v hlavičkovém souboru tohoto modulu. Všechny konstanty jsou vhodně okomentované, aby bylo na první pohled jasné, jak se v programu používají.

Získání aktuální polohy modelu v prostoru a sériová komunikace

Při tvorbě autonomního letadla se nevyhneme situaci, kdy musíme zjistit aktuální orientaci modelu v prostoru vůči zemi. Přesně k tomuto účelu slouží gyroskopická jednotka a s ní spojená funkce pro zjištění aktuální polohy *GetPosition()*, která zajišťuje komunikaci s gyroskopickou jednotkou. Jako parametr přijímá kód příkazu, který se má odeslat na sériové rozhraní. Jeho návratovou hodnotou je aktuální úhel ve stupních.

Za tímto účelem bylo potřeba doplnit druhý směr komunikace modulu *UART0* a změnu protokolu pro příjem zprávy. Komunikaci ve směru mikrokontroler → RS232 → připojené zařízení nově zajišťuje funkce *SendCommand()*. Prozatím je pouze v testovací fázi a nepočítá s připojením jiných zařízení, než je právě pomocný gyroskop. Velmi jednoduchou úpravou bude možné zasílat řídící povely a data i pro jiná zařízení. Momentálně jde téměř o kopii výše zmíněné funkce.

Příjem zprávy přes rozhraní RS232 je řešeno pomocí přerušení. Jakmile je osmibitový buffer naplněn, vyvolané přerušení se postará o zpracování přijaté zprávy. Původní rutina pro příjem zprávy byla pouze zakomentována, aby bylo možné se k ní, v případě nutnosti, vrátit. Nově lze přijmout pouze dva bajty, které reprezentují úhel náklonu jedné z os modelu.

Bezpečnostní mód

Funkce *FailSafe()* pro převzetí řízení modelu člověkem je velice jednoduchá. Program se zde zacyklí do nekonečna, protože o příjem a zpracování PWM signálů a jejich vyslání na výstup se starají moduly *capturePWM* a *calcCMD*, které byly již vytvořeny. Protože měření signálů je prováděno pomocí přerušení, není třeba v těle funkce nic provádět.

Předletová kontrola správnosti zapojení mechanických částí modelu

Model nemůžeme vypustit do prostoru, aniž by byly na zemi vyzkoušeny veškeré mechanické části, které zajišťují pohyb a řízení modelu. Prozatím jsou ve funkci *PreFlightTest()* odzkoušeny pohyby serv na řízení výškovky, směrovky a křidélek v obou směrech na maximální výchylky a otestování motoru v rozsahu 0-75%. Mezi každým nastavením polohy je nastaven cyklus, který zajišťuje zdržení před nastavováním dalších komponent. Je to proto, že množství zpracovaných instrukcí mikrokontroleru je, v poměru k rychlosti reakce serva (až 20ms), mnohonásobně vyšší a změna na PWM výstupech by nebyla zaznamenatelná.

Aktuální stav projektu nedovoluje žádné uživatelské vstupy, proto funkce slouží ke kontrole správnosti zapojení všech komponent letadla.

Nastavení úhlu serva

Pro usnadnění práce s nastavováním úhlů na jednotlivých servech jsem vytvořil velmi jednoduchou funkci *SetAngle()*, která očekává na vstupech výstupní PWM kanál a velikost úhlu ve stupních. Pro zjednodušení určení kanálu se používají konstanty, aby byla funkce co možná nejuniverzálnější.

Funkce nejprve ověří, zda-li daný úhel, na který má být výstup nastaven, je v povolených mezích. Kdyby tomu tak nebylo, bylo by možné vyslat špatný PWM signál a servo nenávratně zničit. Pokud je vstupní úhel v rozmezí <-45; +45> stupňů, je přepočítána potřebná délka řídícího signálu a poslána na odpovídající výstupní kanál. I zde je však potřeba počítat s prodlevou mezi nastavením PWM signálu a změnou na servech, proto je zařazena velmi malá čekací smyčka.

Ovládání výkonu pohonné jednotky

Při automatizaci musíme také neustále dbát na výkon pohonné jednotky, aby byly dodrženy fyzikální zákony, které drží model ve vzduchu. Nevýhodou samotného motoru v modelu je, že pracuje pouze v režimu zapnuto/vypnuto. Na trhu lze zakoupit regulátor pro pohonné jednotky, který je taktéž řízen PWM signály. Oproti servům je zde jeden zásadní rozdíl. Délka PWM signálu v centrální poloze, tj. 1,5ms není 0% výkon, ale výkon 50% $\Rightarrow$ je potřeba nastavovat délku řídícího signálu od spodní hranice.

SetEnginePower() tedy funguje na podobném principu, jako nastavení úhlu serva s tím rozdílem, že jako vstupní parametr čeká výkon v procentech.

Stabilizace křídel do vodorovné polohy

Při vytváření funkce pro stabilizaci jsem narazil na několik problémů a předem nejasných otázek.

1. Jsou k dispozici křidélka, která zajistí rotaci?
2. Lze za stabilní polohu považovat i polohu převrácenou, tj. kabinou směřující k zemi?
3. Jakým směrem a s jakou intenzitou bude stabilizace probíhat?

Na obrázku 4.2[1] můžeme vidět, jaký je směr rotace, přijdou-li data z gyroskopické jednotky.

Pokud nemáme k dispozici křidélka, bude poměrně velký problém stabilizovat model, protože nemáme k dispozici žádný prvek, který by umožnil modelu rotovat kolem své podélné osy (v bakalářské práci označována jako osa X). Bez pomoci křidélek dojde k vyrovnání modelu za delší dobu, protože máme k dispozici pouze směrovku a proto při změně její polohy nebude letadlo rotovat pouze podél osy X, ale také bude měnit směr letu.

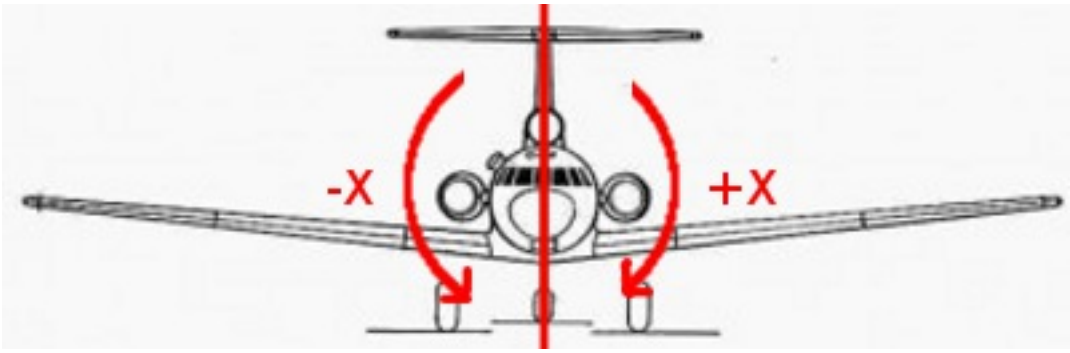
Dalším problémem, který je třeba vyřešit, je definice stabilního stavu. Zda-li je to stav, kdy má model křídla vodorovně vůči zemi s přihlédnutím na orientaci kabiny nebo se pozice kabiny vypustí. Zvolil jsem variantu s kabinou směrem vzhůru, neboť let na zádech už patří mezi akrobatické prvky, které je možné provádět za předpokladu, že máme letadlo zcela ovladatelné a řídící jednotka je schopna nezávisle a současně správně rozhodovat.

Posledním úkolem při návrhu stabilizace bylo, aby se zajistilo vždy správné rozhodnutí, jakým směrem se bude model vracet do své stabilní polohy. Měl jsem k dispozici dvě možné cesty:

1. Vždy nejkratší možnou cestou.
2. Pokud model rotuje rychle, pokračovat v rotaci při plném vychýlení křidélek až do polohy $\pm 90°$ od stabilní pozice. Jakmile bude tato hranice překonána, dojde ke zmenšení náklonu křidélek, aby rychlost rotace klesla.

Opět z důvodu složitějšího akrobatického prvku, kterým je pokračování rotace do vodorovné polohy jsem zvolil cestu nejkratší možné rotace. Pokud je úhel větší než $90°$, vychýlí se křidélka na maximum, jinak se vychýlí do poloviny aktuálního naklonění. Čili v průběhu rotace se náklon křidélek snižuje vždy o polovinu zbývajícího úhlu do rovnovážné polohy.

[1] Nákresy použity ze serveru http://www.lkmtspotter.estranky.cz/fotoalbum/schemata-letadel



Obrázek 4.2: Rotování kolem podélné osy.

Stabilizace trupu do vodorovné polohy

Stabilizovat model podle jeho příčné osy je velmi podobné, jako vyrovnat jej podle osy podélné. Zde je ale rozdíl ten, že je potřeba si pohlídat, zda-li model stoupá či klesá a podle toho nastavovat nejen směr vychýlení výškovky, ale také otáčky pohonné jednotky, aby nedošlo buď k poklesu rychlosti a ztráty kontroly z důvodu nízkého tahu motoru nebo naopak k nabrání příliš velké rychlosti při klesání a možnému poškození modelu jak v důsledku působících sil, tak v důsledku velmi rychlého přiblížení překážky.

Na obrázku 4.3 můžeme vidět, jaký je směr rotace, přijdou-li data z gyroskopické jednotky. V tomto momentě si bylo potřeba položit otázku, zda-li kontrolovat, jestli je model stabilizován podél osy X nebo tuto kontrolu vynechat. Protože jde o pomocnou funkci, ke kontrole stabilizace podle osy X nedochází. Tuto vlastnost by šlo jednoduše vyřešit, avšak myslím si, že to nutné není a pouze by se tak komplikoval kód. Výsledný verdikt však určí až praktický test ve vzduchu, ke kterému ještě nějakou dobu nedojde.

Obrázek 4.3: Rotování kolem příčné osy.

Kompletní stabilizace

Pro uvedení modelu do zcela stabilní polohy, kterou je myšleno vyrovnání podle osy X a osy Y je určena funkce *Stabilizate()*. Jejími vstupy jsou tolerance náklonu v obou osách. Tolerance je zde nezbytná z již zmíněného důvodu, že se model během svého letu neustále vychyluje z vyrovnané polohy a díky povětrnostním podmínkám neustále mění směr, ačkoli to lidské oko na dálku nezaznamená.

Aby bylo možné prohlásit, že je orientace letadla v mezích stability, musí být splněny následující tři podmínky.

1. Osa X je v toleranci.
2. Osa Y je v toleranci.
3. Jeden kontrolní průchod stabilizací nevykázal žádné překročení jedné z krajních mezí.

Před zahájením jakýchkoli prostorových operací se nastaví výkon motoru na 75%. Při tomto výkonu by model neměl nabrat příliš vysokou rychlost, ale také by se neměl dostat pod nejnižší možnou letovou rychlost. V následujícím kroku je model vyrovnán podle osy X, aby bylo možné začít vyrovnávat podél osy Y, je třeba zajistit, aby směřoval buď směrem vzhůru nebo dolů. Dalším krokem je vyrovnání podle osy Y. V celém projektu je možné narazit na globální proměnnou `stable`. Ta je po téměř celou dobu běhu programu nastavena na výchozí hodnotu `0`, která určuje, že je model v nestabilní poloze. Použití globální proměnné jsem zvolil z důvodů toho, že v některých z budoucích algoritmů se může model odkazovat právě na prostorovou orientaci. Následně dochází ke čtení aktuálních náklonů a porovnání s tolerancí. Neodpovídá-li jedna z hodnot, je tento proces neustále opakován, dokud obě hodnoty nevyhovují. Jakmile je vše v pořádku, nastaví se výkon motoru na 50%.

Zjišťování aktuální letové hladiny

Jednou ze základních informací, nutné pro zautomatizování ovládání, je možnost získat data o aktuální výšce nad zemí. Jedno z využití této informace je například při navádění modelu na přistání, vyhýbání se překážkám při letu na nízkých letových hladinách apod. Problémem ale je, že precizní výškoměry, které by byly schopny rozlišovat změnu výšky s maximální odchylkou jeden metr a současně byly schopny dodat přesné výsledky v okamžiku dotazu, jsou položky v řádu několika tisíc korun.

Některým lidem se hned může vybavit možnost měření nadmořské výšky podle GPS. Tato metoda by byla vhodná pouze při letu na vyšších letových hladinách, kdy nepotřebujeme znát přesnou výšku a odchylka ± 10 metrů nečiní ve výsledku problém, protože nehrozí žádná bezprostřední kolize. Naopak rychlost, s jakou jsou dodávány aktuální údaje o letové hladině, je velice malá.

Prozatím byla funkce *GetCurrentAltitude()* vytvořena jen symbolicky a sestává z cyklu, který neustále vrací jinou hodnodu, která je dekrementována a simuluje tak klesání.

Navedení modelu na danou letovou hladinu

I přes chybějící výškoměr nebyl problém implementovat funkci *GetToAltitude()*, která se postará o vynesení, případně klesnutí modelu na danou letovou hladinu. Modul opět hojně využívá stabilizační funkce, aby bylo zajištěno bezpečné stoupání či bezpečné klesání.

V první fázi dojde ke stabilizaci v obou osách a následně je rozhodnuto, zda-li bude model stoupat či klesat. Při každém průchodu cyklu jsou stabilizována křídla, aby nedošlo k havárii. Následně je zjištěn aktuální náklon a pokud neodpovídá danému rozsahu, je pravděpodobné, že se stalo něco nepředvídatelného a model je uveden do klidové polohy a pokus o stoupání se opakuje.

Předem definovaným úhlem pro stoupání je elevační úhel 45° a přijatelná odchylka 8°. Pokud se aktuální úhel liší o hodnotu vyšší, než je daná odchylka, je spočítán rozdíl, stupňů mezi fixním úhlem pro stoupání a aktuálním náklonem, následně je rozhodnuto,

kterým směrem se nakloní výškovka a polovina rozdílu úhlů je odeslána v podobě PWM signálu na servo. Tímto se zajistí velmi jemné změny úhlu náklonu.

Jakmile se model přiblíží na cílovou letovou hladinu, kde rozhodující hranící je tolerance, je stabilizován podle os X a Y.

4.3.2 Modul UART0

Drobných změn se dočkal i modul *UART0*, protože bylo potřeba zajistit komunikaci s testovací gyroskopickou jednotkou. Byla vytvořena funkce pro zasílání příkazů na sériový port, která již byla popsána v úvodu této kapitoly. Následně byla změněna obsluha přerušení. Kdykoliv je naplněn buffer sériového rozhraní, je vyvoláno přerušení a požadováno zpracování příchozích dat.

Testovací gyroskopická jednotka zasílá celočíselnou hodnotu velikosti dva bajty, což odpovídá velikosti celočíselné proměnné v mikrokontrolerech MSP430. Jedinou vadou komunikace mezi mikrokontrolerem ATMEGA8-16 a MSP430 je změna pořadí čtení bajtů. Prvních osm bitů, ktere jsou přijaty odpovídají spodním bitům celočíselné proměnné a zbylých osm bitů zase bitům horním. Z tohoto důvodu byla vytvořena funkce *GetTransferedInteger()*, která převede přijatá data na správný tvar.

Kapitola 5

Potřebné nástroje a zapojení

Pro jednoduchý vývoj a testování projektu doporučuji použít nástroj *IAR Embedded Workbench* 3.1 právě kvůli jednoduchosti použití, přiložení veškerých ovladačů a podporou mnoha modelů procesorů nejen z rodiny MSP430. Pro potřeby vývoje jsem si zažádal o třicetidenní zkušební verzi, verzi Kickstart, která je zdarma, jsem nezkoušel.

Po přeložení a následném spuštění na cílovém hardware se může objevit zpráva o tom, že je k dispozici novější verze firmware pro emulátor. Neaktualizoval jsem jej a používal původní. Je možné, že toto bylo příčinou častých komunikačních problémů, které občas vedly i k pádu samotného vývojového prostředí. Avšak odpojení a připojení emulátoru do portu USB ve většině případů problémy vyřešilo. Dále je třeba pamatovat, že při testování metodou krok-po-kroku nelze čekat na přerušení a některé komponenty nefungují. Takové příkazy je třeba provést naráz, aby se změna projevila.

Jednou ze základních věcí, na které je třeba myslet, je napájení cílové desky. Nestačí +3,3V, které jsou v JTAG rozhraní, protože serva pracují na +5V. Napájení je řešeno stejně jako v modelech letadel, tj. stačí připojit na správné piny určené pro připojení serv. Nezáleží na tom, jde-li o vstup nebo výstup. Další problém může přinést připojení testovací gyroskopické jednotky, která se vlivem větší spotřeby reseture při pohybu serva. Je třeba ji napájet z vlastního zdroje s napětím +5V. Pokud dojde k přerušení napájení při komunikaci po RS232 rozhraní, program "zatuhne". Je to proto, že čeká na zápis do vstupního bufferu, ale gyroskop po restartu žádnou zprávu nezašle.

Testovat lze na jakémkoliv modelu, avšak projekt je primárně určen pro modely letadel. Řídící jednotka nerozlišuje, jaký typ pohonné jednotky má k dispozici, protože na celkové řízení to nemá vliv. Jediné, na co je třeba pamatovat, je předpřipravené rozložení výstupních pinů určených pro konkrétní součást letadla. Na obrázku 3.1 jsou výstupní piny na konektoru `J6`. Připojení jednotek je zprava výškovka, směrovka, křidélka, motor. Toto nastavení lze změnit změnou konstanty v modulu *flightcontrol*.

Kapitola 6

Závěr

Cílem celé práce bylo vytvořit algoritmy pro částečnou kontrolu letových vlastností v rámci velkého projektu zaměřeného na vytvoření autonomního modelu. Tento projekt je prozatím ve velmi ranném stádiu vývoje a veškerý hardware, potřebný k vytvoření takového modelu, prozatím není k dispozici mimo jiné i pro cenovou nedostupnost některých komponent. Jedním z možností zpracování zadání bylo využít současnou desku s mikrokontrolerem MSP430.

Během práce se vyskytlo hned několik zajímavých problémů. Tím největším bylo polohové zařízení, které by určilo, jakou orientaci má momentálně model v prostoru. Zkoumal jsem několik možností, jak vytvořit gyroskopickou jednotku alespoň pro prozemní testy a jako nejvhodnější řešení bylo použití základu z kuličkové myši, kdy si vývojář může velmi pohodlně nastavovat orientace bez nutnosti zásahu do zdrojového kódu.

Vytvořením stabilizačních rutin, pomocných funkcí pro pohodlné nastavování poloh jednotlivých serv, navigační funkce pro let na dané letové hladině a možnosti kontroly výkonu elektromotoru byly vyčerpány možnosti, které současný stav projektu nabízí. Cíl zadání byl splněn a díky novému hardware i rozšířeny aktuální možnosti.

Možnosti projektu do budoucna

Projekt má před sebou ještě velmi mnoho práce a budu-li mít možnost, rád bych se na ní podílel i nadále. Mezi prvními změnami navrhuji změnit konektor na GPS a modul *UART1* vyhradil výhradně pro komunikaci s budoucí pozemní stanicí, aby bylo možné komunikovat s letícím modelem. V současné době není možné model nijak vzdáleně ovládat, což je velmi svazující.

Také bych se přiklonil k vytvoření komunikačního protokolu mezi jednotlivými zařízeními na sériové lince například: `#ID_ZAŘÍZENÍ;DATA$`. Komunikace by probíhala tak, že se zašle po sériové lince identifikátor cílového zařízení a řídící příkaz. Zařízení potvrdí příjem a odešle zpět data. Následně odešle data ještě jednou, ale v inverzní podobě. Pak pomocí exkluzivního logického součtu musí výsledná hodnota být nula. Je-li tomu tak, přenos pravděpodobně proběhl v pořádku a data můžeme považovat za platná.

Bylo by zajímavé nastudovat možnosti bezdrátového spojení s přihlédnutím na rychlost, kvalitu a možnou maximalní vzdálenost komunikace mezi modelem a pozemní stanicí. Zde by byl potřeba klást důraz na nízkou spotřebu, protože přidávání nových komponent bude mít za následek kratší výdrž, zejména jedná-li se o model poháněný elektromotorem.

Pro navádění modelu na přistání by mohlo být zajímavé nastudovat možnosti měření

vzdálenosti pomocí laserových paprsků, které by zcela postačovali s dosahem dvacet metrů. Jejich náklon by teoreticky mohl být 45° v přední části modelu a měření vzdálenosti od země by probíhalo pouze ve fázi přistání či nízkých průletů. V běžném letu ve vyšších hladinách by zcela postačovaly údaje z GPS modulu.

Literatura

[1] Bordelon, T. J.: Freespace EZStar. *Terence J. Bordelon*, 2009, [online]. [cit. 2010-04-27].
URL http://bordelon.net/ezstar.html

[2] Chytilová, M.: Archimedův zákon. Technická zpráva, Univerzita Karlova v Praze, rok neznámý, [online]. [cit. 2010-03-31].
URL http://fo.cuni.cz/texty/arch_zak.pdf

[3] ENAC University: Paparazzi Project. *École Nationale de l'Aviation Civile*, 2010, [online]. [cit. 2010-04-28].
URL http://paparazzi.enac.fr/wiki/Main_Page

[4] Hynek Bulíř, J.: Proč letadlo létá? *Český rozhlas*, 2007, [online]. [cit. 2010-03-31].
URL http://www.rozhlas.cz/vedaarchiv/veda/_zprava/349954

[5] Jaroslav Reichl, M. V.: Základy fyziky letu. *Encyklopedie Fyziky*, rok neznámý, [online]. [cit. 2010-03-31].
URL http://fyzika.jreichl.com/index.php?sekce=browse&page=130

[6] Kolektiv autorů: *ČSN 31 0001 (310001) Letecké názvosloví*. Český normalizační institut, 2005.

[7] Procerus Technologies: KESTREL Autopilot. *Procerus Technologies*, 2010, [online]. [cit. 2010-04-25].
URL http://www.procerusuav.com/productsKestrelAutopilot.php

[8] RecreationAviation.com: RC Plane History. *recreationaviation.com*, rok neznámý, [online]. [cit. 2010-03-24].
URL http://www.recreationaviation.com/rc_plane_history.htm

[9] Slaný, K.: Konstrukce. *Letecký ústav, Fakulta strojního inženýrství, Vysoké učení technické v Brně*, rok neznámý, [online]. [cit. 2010-03-31].
URL http://lu.fme.vutbr.cz/ucebnice/opory/construction.php

[10] Texas Instruments Incorporated: MSP430F15x, MSP430F16x, MSP430F161x Mixed Signal Microcontroller (Rev. F). *Texas Instruments*, 2009, [online]. [cit. 2010-04-11].
URL http://focus.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=msp430f169&fileType=pdf

[11] Texas Instruments Incorporated: MSP430F169. *Texas Instruments*, 2009, [online]. [cit. 2010-04-11].
URL http://focus.ti.com/docs/prod/folders/print/msp430f169.html

[12] WWW stránky: Aerodynamika. *Wikipedia*, rok neznámý, [online]. [cit. 2010-03-31]. URL http://cs.wikipedia.org/wiki/Aerodynamika