VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

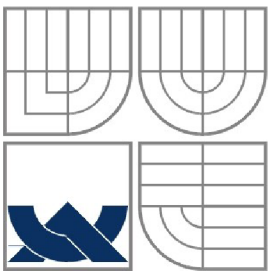
PLATFORMA PRO VÝVOJ TŘÍ-ROTOROVÉ HELIKOPTÉRY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

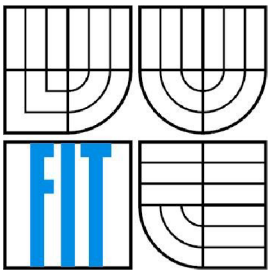
AUTOR PRÁCE
AUTHOR

Bc. MARTIN VOTAVA

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PLATFORMA PRO VÝVOJ TŘÍ-ROTOROVÉ HELIKOPTÉRY

DEVELOPMENT OF PLATFORM FOR THREE-ROTOR HELICOPTER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN VOTAVA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOSEF HÁJEK

BRNO 2012

Abstrakt

Cílem této práce je návrh a stavba platformy pro vývoj třírotorové helikoptéry, nazývané tricopter.
Teoretická část práce vysvětluje principy letu helikoptér a způsob ovládání tricopteru. Popisuje činnost inerciálního navigačního systému a jednotlivých senzorů. Seznamuje čtenáře s použitými řídící a měřící prvky.
Praktická část se zabývá návrhem vhodné konstrukce tricopteru, která byla pro účely této práce postavena. Je uveden návrh hardwarové části řídícího systému. Popisuje vzájemnou komunikaci dílčích subsystémů, dekódování řídících signálů a PID regulátor, který zajišťuje stabilizaci letu.

Abstract

The goal of this master's thesis is design and built of platform for three-rotor helicopter development. The helicopter is also known as tricopter.
Theoretical part describes principle of tricopter's flight and stabilization. There is also described basics inertial navigation system and sensors which are required for correct functionality.
Practical part is dedicated to development of tricopter's frame, schematics diagram, communication between subsystems and stabilization system development. Flight stablization system is base on ATmega128A an using PID Controller. In the end is described testing of developed platform.

Klíčová slova

Atmega128A, tricopter, stabilizace letu, akcelerometr, gyroskop, magnetometr, ultrazvukový dálkoměr, PID regulátor

Keywords

Atmega128A, tricopter, flight stablization, accelerometer, gyroscope, magnetometer, ultrasonic range finder, PID controller

Citace

Martin Votava: Platforma pro vývoj tří-rotorové helikoptéry, diplomová práce, Brno, FIT VUT v Brně, 2012

Platforma pro vývoj tří-rotorové helikoptéry

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Josefa Hájka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Martin Votava
18.5.2012

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Josefu Hájkovi za poskytnuté konzultace a rady.

Obsah

1 Úvod

S rozvojem výkonu vestavěných systémů se rozšiřuje i možné pole jejich působnosti. Vestavěné systémy lze nalézt v téměř každém kolovém či pásovém robotovu. Se stoupající dostupností MEMS senzorů se také otevírají možnosti využití těchto systémů v létajících robotech. Vestavěný řídící systém může usnadňovat operátorovi řízení transformováním jeho povelů z vyšší úrovně abstrakce na konkrétní změny řídících signálů pro jednotlivé akční prvky. Další úlohou systémů je stabilizace letových vlastností těchto robotů.

Oproti dříve rozšířeným koncepcím, kdy se létající roboty stavěly převážně jako vrtulová letadla, se v dnešní době převážně používají vrtulníky. Avšak konstrukce jednorotorového vrtulníku není pro svojí nestabilitu vhodná. Ve velkém se rozšiřují vrtulníky, často také nazývané coptery, které mají tři a více rotorů. Tyto vrtulníky nemusí mít rotory s pohyblivými listy, což jejich stavbu značně zjednodušuje a tím je činí finančně dostupnějšími. Částečnou nevýhodou této koncepce je vyšší počet motorů, protože naklápění copteru se provádí změnou otáček rotoru. Tento fakt částečně zvyšuje pořizovací cenu, která stoupá s počtem rotorů, které stroj má.

Coptery lze obecně rozdělit na stroje s lichým nebo sudým počtem rotorů. Ovládání každé z uvedených skupin se v určitých aspektech různí. U copterů se sudým počtem rotorů je u poloviny rotorů vyžadován opačný smysl otáčení, čímž se ruší momenty jednotlivých rotorů. Tato potřeba značně omezuje možnost volby součástek pro stavbu. Na druhou stranu u strojů s lichým počtem rotorů nelze anulovat kroutící momenty jednotlivých rotorů protiběžnými otáčkami poloviny z nich. Nevzniká zde tedy požadavek na protiběžné rotory. Nutnost vyrušit rotační moment zde zůstává. Řešením je naklápění jednoho z rotorů tak, aby část tahu vyrovnávala momenty ostatních rotorů. Cenou za větší výběr stavebního materiálu je složitější konstrukce.

V akademických pracích se autoři zabývají pouze čtyř nebo šestirotorovými vznášedly. Třírotorovým copterem se žádná práce nezabývá. Právě proto je cílem této diplomové práce návrh a stavba třírotorového copteru. Kromě konstrukce je v této práci také popsán návrh řídícího a stabilizačního systému za pomoci inerciálního senzorů. Systém je koncipován jako vývojová platforma, která bude umožňovat další rozvoj vznášedla.

Druhá kapitola této práce popisuje základní principy letu strojů těžších než vzduch. Kromě popisu letadel a konvenčních vrtulníků jsou zde popsány základní letové vlastnosti copterů, především tricopteru včetně principů řízení.

Třetí kapitola seznamuje čtenáře s navigačními systémy, které umožňují létajícím strojům určovat svojí pozici a rychlost vůči počátečnímu bodu pouze na základně zpětnovazebního měření. Dále jsou zde popsány senzory, které umožňují detekovat absolutní azimut a výšku. U každého ze senzorů je uveden alespoň jeden, na kterém může senzor pracovat.

Ve čtvrté kapitole je popsána a zobrazena mechanická konstrukce copteru, který byl v rámci této práce vytvořen. Copter využívá modelářské součástky, především motory a regulátory. Tyto součástky se v poslední době stávají velmi dostupnými.

Pátá kapitola se zabývá návrhem hardwarové části řídícího systému. Především je zde zobrazeno schéma zapojení řídící jednotky včetně popisu použitých mikrokontrolérů a senzorů. Jsou zde uvedeny možnosti komunikace s použitými senzory a způsoby jejich nastavení pro efektivní činnost. Na konci kapitoly jsou uvedeny pomocné obvody, které slouží pro programování a komunikaci s mikrokontroléry během vývoje.

Poslední kapitola popisuje softwarové řešení řídícího systému. Jednak jsou zde vysvětleny techniky, kterými se komunikuje s jednotlivými senzory, dále způsob připojení k modelářskému setu vysílačky a přijímače včetně dekódování přenášeného signálu a také nastavování parametrů za běhu. Je zde také popsán použitý řídící algoritmus vycházející z PID regulátorů, které se starají o základní stabilizaci copteru. Závěr kapitoly je věnován metodám testování algoritmů a prezentuje dosažené výsledky.

2 Vrtulníky a coptery

Cílem této kapitoly je vysvětlit základní principy letu a stabilizace létajícího vznášedla. Základní principy jsou vysvětleny na nejběžnějším stroji – na vrtulníku. Dále je zde popsáno rozdělení vrtulníků a vznášedel do několika kategorií v závislosti na počtu nosných rotorů. Primárně se jedná o jedno a dvou-rotorové vrtulníky. Tyto stroje patří mezi běžnější létající stroje pro přepravu osob a materiálu. Létající stroje s vyšším počtem rotorů se již nenazývají vrtulníky, ale coptery. Jejich použití je prozatím omezeno pouze na pokusné a výzkumné konstrukce, případně na modely v příslušných měřítkách. Mezi základní představitele těchto copterů patří tricopter, quadrocopter a hexacopter. Jak již názvy napovídají, jedná se o stroje se třemi, čtyřmi nebo šesti nosnými rotory. Mezi nesporné výhody copterů patří jejich jednodušší konstrukce, se kterou souvisí i jednodušší údržba a nižší pořizovací cena.

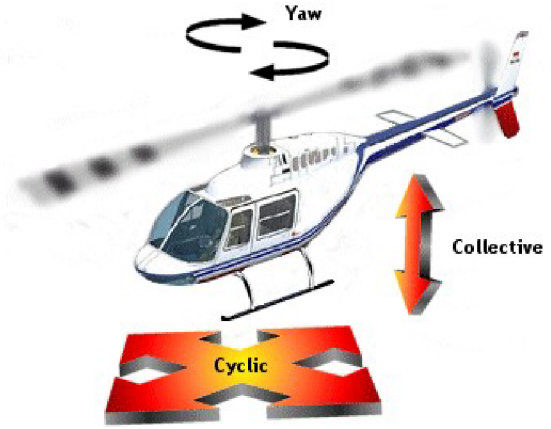
2.1 Vrtulníky

Vrtulník je motorový stroj, který je těžší než vzduch [1][2]. Jediným zdrojem nosné i pohonné síly jsou motorem poháněné nosné plochy rotující v přibližně vodorovné rovině. Na rozdíl od letadla tak nevzniká vztlak pouze díky relativní rychlosti profilu křídla vůči proudění vzduchu, ale především nuceným oběhem listů rotoru. Proto nemá vrtulník kritickou rychlost, při které by přestal profil křídla vytvářet potřebný vztlak. Vrtulník dokáže přejít do tzv. visu (tj. vznášení se nad jedním místem) či startovat z místa. Daní za tyto výhody je mnohonásobně vyšší spotřeba paliva, případně jiné energie sloužící pro pohon rotoru. Zatímco u letadla motor vytváří pouze potřebný tah pro udržení rychlosti, takže stroj dokáže do určité vzdálenosti doplachtit i při úplném výpadku motoru, u vrtulníku musí motor udržovat konstantní otáčky rotoru, jinak by došlo ke ztrátě vztlaku a pádu. Pro úplnost je nutné dodat, že ještě existuje možnost letu za pomoci autorotace, tj. stav, kdy se rotor otáčí pouze vlivem odporu vzduchu, nicméně její možnosti jsou silně omezeny.

Zdrojem vztlaku, který drží vrtulník ve vzduchu, je jeden či více rotorů. Každý rotor se skládá z hlavice rotoru a listů, které jsou na ní připevněny. Počet listů rotoru se odvíjí od požadované nosnosti vrtulníku a rychlosti otáček – běžné jsou 2 až 4 listy u běžných vrtulníků, u vysokotonážních až 8 listů. List rotoru je vhodně tvarovaná nosná plocha. Často se využívá profil shodný s profilem běžného křídla, ačkoliv z aerodynamického hlediska není jednoznačné, který profil je nejvhodnější.

Základní a nejběžnější koncepcí vrtulníku je jednorotorový vrtulník [3]. Tato koncepce používá jeden hlavní nosný rotor pro vytvoření vztlaku i pro pohon vrtulníku požadovaným směrem. Pro vyrovnání reakčního (kroutícího) momentu má ještě jeden menší vyrovnávací rotor. Pohon tohoto rotoru spotřebovává část výkonu motoru. Při visu poskytuje nosný rotor dostatečný vztlak a zároveň

zadní rotor vyrovnává kroutící moment. Má-li vrtulník zrychlit kterýmkoliv směrem, musí nosný rotor vytvořit dodatečný tah právě v požadovaném směru. Toho lze dosáhnout pouze nakloněním celého rotoru v tomto směru. Tím nebude veškerá síla vytvářená rotorem působit pouze proti gravitačnímu zrychlení, ale část bude vytvářet zrychlení v požadovaném směru, a proto je třeba zároveň přímo úměrně upravit tah rotoru tak, aby vertikální tah rotoru zůstal stejný. V praxi však není reálné, aby se celý rotor nakláněl a de facto není možná ani přesná regulace otáček rotoru. Namísto toho se používá řešení, kdy jednotlivé listy rotoru mohou dynamicky měnit svůj úhel náběhu a tím i tah, který vytvářejí. Naklánění rotoru se vytváří pomocí tzv. cyklického řízení. Má-li se celý rotor natočit, upraví se nastavení úhlu listů rotoru tak, aby listy, které jsou na straně, kam se má rotor natočit, měly nižší úhel náběhu. Naopak listy na opačné straně by měly vyšší úhel náběhu. Každý list pak při otáčení postupně mění svůj úhel náběhu a tím vytváří stejný efekt jako by byl nakloněný do daného směru. Je-li třeba změnit tah celého rotoru, pak se pomocí tzv. kolektivního řízení upraví shodně úhel všech listů. Pro otáčení vrtulníku podél osy hlavního rotoru slouží vyrovnávací rotor. Pilot může ovládat pomocí pedálů náběh listů tohoto rotoru, tím i jeho tah a ve výsledku i rychlost rotace vrtulníku podél svislé osy. Přesný popis mechanizmů náklonu a letu vrtulníku není součástí této práce a je podrobně popsán např. v [1].



Obrázek 2.1: Princip řízení vrtulníku [3].

2.2 Dvourotorové vrtulníky

Do kategorie vícerotorových vrtulníků patří mnoho různých typů. V první řadě se jedná o dvourotorové vrtulníky. Výhoda těchto strojů spočívá především v tom, že není potřeba použít vyrovnávací rotor, protože reakční momenty rotorů se navzájem vyruší, a proto lze využít plný výkon motoru. Existují celkem tři možné typy konstrukce dvourotorového vrtulníku.

První z možností je souosé uspořádání rotorů. Z celkem logických důvodů musí mít rotory protichůdné otáčky. Vzhledem k nulovému zlepšení letových vlastností oproti jednorotorovému řešení a technické náročnosti konstrukce se toto řešení v praxi nepoužívá. S tímto řešením je možné setkat se u nejjednodušších RC modelů. Druhé uspořádání rotorů je tandemové řešení – rotory jsou uspořádány podélně za sebou. Toto řešení se používá u těžkých transportních vrtulníků, protože umožňuje podélný posun těžiště stroje, což je pro převoz nákladu požadovaná vlastnost. Vztlak zajišťují dva rotory s protiběžnými otáčkami. Zadní rotor musí být umístěn výš než přední, aby se snížilo vzájemné ovlivňování rotorů při dopředném letu. Také musí být zajištěno vzájemné propojení rotorů. Listy rotoru se během rotace mírně kývají (tj. konec listu se může pohybovat nahoru a dolu) a je tedy nutné zajistit, aby nemohlo dojít ke kontaktu listů rotoru a jejich destrukci. Ovládání letu je shodné s principem jednorotorového vrtulníku. Jediným rozdílem je rotace podél svislé osy, která se provádí náklonem cyklického řízení jednoho či obou rotorů tak, aby vznikl požadovaný kroutící moment. Třetí konstrukční uspořádání je umístění rotorů příčně ke směru letu. Princip letu je podobný jako u vrtulníku s tandemovým uspořádáním rotorů s tím rozdílem, že celý stroj je otočený o 90°. Běžně se takovéto řešení nepoužívá. Je však možné nosníky rotorů využít jako běžné křídlo, které by sloužilo vytvářelo vztlak při vyšších rychlostech a rotory by pak sloužily pouze k vytváření síly pro pohyb vpřed. Toto řešení využívá například stroj Boeing V22 Osprey nebo AgustaWestland AW609. Tyto stroje při nízkých rychlostech fungují jako vrtulník s příčně umístěnými rotory, při dosažení potřebné rychlosti se pak rotor překlopí do vodorovného směru a stroj se chová jako běžné letadlo. Výhodou těchto strojů je možnost dosažení vysoké rychlosti, které z konstrukčních důvodů běžné vrtulníky nikdy dosáhnout nemohou.

2.3 Tricopter

Tricopter je vznášedlo se třemi rotory, které jsou uspořádány do tvaru písmene Y [8]. Rotory jsou umístěny ve stejné vzdálenosti od těžiště tak, aby všechny spojnice os rotorů s těžištěm svíraly navzájem úhel 120°. Pro zlepšení letových vlastností tricopteru je vhodné, aby jeden z rotorů měl opačný směr otáčení než zbylé dva. Částečně by tak došlo ke zmírnění účinků točivého momentu rotorů. Ale i při použití rotorů se shodným směrem otáčení je možné momenty anulovat. Řízení tricopteru se provádí změnou tahu jednotlivých rotorů. Tím vzniká moment, který způsobí natočení modelu a ve výsledku i zrychlení. Na rozdíl od vrtulníků se u tricopteru nemění tah rotoru změnou náběžného úhlu listu, ale změnou otáček motoru. Díky tomu je možné použít jednoduché vrtule s pevným náklonem listů, které jsou levnější. Pro rotaci kolem svislé osy a kompenzaci rotačního momentu, který vytvářejí jednotlivé rotory, se používá naklánění zadního rotoru. Je samozřejmé,

že při natočení rotoru musí také bezpodmínečně dojít k úpravě otáček rotoru, protože jeho tah svislým směrem se znatelně změní.

Obrázek 2.2: Tricopter [8].

2.3.1 Stabilizace tricopteru a letové vlastnosti

Na rozdíl od běžných vrtulníků nepoužívají coptery k úpravě směru letu naklánění rotoru, ale natáčí se celý copter. Pohyb tricopteru lze rozdělit do dvou složek. První z nich je posun (pro zjednodušení lze předpokládat pouze těžiště), druhá je rotace okolo těžiště. Stejně tak lze i rozdělit tah rotorů. První složka, jejíž velikost lze vypočítat jako aritmetický průměr tahu všech rotorů, vytváří sílu působící ve směru předních dvou rotorů. Je nutné dodat, že u zadního rotoru je nejprve nutné vypočítat rozdělení tahu do svislého a vodorovného směru. Druhá složka, jejíž velikost je rozdílem průměru a velikosti tahu příslušného motoru, vytváří rotační moment. Princip natáčení lze srovnat s principem dvojzvratné páky. Rotaci okolo podélné osy copteru ovlivňují pouze motory na předních nosnících, protože zadní motor je umístěn přímo v ose rotace. Rotace se dosáhne tak, že se zvýší otáčky jednoho z motorů a zároveň se sníží otáčky druhého stejným dílem. Výsledkem pak je rotace při zachování stabilní polohy.

$$M_X = F_l \cdot l \cdot \sin(60\,^\circ) - F_r \cdot l \cdot \sin(60\,^\circ) \tag{2.1}$$

Rotaci podél příčné osy způsobuje rozdílný tah předních a zadního motoru. Při zvýšení tahu zadního rotoru a současném snížení tahu předních rotorů bude stroj rotovat kolem příčné osy.

$$M_Y = F_l \cdot l \cdot \cos(60\,^\circ) + F_r \cdot l \cdot \cos(60\,^\circ) - F_z \cdot l \cdot \cos(\alpha) \tag{2.2}$$

Rotace okolo svislé osy se dá dosáhnout nakloněním zadního rotoru podél příčné osy. Tím vznikne rozdíl mezi točivým momentem jednotlivých rotorů a momentem, který působí naklonění

zadního rotoru. Znaménka jednotlivých členů závisí na směru otáčení jednotlivých rotorů. V případě, že rotor bude mít opačný smysl otáčení, změní se u jeho momentu znaménko.

$$M_Z = M_l + M_r + M_z - F_z \cdot l \cdot \sin(\alpha) \tag{2.3}$$

Pro vytvoření zrychlení v libovolném směru je nutné nejprve naklonit celý stroj. Ve vodorovné pozici působí tah motorů pouze proti gravitačnímu zrychlení a udržuje tak stabilní pozici. Při naklonění dojde k rozdělení tahu. Část síly bude působit ve směru naklonění copteru. Vzhledem k tomuto rozdělení sil je následně nutné upravit tah motorů, aby došlo k zachování výšky.

Tah vrtulí přímo odpovídá otáčkám rotoru. Matematické vyjádření závislosti tahu vrtule na jeho otáčkách a následně i na délce řídícího pulsu lze označit za výpočetně složité a nad možnosti řídícího procesoru. Zároveň je nutné brát v úvahu i rozdílné vlastnosti jednotlivých rotorů a také možné drobné odchylky regulátorů. Místo matematického modelu výpočtu tahu je tedy jednodušší tah změřit a naměřené hodnoty aproximovat vhodnou funkcí, ideálně lineární. K měření tahu motoru je možné využít metodu měření sil, která ale vyžaduje uchycení motoru dostatečně pevné, aby se motor neotáčel, a zároveň dostatečně volné, aby uchycení neovlivňovalo výsledek měření. Alternativou je použití váhy a měření hmotnosti. Každé těleso ležící na podložce na ní působí konstantní silou. Velikost této síly využívá váha ke změření hmotnosti tělesa.

Rotací vrtule vzniká kromě tahu také reakční točivý moment, který je následkem odporu vzduchu. Každý z těchto momentů bude působit na model tak, že se jej bude snažit roztočit okolo svislé osy. Je tedy nutné zjistit velikost tohoto momentu, aby bylo možné jej kompenzovat změnou náklonu zadního rotoru. Pro změření tohoto momentu lze použít jednoduchý přípravek na principu jednozvratné páky.

2.4 Quadrocopter

Quadrocopter je stroj se čtyřmi rotory. Pro konstrukci je bezpodmínečně nutné, aby se dva rotory točily po směru hodinových ručiček, druhé dva pak opačným směrem. Motory jsou umístěny do čtverce tak, že motory se souhlasným směrem otáčení leží diagonálně. Díky tomu, na rozdíl od tricopteru, nepotřebuje natáčecí rotor, který by vyrovnával momenty. Pro rotaci podle svislé osy modelu se pouze upraví rychlost otáčení jednotlivých rotorů. Rotorům se směrem otáčení shodným se směrem požadované rotace se zvýší otáčky a zbývajícím se sníží. Výsledkem této změny je nenulový součet všech čtyř momentů a tedy i tendence quadrocopteru rotovat. Díky umístění souhlasných rotorů diagonálně dokonce nedojde ani k změně stabilní polohy.

Obrázek 2.3: Quadrocopter [28].

2.5 Vícerotorové coptery

Copter se dá sestavit s teoreticky libovolným počtem rotorů vyšším než dva. Jejich řízení ale je založeno na stejném principu. Má-li copter lichý počet rotorů, musí být jeden z nich natáčecí a ovládání je podobné tricopteru, je-li počet sudý, musí mít polovina rotorů opačný smysl rotace a řízení je podobné quadrocopteru. Důvody pro stavbu vícerotorových copterů jsou především požadavky na vyšší nosnost při zachování přibližně stejné ceny. Zvýšením počtu rotorů se také zvyšuje stabilita. Na druhou stranu přínosy nejsou až zas tak veliké, aby částečně zvýšené náklady vynahradily. S vyšším počtem rotorů stoupá energetická náročnost a je tedy nutné zvýšit kapacitu zdroje nebo počítat s nižším letovým časem.

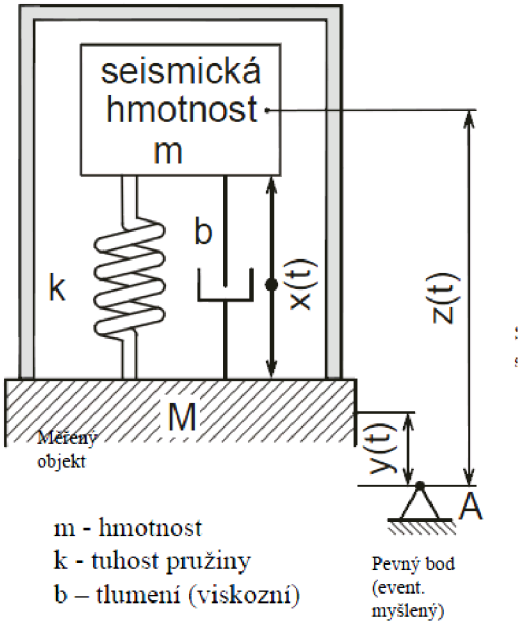
3 Navigační systém

Pro stabilizaci a řízení tricopteru je nutné zjišťovat aktuální polohu, náklon, rychlost a další proměnné. K tomu slouží Inerciální navigační systém (INS). Základní princip INS vychází z měření úhlových rychlostí a hodnot zrychlení k vypočítání změny polohy vůči počátečnímu bodu. Při znalosti polohy počátečního bodu je možné přesně určit současnou polohu. Ačkoliv myšlenka je jednoduchá, její realizace je technicky velmi obtížná. Hlavním problémem je právě přesné zjištění zrychlení a úhlové rychlosti. Žádný reálný senzor nemá nekonečně vysokou přesnost a integrací se chyba akumuluje. V historii letectví existuje několik případů, kdy tato chyba způsobila pád letadla. INS lze rozdělit do dvou kategorií v závislosti na způsobu měření. První kategorií jsou kardanové inerciální navigační systémy. Tyto systémy používají gyroskopicky stabilizovanou měřící plošinu. V závislosti na hodnotách naměřených gyroskopy upravují servomotory náklon stabilizované plošiny tak, aby byla vždy vyrovnána. Na plošině jsou umístěny akcelerometry, které měří zrychlení. Integrací této hodnoty je vypočítána rychlost a další integrací i pozice. Výhodou tohoto systému je možnost přímého použití hodnot z akcelerometrů bez nutnosti přepočítávání. Za největší nevýhodu lze považovat hmotnost a mechanickou složitost. Právě kvůli hmotnosti a ceně nelze použít tento systém ke stabilizaci copterů [4].

Druhou skupinou jsou bezkardanové stabilizační systémy. Gyroskopy neslouží ke stabilizaci plošiny, ale z naměřených úhlových rychlostí jsou integrací vypočítány úhly a těmi jsou upravovány všechny další naměřené rychlosti, tedy i další úhlové rychlosti. Při znalosti úhlů náklonu lze pomocí goniometrických operací zjistit hodnotu v inerciální soustavě pomocí rotační matice.

3.1 Akcelerometr

Každý akcelerometr, bez ohledu na použitou technologii, využívá stejný princip. K vnější části, která je spojena s předmětem, jehož zrychlení měříme, je pomocí pružného prvku připojena seismická hmotnost. Mimo to je ještě s vnější částí spojena pomocí tlumiče. Celá soustava, kterou schématicky zobrazuje obrázek 3.1 a lze popsat rovnicemi 3.1 až 3.3 [4][5].



Obrázek 3.1: Schéma akcelerometru [5].

$$m\frac{d^2z}{dt^2}+b\frac{dx}{dt}+kx=0 \tag{3.1}$$

$$z(t)=x(t)+y(t) \tag{3.2}$$

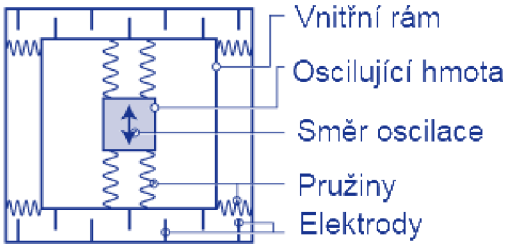
$$\frac{x_0}{y_0}=\frac{\omega^2}{\omega_0^2-\omega^2} \tag{3.3}$$

Z amplitudové charakteristiky 3.3 lze vyvodit následující závěr. Pokud je ω mnohem menší ω_0 , pak výchylka x_0 přímo odpovídá zrychlení. V opačném případě lze prohlásit, že x_0 se rovná y_0 , tedy akcelerometr měří pouze kmitání pouzdra okolo nehybné seizmické hmoty.

Použitý akcelerometr využívá diferenciálního kapacitního principu měření výchylky seizmické hmoty. S pomocí MEMS technologie, která slučuje mikroelektronickou a mikromechanickou část senzoru, je vytvořen dvojitý kondenzátor, jehož prostřední elektroda je přímo spojena se seizmickou hmotou.

3.2 Gyroskop

MEMS gyroskop měří velikost úhlové rychlosti s využitím Coriolisovy síly [5][6]. Schématické znázornění struktury takovéhoto senzoru ukazuje obrázek 3.2.



Obrázek 3.2: Schéma gyroskopu [6].

Základem je periodicky kmitající hmota o přesné hmotnosti, která kmitá uvnitř rámu na pružinách. Pro správnou funkci musí být směr kmitání kolmý ke směru měřené rotace. Na takto kmitající hmotu působí virtuální Coriolisova síla s velikostí přímo úměrné velikosti úhlové rychlosti. Tato síla působí na pružiny mezi vnitřním a vnějším rámem. Tento pohyb je zachycen pomocí měřících plošek. K měření se využívá kapacitního principu – plošky jsou elektrodami kondenzátoru s vzduchovým dielektrikem. Kapacita je tedy úměrná vzdálenosti elektrod a tím i rychlosti otáčení. Tento údaj je nutné změřit a převést do digitální podoby. Takto popsaný senzor samozřejmě měří pouze úhlovou rychlost v jedné rovině. Pro měření rotace ve všech 3 osách musí být tento měřící prvek obsažen v senzoru třikrát a pokaždé natočený kolmo k měřené ose.

3.3 Magnetometr

Magnetometr je senzor, který měří velikost magnetického pole. Pro účely navigace je nutné použít senzory, které jsou primárně schopné detekovat geomagnetické pole Země. To je možné reprezentovat jako magnetický dipol, který je odkloněn asi o hodnotu 11° od osy otáčení planety. Magnetometry jsou velmi náchylné na okolní rušení jakýmkoliv blízkým magnetickým polem. Je tedy otázkou, do jaké míry budou data senzoru spolehlivá v zarušeném prostředí [4][5].

V závislosti na technologii lze magnetometry rozdělit do několika skupin. Asi nejzákladnější a každému známý je mechanický kompas. Volně otáčející se magnetická střelka si neustále zachovává konstantní natočení ve směru sever-jih.

Magnetorezistivní kompas využívá změny elektrického odporu feromagnetických materiálů. Velikost odporu přímo závisí na velikosti a směru magnetického pole. Změna odporu je měřena a převáděna na hodnotu magnetického pole. K výrobě kompasů se využívají senzory označené AMR (Anizotropic magnetorezistive). Jejich výhodou je relativně vysoká citlivost oproti ostatním senzorům.

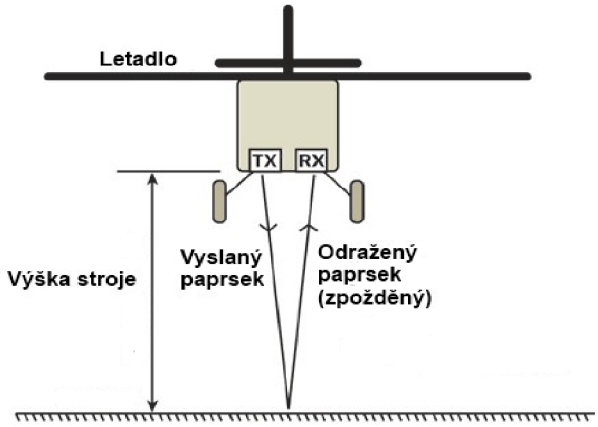
Kompasy založené na Hallově efektu využívají působení Lorentzovy síly na elektrony. Protéká-li polovodičem, který se nachází v magnetickém poli, proud a svírá-li vektor magnetické pole s vektorem proudu nenulový úhel, budou elektrony působením Lorentzovy síly posunovány k jednomu okraji polovodiče. Tím vznikne nerovnováha a na okrajích bude možné naměřit napětí.

3.4 Výškoměr

Výškoměr slouží k určení výšky vznášedla vzhledem ke vztažnému bodu. Jako vztažný bod se používá buďto povrch pod strojem nebo mořská hladina. Výškoměry využívají několik různých principů. Nejběžnějším je barometrický výškoměr. Ten využívá k určení výšky rozdíl atmosférického tlaku, který se stoupající výškou klesá. Pro správné určení výšky je nutné znát referenční tlak v místě vztažného bodu a tuto hodnotu aktualizovat v závislosti na změně atmosférických podmínek. Výška je pak vypočítána podle vzorce 3.4. Přesnost výpočtu je omezena rozlišením tlakového senzoru [4].

$$z = 16000 \cdot (1 + 0{,}004 \cdot t_m) \cdot \frac{p_0 - p_1}{p_0 + p_1} \tag{3.4}$$

Druhým způsobem měření výšky je použití aktivního radarového výškoměru [7]. Aktivní výškoměry vysílají radiový signál směrem k povrchu, kde se signál odrazí a vrátí se zpět a je zachycen přijímačem. Schématicky je tento princip ukázán na obrázku 3.3



Obrázek 3.3: Princip radarového výškoměru.

Výška se následně vypočítá na základě doby, kterou paprsek potřeboval k uražení cesty k povrchu a zpět. Pokud je výška příliš velká, paprsek zeslábne natolik, že již nebude zachycen přijímačem. Proto se radarové výškoměry používají pouze v malých výškách pro přesné určení výšky například při přistání. Pro potřeby copterů je možné použít ultrazvukový dálkoměr.

$$zpoždění = \frac{2 \cdot výška}{rychlost\ šíření} \tag{3.5}$$

4 Mechanická konstrukce

Součástí této diplomové práce je také návrh a tvorba vhodné konstrukce vznášedla. Jako první a nejdůležitější krok této fáze je volba vhodného měřítka a s tím i použitelných technologií a materiálů. Při volbě měřítka jsem primárně vycházel z rozměrů, dostupnosti potřebného stavebního materiálu a také cenové přijatelnosti. Jako nejvhodnější jsem nakonec vybral modelářské potřeby. Tato volba přináší mnohé výhody, ale zároveň i nevýhody. Mezi výhody lze označit především dostupnost a jednoduché použití. Za nevýhodu lze považovat určité tolerance ve výrobě elektronických a mechanických prvků a tím i pravděpodobně nižší přesnost stabilizace. Při návrhu jsem vycházel z poznatků leteckých modelářů, kteří mají s podobnými vznášedly zkušenosti [8][9] [10].

4.1 Kostra modelu

Tricopter tvoří tři motory umístěné do vrcholů rovnostranného trojúhelníka. Prakticky se však konstrukce realizuje jako obrácené písmeno „Y“ se stejně dlouhými větvemi. Vlastní konstrukci lze rozdělit do čtyř dílčích částí. Hlavní tělo tricopteru, které obsahuje řídící elektroniku, senzorický systém, baterku a případný další hardware jako např. kameru, bude umístěn uprostřed a bude udržovat všechny části vcelku. Druhou částí jsou tři nosníky svírající navzájem úhel 120°, které budou spojovat úchyty motorů s tělem modelu. Další částí jsou úchyty samotných motorů. Poslední a pravděpodobně konstrukčně nejsložitější je kloub, který bude umožňovat náklon zadního motoru podél vodorovné osy tricopteru.

Délka nosníků záleží na požadovaných rozměrech tricopteru. S vyšší vzdáleností motorů od středu by měl být model stabilnější vůči zásahům do řízení. Nosníky je možné vytvořit buďto z uhlíkových trubek nebo ze smrkových lišt. Uhlíkové trubky jsou k dostání o maximálním průměru 12 mm s různou tloušťkou stěn. Pro první pokusy jsem použil trubky s vnějším průměrem 10 mm a vnitřním 8 mm. První problém je upevnění nosníků k tělu i k motorům. Jelikož se jedná o relativně tenkostěnné trubky o malém průměru, není možné je provrtat a uchytit šroubem. Při použití lepidla není možné model rozebrat pro přepravu, což by vzhledem k jeho rozměrům znamenalo velkou komplikaci. Alternativou, kterou jsem použil, bylo vyplnit kraje trubek tavným lepidlem, aby nedošlo k poškození, a následně je stáhnout mezi obě desky těla. Nevýhodou je obtížné zarovnání a protáčení nosníků. Také třecí plocha je příliš malá, takže se nosník může při velkém zatížení pohnout.

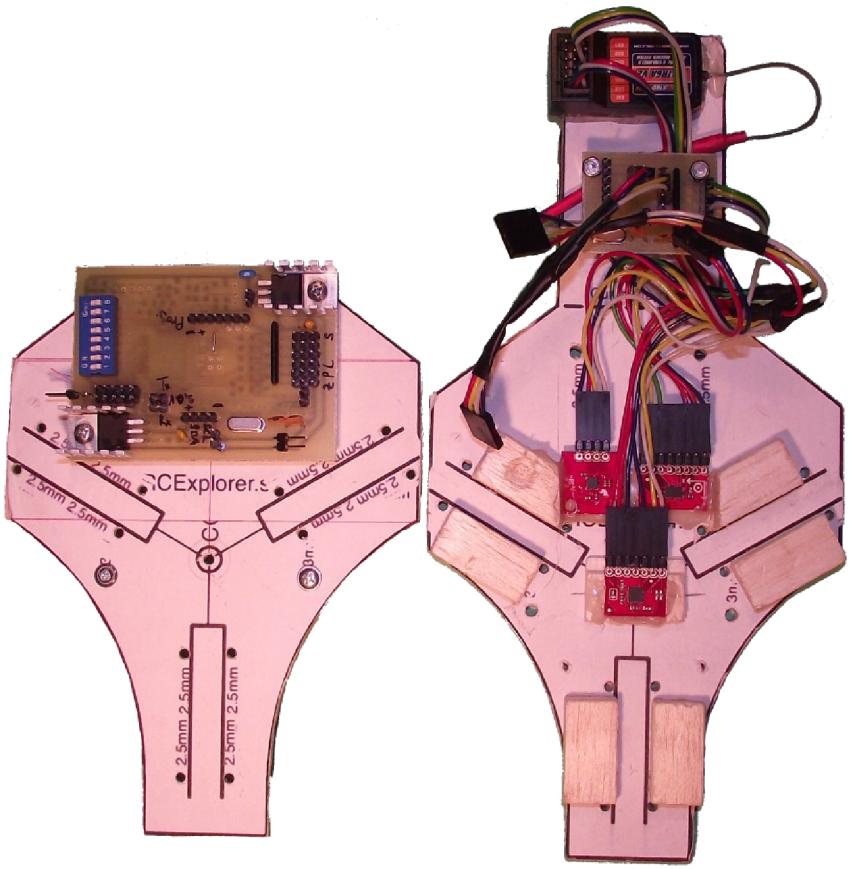
Smrkové nosníky jsou k dostání o různých průřezech. Použité nosníky mají čtvercový průřez o straně 10 mm. Výhodou smrkových nosníků je snadnější zkracování na potřebnou délku

a jednodušší možnosti uchycení díky jejich čtvercovému tvaru. Značnou nevýhodou je vyšší pružnost použitého materiálu, takže dochází k mnohem většímu přenosu vibrací z motorů k senzorům. Hlavní výhodou je však nižší pořizovací cena, lepší dostupnost a také rychlejší výměna poškozeného nosníku.

Délka předních nosníků je 500 mm. Zadní nosník musí být zkrácen na takovou délku, aby byla zachována stejná vzdálenost těžiště a osy motoru jako je u předních.

4.2 Tělo modelu

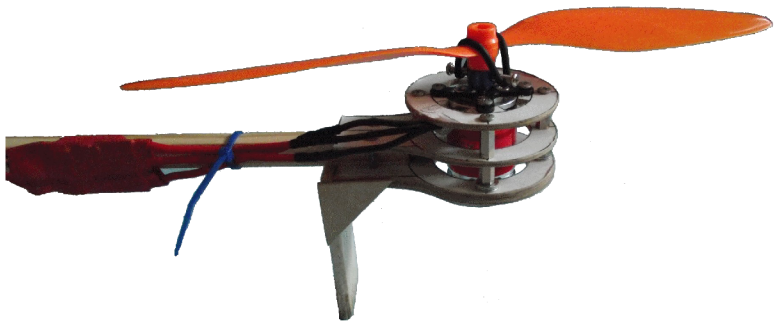
Tělo modelu je hlavním prvkem, který spojuje nosníky motorů a které nese většinu řídící elektroniky. Tělo tvoří dvě identické překližkové desky, které jsou na různých místech spojeny šrouby. Mezi deskami je mezera, jejíž výška odpovídá největšímu rozměru nosníků. Na spodní části těla může být uchycena baterka. Uchycení baterky musí umožňovat posun baterky, díky kterému bude možné tricopter před startem vyvážit. Na horní desku a do mezery mezi deskami je umístěna řídící elektronika včetně senzorů. Senzory jsou umístěny blízko sebe v místě těžiště. Jejich uchycení v tomto místě by je mělo uchránit při případném pádu modelu před poškozením.



Obrázek 4.1: Horní (vlevo) a dolní(vpravo) část těla tricopteru.

4.3 Uchycení motorů

Modelářské motory se vyrábí ve dvou provedeních – s uchycením vpředu a nebo vzadu. Uchycení vpředu předpokládá zabudování motoru do modelu tak, že nosná konstrukce je mezi tělem motoru a vrtulí. Uchycení vzadu naopak předpokládá přišroubování motoru ke konstrukci, která je za motorem, tedy tělo motoru je mezi nosnou konstrukcí a vrtulí. Pro stavbu tricopteru by byly vhodnější právě motory s uchycením vzadu. Pro uchycení motoru by pak sloužily dvě dřevotřískové desky, ke kterým by byl motor přišroubován. Jedna z desek by byla umístěna pod nosníkem, druhá nad ním. Přitažením obou částí k sobě by pak vnikl tlak, kterým by desky působily proti sobě na nosník a ten by držel i bez použití lepidla. Tento typ motorů ale je modeláři považován za „nemodelový" a je zvlášť v České republice obtížné tyto motory sehnat. Proto jsem použil motory s uchycením vpředu.



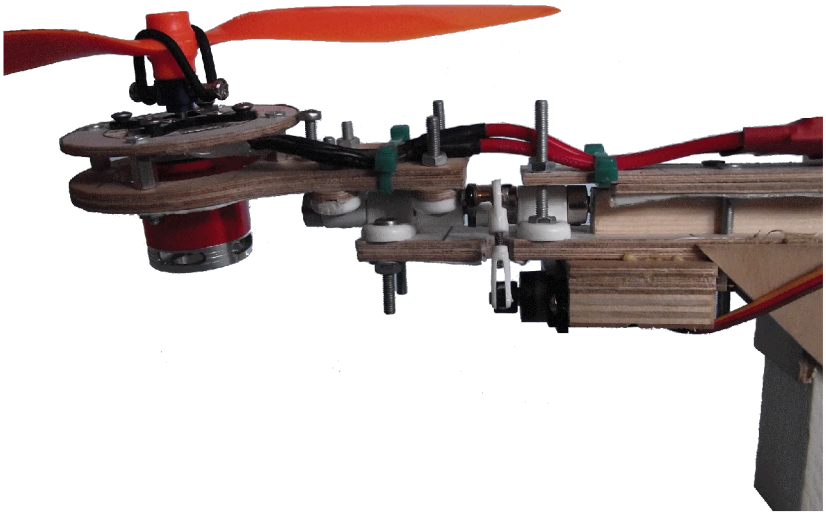
Obrázek 4.2: Uchycení motoru.

Mnou vybrané motory obsahovaly bohatou sadu příslušenství včetně kříže pro zpětnou montáž. Tento kříž umožňuje uchycení motoru obráceným směrem. Navíc nebrání průchodu vzduchu hnaného vrtulí skrz tělo motoru a tím ho ochlazuje. Použitím kříže se přesunuly montážní otvory mimo tělo motoru, nicméně ne dostatečně daleko pro použití distančních sloupků či motorového lože, jelikož by rotační plášť s největší pravděpodobností dřel o tyto distanční sloupky. Proto jsem upravil tvar úchytů motorů tak, aby hlavní část tvořila mezikruží s vnitřním průměrem dostatečně velkým na to, aby mohl motor rotovat, nedřel a měl dostatečný přísun vzduchu pro chlazení. Na horní části jsem ponechal čtyři vnitřní výběžky, pomocí kterých je kříž přišroubován. Tím jsem dosáhl dostatečně pevného uchycení motorů. Problémem se však ukazuje uchycení vrtule. Takto by vrtule byla těsně nad vrchní hranou nosníku. U předních motorů by se jednalo o ideální uchycení, u zadního je však daná výška příliš nízká. Vrtule by při natočení dřela konstrukci kloubu nebo jiné konstrukční prvky. Bylo tedy nutné zvýšit uchycení motoru. Ve výsledku není motor přichycen k vrchnímu dřevotřískovému

výřezu, ale k vložce, která je přes distanční sloupky přišroubována k úchytům. Tímto je zadní vrtule dostatečně zvednuta a nehrozí kontakt. Pro účely řízení tricopteru a zjednodušení modelu je nutné, aby byly motory ve stejné výšce, takže jsem toto uchycení použil i u zbylých dvou rotorů.

4.4 Otočný kloub

Konstrukce tricopteru vyžaduje, aby bylo možné zadní rotor natáčet podél hlavní osy modelu a tím vytvářet dodatečný moment v bočním směru. Konstrukce tohoto kloubu musí být dostatečné pevná na to, aby nevytvářela dodatečné vibrace, ale zároveň s ní musí být schopné pohnout modelářské mikroservo. V první konstrukci jsem použil součástky ze známé stavebnice Merkur. Osa byla volně uchycena v úrovni nosníku na jedné straně, na straně motoru pak pevně. Servo umístěné na nosníku pak otáčelo celou osičkou a tím natáčelo motor. Bohužel stavebnice Merkur používá vlastní rozměry pro všechny své součástky a tolerance rozměrů je příliš veliká. Ve výsledku pak osička na volné straně měla až přílišnou vůli v úchytu a kloub při určitých otáčkách motoru začal vibrovat. V druhé konstrukci jsem použil modelářské úchyty podvozku s průměrem 4 mm a čtyřmilimetrovou dutou mosaznou trubičku. Výsledný kloub má mnohonásobně lepší vlastnosti než původní.



Obrázek 4.3: Otočný kloub – detail.

4.5 Motory

K pohonu rotorů tricopteru jsem použil modelářské střídavé motory, které jsou v anglické terminologii nazývány „brushless outrunners“. Tyto motory jsou konstruovány jako bezkartáčové, aby nedocházelo k opotřebování kartáčků provozem. Motorky mají prohozený rotor se statorem.

Na rozdíl od běžných motorků, které mají jako rotor elektromagnet a statorem bývá permanentní magnet, u těchto motorů je permanentním magnetem stator. Navíc je zaměněna i jejich pozice. Střed motoru je pevně uchycen ke konstrukci modelu a plášť je spojen s hřídelí a spolu s ní rotuje. Otáčky motoru pak závisí na napětí, kterým je motor napájen. U každého motoru lze dohledat konstantu, která udává přibližný počet otáček na jeden volt napětí. U použitých motorů Turnigy 2217 je tato konstanta přibližně 1000 ot/v.

K buzení motoru je potřeba střídavého napětí s regulovatelnou amplitudou. Je tedy nutné vytvořit budič, který by byl schopen generovat vhodný průběh napětí na všech třech vývodech a zároveň hlídat, aby byly fáze na jednotlivých vodičích korektně posunuté. Stavba takovéhoto budiče však přesahuje rozsah této diplomové práce. Alternativním řešením je použití speciálního budiče, který používají i modeláři ve svých modelech. Tyto budiče jsou nazývány „ESC“ (*Electronic Speed Controler*). Jejich funkce je jednoduchá – ze stejnosměrného napětí vytváří střídavé napětí o požadované frekvenci a amplitudě v závislosti na vstupním signálu. ESC je řízeno signálem, který používají modelářská serva i přijímače a bude popsán dále v textu.

5 Hardware řídícího systému

Ačkoliv jsou coptery díky své jednoduché konstrukci lépe ovladatelné, není pro operátora reálné přímé ovládání pouhou změnou tahu jednotlivých motorů. Je nutné navrhnout a vytvořit alespoň základní řídící software, který bude převádět povely operátora na tah motorů. V závislosti na zvolené míře abstrakce pak bude operátor zadávat například momenty rotace nebo rychlosti v jednotlivých směrech a podobně. Navíc při letu na copter působí externí síly, které jej vyvádí ze stabilní polohy. Může to být poryv větru nebo náraz volně se pohybujícího copteru do předmětu. Také motory a regulátory způsobují nestabilitu. To je způsobeno jednak drobnými rozdíly při výrobě a také jistou nestabilitou a tolerancí součástek použitých v regulátorech.

Jádrem řídícího systému je mikrokontrolér, který zpracovává naměřené údaje a počítá nové výstupní hodnoty. Mezi hlavní požadavky na tento mikrokontrolér patří dostatečný výpočetní výkon pro zpracování stabilizačního algoritmu, hardwarová podpora komunikační sběrnice I2C a USART pro komunikaci se senzory a nastavování parametrů stabilizačního algoritmu a v neposlední řadě nízká pořizovací cena s vysokou dostupností. Za vhodné se dá považovat i podpora výpočtu desetinných čísel ve formátu s plovoucí desetinnou čárkou. Těmto parametrům nejlépe odpovídá mikrokotrolér z rodiny AVR ATmega. Z této rodiny je možné vybrat několik mikrokontrolérů, které vyhovují výše uvedeným požadavkům. Pro účely vývoje jsem se rozhodl použít ATMega128A, především kvůli velkému množství rozšiřujícího hardware a velké paměti. Po dokončení vývoje je možné zvolit i levnější mikrokontroléry, které budou vyhovovat požadavkům výsledného software.

V průběhu vývoje software jsem objevil problém se zvoleným mikrokontrolérem. Protože zvolený mikrokontrolér umožňuje připojení externí paměti SRAM, jsou piny rozšiřujícího hardware sdíleny a tím použití jedné periferie vylučuje použití druhé. Po zapojení komunikačních sběrnic USART, I2C a konektoru pro programátor již nebylo možné použít přerušovací vstupy mikrokontroléru pro čtení signálů z přijímače RC. Nejjednodušším řešením tohoto problému bylo použití koprocesoru, který by signál převáděl na digitální údaj a ten by řídící mikrokontrolér četl spolu s údaji ze senzorů po sběrnici I2C. Použitý koprocesor je také také rodiny AVR, konkrétně ATmega48.

5.1 Použité senzory

Ke stabilizaci a snímání polohy a pohybu tricpoteru je nutné snímat minimálně základní veličiny jako úhlová rychlost a zrychlení. Pro stabilizaci letu je nutný gyroskop, bez kterého není možné ani základní řízení tricopteru. Ostatní senzory jsou volitelné a umožňují pokročilejší funkce stabilizace. Základním krokem při volbě senzorů je rozhodnutí, zda-li použít jednotlivé senzory nebo

sofistikovaný senzorický modul, který by kombinoval více senzorů a zároveň byl vybaven procesorem pro filtraci a zpracování naměřených údajů. Vzhledem k faktu, že mým cílem je navrhnout a vytvořit řídící systém pro tricopter, který by byl cenově dostupný leteckým modelářům, zvolil jsem první variantu, která je pro účely základní stabilizace letu dostačující. Druhým krokem při volbě senzorů je výběr komunikačního rozhraní. Základním rozdělením je rozdělení na analogové senzory a senzory s digitálním výstupem. Výhodou analogových senzorů je jednoduchá filtrace šumu ve výstupním signálu pomocí dolní propusti ve formě RC článku. Nevýhodou je vysoká náchylnost na nestabilitu napájecího napětí vinou regulátoru, potřeba AD převodníku pro digitalizaci každého z výstupních signálů. Na druhou stranu i u senzorů s digitální komunikací se mohou vyskytnout problémy. Především je to potřeba hardwarového řadiče pro daný typ sběrnice. Dále je nutné zabezpečit bezchybnou komunikaci nebo zotavení se z chyb při přenosu. Při volbě senzorů je nutné také vybírat takové senzory, které komunikují po stejné sběrnici, jinak by stoupaly hardwarové požadavky na řídící mikrokontrolér za únosnou mez.

5.1.1 Akcelerometr

Pro snímání zrychlení tricopteru jsem zvolil senzor ADLX345 [14]. Akcelerometr ADLX345 je tříosý akcelerometr vyrobený MEMS technologií od firmy Analog Devices. Senzor měří jak statické zrychlení vytvářené gravitačním zrychlením země, tak dynamické zrychlení vyvolané nárazem nebo změnou rychlosti pohybu. Rozlišení senzoru je 4 mg s maximálním rozsahem ±16 g. Senzor umožňuje komunikaci pomocí sběrnice SPI nebo sběrnice I2C. Adresu zařízení je možné upravit – pomocí pinu se upravuje LSB adresy. Naměřené hodnoty jsou uloženy v 16 bitových registrech jako znaménková čísla ve dvojkovým doplňkem.

Tabulka 5.1 zobrazuje adresy jednotlivých registrů a jejich význam. Ze všech dostupných registrů jsou pro účely měření potřebné pouze registry 0x32 až 0x37, které obsahují výstupní hodnoty, a dále registry pro nastavení parametrů na adresách 0x2C, 0x2D, 0x31 a 0x38. Na adrese 0x2C se nastavuje spotřeba senzoru a parametry filtru. V režimu snížené spotřeby se zvyšuje šum senzoru. Jako optimální se jeví ponechání výchozí hodnoty. Registr *POWER_CTL* na adrese 0x2C určuje režim, v jakém se právě senzor nachází. Zapsáním hodnoty 0x08 dojde k nastavení bitu *MEASURE* a senzor přechází do režimu aktivního měření. Registr *DATA_FORMAT* s adresou 0x31 určuje formát výstupních dat a tím i maximální rozlišení senzoru. Nejnižší dva bity určují rozlišení senzoru, třetí bit rozlišuje, zda-li výstupní data budou zarovnána vlevo nebo budou hodnot uloženy jako znaménkově rozšířené šesnáctibitové číslo. Další bit umožňuje vynucení plného rozsahu senzoru. Zbylé bity určují, zda-li mají být výstupní hodnoty negovány a formát komunikace, pokud bude použito rozhraní SPI. Pro přepnutí senzoru do plného rozlišení a komunikaci ve formátu, který je běžný pro interpretaci čísel v jazyce C, je nutné do registru zapsat hodnotu 0x08. Posledním

registrem, který ovlivňuje základní chování senzoru, je *FIFO_CTL*. Tento registr určuje, jaký způsobem budou načtená data předávána komunikačnímu rozhraní. Senzor má k dispozici frontu o velikosti až 32 vzorků. Použití fronty by znamenalo číst data se stejnou frekvencí, s jakou jsou generována, nebo počítat s faktem, že načtená data budou zastaralá. Na druhou stranu po vyřazení fronty hrozí nebezpečí, že data budou aktualizována v průběhu čtecí operace. Pokud by aktualizace nastala mezi načítáním různých složek zrychlení, nebyl by toto problém. Pokud aktualizace nastane mezi načtením spodního bytu a horního bytu, budou právě načtená data nesmyslná. Je tedy nutné, načítat data minimálně třikrát v těsném sledu a porovnáním vzorků určit dvě shodné hodnoty. Právě tento způsob je zvolen pro načítání hodnot. Tato konfigurace se nastavuje zapsáním hodnoty 0x00 do registru *FIFO_CTL*.

Adresa (hex)	**Název**	**Popis**
0x00	DEVID	Identifikace zařízení
0x1D	THRESH_TAP	Dolní mez pro vyvolání přerušení při nárazu
0x1E	OFFX	Kalibrační hodnota pro vyvážení senzoru v ose X
0x1F	OFFY	Kalibrační hodnota pro vyvážení senzoru v ose Y
0x20	OFFZ	Kalibrační hodnota pro vyvážení senzoru v ose Z
0x21	DUR	Minimální doba překročení prahu pro vyvolání přerušení
0x22	LATENT	Minimální doba mezi událostmi
0x23	WINDOW	Maximální doba čekání na událost
0x24	THRESH_ACT	Minimální hodnota pro detekci začátku události
0x25	THRESH_INACT	Maximální hodnota pro detekci konce události
0x26	TIME_INACT	Minimální doba, po kterou nebude detekce prováděna
0x27	ACT_INACT_CTL	Konfigurační registr pro detekce
0x28	THRESH_FF	Práh pro detekci volného pádu
0x29	TIME_FF	Minimální doba pro detekci volného pádu
0x2A	TAP_AXES	Maska pro povolení detekce na jednotlivých osách
0x2B	TAP_ACT_STATUS	Stavový registr detektoru
0x2C	BW_RATE	Nastavení low-pass filtru a režimu snížené spotřeby
0x2D	POWER_CTL	Nastavení pracovních režimů senzoru
0x2E	INT_ENABLE	Nastavení událostí vyvolávajících přerušení
0x2F	INT_MAP	Mapování jednotlivých událostí na přerušovací piny
0x30	INT_SOURCE	Stavový registr indikující stav jednotlivých přerušení
0x31	DATA_FORMAT	Nastavení rozlišení a způsobu komunikace
0x32	DATAX0	Datový byte osy X – dolní byte hodnoty
0x33	DATAX1	Datový byte osy X – horní byte hodnoty
0x34	DATAY0	Datový byte osy Y – dolní byte hodnoty
0x35	DATAY1	Datový byte osy Y – horní byte hodnoty
0x36	DATAZ0	Datový byte osy Z – dolní byte hodnoty
0x37	DATAZ1	Datový byte osy Z – horní byte hodnoty
0x38	FIFO_CTL	Konfigurační registr výstupní FIFO fronty
0x39	FIFO_STATUS	Stavový registr výstupní FIFO fronty

Tabulka 5.1: Popis konfiguračních registrů akcelerometru ADXL345 [14].

Další registry umožňují pokročilé funkce senzoru jako jsou detekce zrychlení překračující předem nastavený práh po předem stanovenou dobu nebo detekce volného pádu. Každá z těchto pokročilých funkcí umožňuje generovat přerušení na speciálním vodiči. Tyto funkce jsou pro účely měření zrychlení nepotřebné, a proto není třeba je nastavovat.

5.1.2 Gyroskop

Gyroskop ITG-3200 je tříosý MEMS gyroskop založený na diferenciálním kapacitním snímání [15]. Senzor také obsahuje integrovaný senzor teploty, jehož výstup je dostupný v registrech. Naměřenou teplotu lze pak použít pro korekci úhlových rychlostí podle doporučení výrobce. Rozlišení gyroskopu je 14,375 bitů/°/s a maximální hodnota je 2000°/s. Stejně jako akcelerometr i tento senzor umožňuje komunikaci po sběrnici I2C i SPI. Také umožňuje volbu adresy pomocí hardwarové propojky. Na rozdíl od předchozího modulu, zde není pin vyveden až na lištu, ale je třeba propojit plošky přímo na DPS. Navíc ale obsahuje konfigurovatelnou dolní propusť pro filtrování dat.

Adresa (hex)	**Název**	**Popis**
0x00	WHO_AM_I	Identifikace zařízení
0x15	SMPLRT_DIV	Nastavení výstupní frekvence hodnot
0x16	DLPF_FS	Konfigurace rozlišení senzoru a low-pass filtru
0x17	INT_CFG	Konfigurace generátoru přerušení
0x1A	INT_STATUS	Stavový registr generátoru přerušení
0x1B	TEMP_OUT_H	Dolní byte teploty
0x1C	TEMP_OUT_L	Horní byte teploty
0x1D	GYRO_XOUT_H	Datový byte osy X – horní byte hodnoty
0x1E	GYRO_XOUT_L	Datový byte osy X – dolní byte hodnoty
0x1F	GYRO_YOUT_H	Datový byte osy Y – horní byte hodnoty
0x20	GYRO_YOUT_L	Datový byte osy Y – dolní byte hodnoty
0x21	GYRO_ZOUT_H	Datový byte osy Z – horní byte hodnoty
0x22	GYRO_ZOUT_L	Datový byte osy Z – dolní byte hodnoty
0x3E	PWR_MGM	Nastavení režimu činnosti senzoru

Tabulka 5.2: Popis konfiguračních registrů gyroskopu ITG-3200 [15].

Pro korektní činnost gyroskopu je třeba nastavit konfigurační registry. Adresy jednotlivých registrů jsou zobrazeny v tabulce 5.2. Po zapnutí je doporučené vynutit reset všech registrů nastavením bitu H_RESET v registru Power Management. Následně se musí nastavit všechny registry tak, aby senzor fungoval ve správném režimu. V registru Power Management se dále nastavuje zdroj hodinového signálu pro ADC, přechod do režimu spánku nebo vypnutí ADC jednotlivých os. Výchozí hodnota registru po resetu ponechává senzor v požadovaném režimu, takže není nutné jeho hodnotu dále upravovat. Registr DLPF na adrese 0x16 určuje přesnost senzorů. Současná revize senzoru umožňuje pouze měření v plném rozlišení, které je nutné v tomto registru nastavit. Dalším nastavením je aktivace a volba parametrů filtru naměřených dat. Pro nastavení plného rozšíření je třeba zapsat hodnotu 0x18. Podle zvolené konfigurace filtru může být zapsána

hodnota 0x18 až 0x1E. Naměřené hodnoty je možné přečíst z registrů na adresách 0x1D až 0x22. Na adresách 0x1B a 0x1C je namapován výstup ADC převodníku měřícího výstup teplotního senzoru. Zbylé registry slouží pro nastavování událostí, při kterých bude vygenerována sestupná hrana na přerušovacím pinu, a pro zjištění, která událost přerušení vyvolala.

5.1.3 Magnetometr

HMC5883L je magnetometr vyrobený firmou Honeywell [16]. Obvod pracuje na principu magneto-rezistivní technologie a pro zpracování naměřených výsledků je vybaven ASIC, který obsahuje zesilovač a dvanáctibitový ADC. Celková přesnost senzoru je jeden až dva stupně. Stejně jako předchozí dva senzory i tento pracuje s napětím 3 V a komunikuje po sběrnici I2C.

Adresa (hex)	Název	Popis
0x00	CONFIG_A	Konfigurační registr A
0x01	CONFIG_B	Konfigurační registr B
0x02	MODE	Konfigurace způsobu měření
0x03	DATA_X_HIGH	Datový byte osy X – horní byte hodnoty
0x04	DATA_X_LOW	Datový byte osy X – dolní byte hodnoty
0x05	DATA_Z_HIGH	Datový byte osy Y – horní byte hodnoty
0x06	DATA_Z_LOW	Datový byte osy Y – dolní byte hodnoty
0x07	DATA_Y_HIGH	Datový byte osy Z – horní byte hodnoty
0x08	DATA_Y_LOW	Datový byte osy Z – dolní byte hodnoty
0x09	STATUS	Stavový registr
0x0A	IDENT_A	Identifikační registr A
0x0B	IDENT_B	Identifikační registr B
0x0C	IDENT_C	Identifikační registr C

Tabulka 5.3: Popis konfiguračních registrů magnetometru HMC5883L [16].

Rozložení registrů a jejich adresy zobrazuje tabulka 5.3. Základní nastavení režimu činnosti se provádí v registru *MODE*. Nejnižší bit určuje, zda-li proběhne pouze jedno měření nebo měření bude probíhat neustále. Druhý bit určuje, zda-li zařízení je přepnuto do pracovního režimu nebo je v režimu se sníženou spotřebou. Konfigurační registry *A* a *B* umožňují nastavení počtu vzorků, které budou zprůměrovány do výstupní hodnoty, frekvence aktualizace výstupních dat a zesílení výstupního signálu. Výhodou senzoru HMC5883L je jeho chování při čtení datových registrů. V okamžiku, kdy je přečten jeden libovolný registr, jsou všechny registry zamčeny proti změnám až do okamžiku, kdy bylo přečteno všech šest registrů. Díky tomu není nutné číst data několikrát a porovnávat, zda-li nedošlo během čtení k aktualizaci dat. Registry jsou uzamčeny proti zápisu také v okamžiku, kdy je přečten obsah registru mode. Příznak, zda-li jsou registry uzamčeny proti zápisu, lze přečíst z registru STATUS. Dalším příznakem v tomto registru je bit RDY. Tento bit signalizuje, že byly nahrána nově naměřené hodnoty do datových registrů. Je vynulován automaticky v okamžiku,

kdy začalo čtení z datových registrů nebo začal zápis nových hodnot. Identifikační registry slouží k rozpoznání typu zařízení. Jejich hodnoty jsou 'H', '4', '3'.

5.1.4 Ultrazvukový dálkoměr

Jako rádiový výškoměr slouží senzor SRF02. Tento senzor je založen na mikrokontroléru PIC 16F687. K řadě senzorů SRF0x neexistuje žádná dostupná technická dokumentace, pouze internetové stránky [17] popisující komunikaci se senzorem. Na rozdíl od všech předchozích senzorů, SRF vyžaduje ke své činnosti napájecí napětí 5 V. Senzor dokáže komunikovat pomocí sběrnice I2C i asynchronního sériového kanálu. Volba režimu komunikace se volí vývodem MODE.

Minimální vzdálenost, kterou je senzor schopen naměřit, je v rozmezí 15 – 18 cm podle teploty okolí. Maximální vzdálenost v přímém směru je 249 cm. V závislosti na odklonu objektu od hlavní vyzařovací osy se maximální vzdálenost snižuje.

Adresa (hex)	**Název**	**Popis**
0x00	Command / Version	Zapsání příkazu / přečtení verze software
0x02	Range High Byte	Naměřená hodnota – vyšší byte
0x03	Range Low Byte	Naměřená hodnota – nižší byte
0x04	Autotune Min High	Minimální měřitelná hodnota – vyšší byte
0x05	Autotune Min Low	Minimální měřitelná hodnota – nižší byte

Tabulka 5.4: Popis registrů ultrazvukového dálkoměru SRF02 [17].

Ovládání senzoru za pomoci sběrnice I2C se provádí zápisem požadovaného příkazu do registru *COMMAND*. Adresy registrů jsou zobrazeny v tabulce. Seznam všech možných příkazů zobrazuje tabulka. Příkazy, které jsou podbarveny šedě, jsou přípustné pouze při komunikaci pomocí UART. Zahájení měření se provede odesláním jednoho z příkazů 0x50 až 0x52. Tyto příkazy odešlou krátký impuls o frekvenci 40 kHz a vyčkají na příjem odraženého signálu. Příkazy 0x56 až 0x58 neodesílají impuls, pouze zahájí čekání na příjem impulsu odeslaného jiným senzorem. Toto chování je vhodné například pro měření vzdálenosti dvou identických robotů. Komplementárním příkazem je příkaz 0x5C, který způsobí pouze odeslání 8 impulsů s frekvencí 40 kHz, aniž by čekal na příjem odrazu.

Po zahájení měření mikrokontrolér přestává sledovat veškerou komunikaci po sběrnici až do okamžiku, kdy je měření dokončeno. Toto chování umožňuje jednoduchou detekci konce měření, ale neumožňuje komunikaci s dalšími senzory, protože by následné zapsání výstupní hodnoty poškodilo právě přenášená data. Vzhledem k době měření, které trvá až 65 ms, není vhodné připojit mikrokontrolér spolu s ostatními senzory. Druhou komplikací je fakt, že adresy, které je možné senzoru přidělit, se nacházejí v intervalu adres vyhrazených pro budoucí použití.

Adresa (hex)	Popis
0x50	Měření vzdálenosti – výsledek v palcích
0x51	Měření vzdálenosti – výsledek v centimetrech
0x52	Měření vzdálenosti – výsledek v mikrosekundách
0x53	Měření vzdálenosti s odesláním výsledku – výsledek v palcích
0x54	Měření vzdálenosti s odesláním výsledku – výsledek v centimetrech
0x55	Měření vzdálenosti s odesláním výsledku – výsledek v mikrosekundách
0x56	Falešné měření – výsledek v palcích
0x57	Falešné měření – výsledek v centimetrech
0x58	Falešné měření – výsledek v mikrosekundách
0x59	Falešné měření s odesláním výsledku – výsledek v palcích
0x5A	Falešné měření s odesláním výsledku – výsledek v centimetrech
0x5B	Falešné měření s odesláním výsledku – výsledek v mikrosekundách
0x5C	Odeslání 8 impulsů o frekvenci 40kHz bez čekání na výsledek
0x5D	Čtení verze software
0x5E	Čtení naměřeného výsledku
0x5F	Čtení minimální měřitelné vzdálenosti
0x60	Vynucení spuštění auto-kalibrace
0xA0	První byte sekvence pro změnu adresy
0xA5	Druhý byte sekvence pro změnu adresy
0xAA	Třetí byte sekvence pro změnu adresy

Tabulka 5.5: Přehled příkazů pro SRF02 [17].

Při komunikaci pomocí UART je třeba vždy posílat sekvenci dvou bytů. Prvním bytem je adresa senzoru, druhým je příkaz dle tabulky 5.5. V tomto případě je vhodné používat některý z příkazů, které jsou v tabulce podbarveny šedě. Použití těchto příkazů způsobí automatické odeslání naměřené hodnoty zpět, jakmile bude získána. Není tedy nutné měřit čas, zda-li již byla operace dokončena, a následně číst naměřenou hodnotu.

5.2 Řízení výkonových prvků

Servo i ESC jsou řízeny signálem s pulzně šířkovou modulací (PWM) [8][36]. Pro každý prvek je veden právě jeden signál. Pro řízení serva není využita celá střída signálu, ale pouze jeho část. Perioda signálu je 20 ms, signál je tedy opakován 50 krát za vteřinu. Za platný signál je označen takový průběh, kde délka impulsu je v rozmezí 900 µs – 2100 µs. Někteří výrobci podporují i hodnoty v rozmezí 600 µs – 2 400 µs. Středovou hodnotou je puls o délce 1,5 ms. V případě, že signál nesplňuje tyto hodnoty, není výstupní chování definováno. Z provedených testů jsem zjistil, že servo přestane korigovat hodnotu výchylky, ESC odpojí a zastaví motor.

K řízení tricopteru bude třeba tento signál dekódovat (příjem signálu od RC Soupravy) i generovat (pro účely řízení akčních prvků). Generování signálu je možné dvěma způsoby. První způsob je jednodušší z hlediska řízení a také méně zatěžuje procesor, ale vyžaduje speciální hardwarovou podporu. Jedná se o generování PWM pomocí hardwarového generátoru – nejčastěji

čítač s komparátorem. Princip je následující: volně běžící čítač počítá vzestupně či sestupně (na směru příliš nezáleží) v rozsahu 0 až maximum. Při dosažení dna čítače dojde k nastavení hodnoty výstupního bitu. Jakmile dosáhne hodnota čítače shodné hodnoty jako v registru komparátoru, komparátor provede negaci signálu. Jistý problém působí nastavování nové hodnoty komparátoru. Pro následující část textu předpokládejme, že čítač čítá sestupným směrem. Pokud nově nastavovaná hodnota je nižší než stávající a k aktualizaci čítače dojde v okamžiku, kdy hodnota volně běžícího čítače překročila nově nastavovanou hodnotu, nedojde ke shodě hodnot a na výstupu tedy bude vysoká hrana celých 20 ms a signál bude s největší pravděpodobností označen za chybný. Abychom se této potencionální chybě vyhnuli, je nutné aktualizovat hodnotu komparátoru pouze v okamžiku, kdy je nová hodnota nižší než obsah volně běžícího čítače. Některé hardwarové generátory PWM obsahují i režim, kdy tuto činnost obstarávají za programátora tzv. dvojitým bufferováním. Jedná se o techniku, kdy nová hodnota je zapsána do dočasného registru a k jejímu překopírování dojde až v okamžiku, kdy hodnota volně běžícího čítače dosáhne nuly. Na druhou stranu tato technika zavádí zpoždění během aktualizace, které může dosahovat až hodnoty 20 ms. Je samozřejmé, že pro jeden signál je potřeba jeden hardwarový komparátor.

Druhý způsob generování využívá vlastností signálu. Maximální délka pulsu je 2,4 ms, což je přibližně 1/8 periody. Je tedy možné vygenerovat až 8 pulsů za jednu periodu. Ke generování 8 signálů stačí jeden časovač s jedním komparátorem. Frekvence časovače bude 8 krát vyšší – tedy 400 Hz. Na rozdíl od prvního způsobu, kde celé generování obstarával samotný hardware, zde je třeba, aby generování bylo řízeno od přerušení, což zanáší určitou nepřesnost. Princip je následující: v okamžiku přetečení je vygenerováno přerušení, které podle pomocné proměnné nastaví výstupní bit do logické 1 a nastaví hodnotu komparátoru na požadovanou délku pulsu. Jakmile dojde ke shodě v komparátoru, další přerušení vynuluje výstupní pin – zde je vhodné, aby všechny piny byly na jednom výstupním portu, což by snížilo režii – a připraví si hodnoty pro další kanál. Tento cyklus se opakuje 8 krát během jedné periody a tím dojde k vygenerování požadovaného průběhu na všech kanálech. Určitým problémem může být fakt, že hodnoty signálů jsou navzájem posunuty a také tento způsob generování vytváří určitou režii, která může zpomalovat procesor při výpočtu.

Pro řízení modelu tricopteru může být použita modelářská RC souprava. Je tedy nutné kromě generování signálu jej také měřit. Opět existují dva různé způsoby. První z nich bude využívat specializovaný hardware. Většina mikroprocesorů je vybavena jednotkou ICR – Input Capture (zachycení vstupu). Tato jednotka funguje tak, že monitoruje vstupní pin a v okamžiku, kdy na něm zachytí hranu požadovaného směru, překopíruje obsah volně běžícího čítače do záchytného registru a vyvolá přerušení. Bohužel zatímco generátorů PWM obsahuje každý mikroprocesor několik, zachytávací jednotky mají pouze některé a ještě k tomu v malém množství. Je tedy nutné použít jiné řešení. Jednotky zachycení hrany musí program emulovat. Má-li mikrokontroler dostatečný počet

přerušovacích vstupů, signál se připojí na tyto vstupy a zachytávání bude součástí obsluhy přerušení. V opačném případě musí mikroprocesor neustále kontrolovat stav vstupních pinů, což naprosto vylučuje jakoukoliv jinou činnost. Při změně vstupní hodnoty musí program nejprve určit, na kterých pinech došlo ke změně – jako vhodná operace se jeví výlučný součin, který nastaví bity, jejichž hodnota se liší, a ostatní vynuluje – a uložit si hodnotu volně čítače v okamžiku změny. Pokud detekovaná změna byla vzestupnou hranou, pak se pouze hodnota čítače uloží. Pokud se jednalo o sestupnou hranu, byl detekován konec impulsu a rozdílem současné hodnoty a předchozí je délka impulsu. V rámci programu je také vhodné alespoň přibližně odhadovat dobu od poslední hrany. Pokud je doba mezi dvěma vstupními signály vyšší než 20 ms, pak pravděpodobně přijímač ztratil konstakt s vysílačem a je nutné na tuto situaci patřičně reagovat.

5.3 ATmega128A

Atmega128A je novější modifikace Atmega128. Hlavním rozšířením je snížení dolní meze napájecího napětí až na 2,7 V bez nutnosti snižovat frekvenci hodinového signálu jako je to u verze označené písmenem L. Mikrokotrolér je vybaven celkem čtyřmi čítači/časovači. Dva z nich jsou osmibitové, druhé dva šestnáctibitové. Pro generování výstupního PWM signálu budou použity šestnáctibitové čítače. Každý z nich je schopen generovat až tři PWM signály. Budou tedy k dispozici dva volné kanály pro ovládání dodatečného systému, například stabilizované plošiny. Jeden z osmibitových čítačů bude použit pro generování zahajovacích impulsů pro čtení dat ze senzorů, poslední pak může zůstat nevyužit nebo jej lze použít například pro blikání s LED osvětlením modelu.

Základem každého čítače je volně běžící čítač, který umožňuje čítat buďto externí signál nebo hodinový signál odvozený děličkami od hodinového signálu mikrokontroléru. Dále je zde jeden registr u osmibitového a tři registry u šestnáctibitového čítače s názvem OCRxy (*Output Compare Register*), kde *x* značí číslo čítače (pro 16 bitové čítače jsou to čísla 1 a 3) a *y* označuje kanálu (čítače 1 a 3 mají kanály A,B,C). Každý z těchto registrů obsahuje také komparátor, který porovnává hodnotu registru volně běžícího čítače s daným registrem a při schodě umožňuje generovat přerušení nebo změnu hodnoty příslušného vstupně-výstupního pinu. Oba šestnáctibitové čítače obsahují také registry ICRx (*Input Capture Register*), které slouží pro zachycení stavu volně běžícího čítače v okamžiku změny vstupního pinu.

Základním režimem je normální režim. Čítač počítá od nuly do maximální hodnoty 0xFFFF, pak dojde k přetečení a cyklus se opakuje. Tento režim je ideální pro zachytávání externí události do registru ICR. Pouze je nutné ošetřit, aby doba mezi dvěma po sobě jdoucími událostmi nepřesahovala dobu přetečení časovače, pak by mohlo dojít ke ztrátě přesnosti. Buďto je možné

použít děličku vstupního signálu nebo si uchovávat počet přetečení časovače od minulé události. Pro generování výstupního PWM signálu je tento režim nevhodný, protože by všechny změny musely být generovány pomocí přerušení při komparaci, které by zbytečně zdržovalo výpočetní smyčku. Také při změně dvou signálů v jeden okamžik by nutně muselo dojít ke zpoždění jednoho z nich, protože procesor nedokáže obsloužit dvě přerušení současně.

Dalším režimem čítače je tzv. CTC (*Clear on Compare*) režim, ve kterém je obsah čítače porovnáván s obsahem registru OCRxA nebo ICRx a v okamžiku shody je vynulován. Zda-li bude použit OCRxA nebo ICRx, určuje zvolený režim. Všechny ostatní vlastnosti jsou shodné s předchozím režimem. Tento režim je vhodný pro generování události po určité konstantní době. Pomocí předděličky a registru OCRx se nastaví interval a od signálu komparace se generuje přerušení, které zahájí požadovanou akci – v konkrétním případě například komunikaci po sběrnici *TWI*. Zároveň při shodě dojde k vynulování časovače a čítání začne od začátku.

Pro generování PWM signálu pro řízení serva se jako nejvhodnější jeví režim Fast PWM. Jako jediný z PWM režimů používá pouze jeden směr čítání volně běžícího čítače – konkrétně vzestupný směr. Vrchol čítače záleží na zvoleném režimu, mohou jím být buďto předdefinované hodnoty 0x00FF, 0x01FF, 0x03FF, obsah registru OCRxA nebo obsah registru ICRx. Pro generování signálu s konstantní periodou je nutné použít režim, který bude porovnávat hodnotu rovnající se poměru frekvence procesoru a požadované periody. Lze tedy použít režimy s registry OCRxA nebo ICRx. Při použití registru OCRxA by ale došlo ke ztrátě jednoho PWM kanálu, proto jsem použil režim s ICRx. Při režimu generování PWM signálu navíc hardware čítače provádí tzv. dvojité bufferování registrů OCRxy. To znamená, že při zápisu nové hodnoty do registru nedojde k jejímu přepsání okamžitě, ale až při přetečení čítače. Tím je ošetřena situace, kdy by zapsaná hodnota byla nižší než aktuální obsah volně běžícího čítače a nemuselo by v rámci celé jedné periody dojít ke shodě registru a vygenerování změny. Čítače umožňují generování signálu na výstupní piny v závislosti na nastavení budičů pinu.

1. Piny nejsou čítačem ovládány, jejich hodnota závisí na nastavení registrů PORTx
2. Piny jsou negovány při každé detekované shodě
3. Piny jsou vynulovány při shodě a při přetečení čítače nastaveny
4. Piny jsou nastaveny při shodě a při přetečení čítače vynulovány

Pro generování vhodného průběhu by byl ideální režim číslo 3, nicméně kvůli negaci signálu na výstupu řídící desky (viz. dále v textu) je použit režim 4.

Další dva režimy časovače jsou režimy s fázovou (a frekvenční) korekcí výstupního signálu. Tyto režimy používají čítač k čítání oběma směry. Čítač nejprve čítá k vrcholu a pak zpět ke dnu. V rámci jednoho cyklu tedy dojde k dvěma událostem porovnání. V závislosti na nastavení je výstupní pin nastaven při čítání směrem nahoru a vynulován při čítání směrem dolů, případně obráceně.

Mikrokontroléry AVR jsou vybaveny sběrnicí TWI. Tato sběrnice je kompatibilní se sběrnicí I2C a obsahuje i částečná rozšíření. Hlavním důvodem, proč mikrokontrolér neobsahuje I2C řadič, jsou licenční podmínky sběrnice I2C. Řadič sběrnice TWI funguje jako stavový automat. Pomocí registru TWCR se ovládá akce, kterou má řadič provést, a vynulováním bitu TWINT se požadovaná akce spustí (bit se nuluje zapsáním log. 1). Řadič nastavením tohoto bitu do log. 1 signalizuje dokončení operace a v horních 5 bitech registru TWSR signalizuje výsledek operace. Registry TWBR slouží jako dělička hodinového signálu pro dosažení požadované přenosové rychlosti, registr TWDR obsahuje odesílanou či přijatou hodnotu. Pokud je TWI jednotka použita jako slave zařízení, TWAR obsahuje adresu tohoto zařízení.

ATMega128A má dva nezávislé sériové kanály. Každý z nich umožňuje odesílat a přijímat data libovolnou rychlostí v asynchronním i synchronním režimu. V synchronním režimu může obvod generovat i přijímat hodinový signál. Požadovaná přenosová rychlost se určuje zápisem vhodné konstanty do registru UDR, který slouží pro děličku hodinového signálu. Následně se zápisem do registru UDR spustí odesílání dat. Přijatá data se čtou také ze stejného registru. Tento registr je dvojitě bufferovaný pro každý směr. Přijatá data či prázdný odesílací buffer indikují bity RXC, TXC, případně UDRE v registru UCSRAx.

Mikrokontrolér umožňuje vyvolat přerušení od jednoho z osmi vstupů. U každého z nich lze nastavit, má-li se přerušení generovat při náběžné, sestupné nebo libovolné hraně, případně má-li se přerušení generovat tak dlouho, dokud je daný vstup držen v nízké logické úrovni. Každý ze vstupů má vlastní vektor přerušení. Určitou nevýhodou je fakt, že skoro všechny vstupní přerušovací vstupy sdílí pin s výstupem ze speciálních funkčních jednotek (čítač/časovač 3, řadič TWI a USART), takže jejich použití není možné.

5.4 ATmega48A

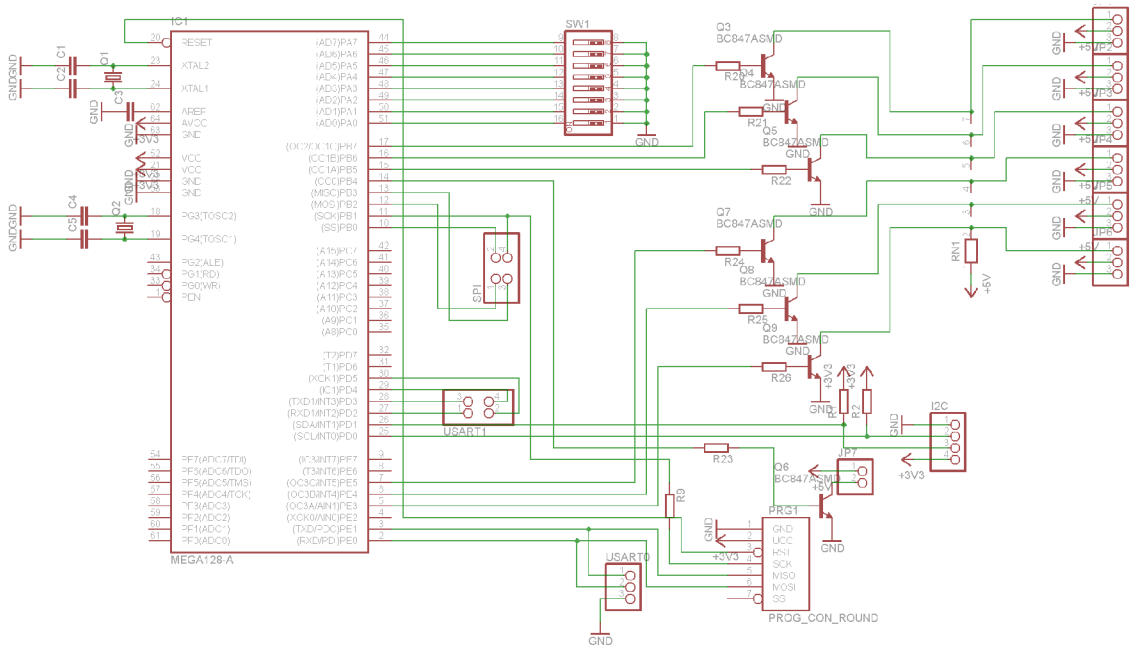
ATmega48A je osmibitový mikrokontrolér rodiny Atmel AVR. Jádro má tedy stejné jako hlavní řídící mikrokontrolér. Hlavním rozdílem mezi těmito mikrokontroléry je dodatečný hardware, kterým jsou vybaveny, a velikost pamětí RAM, FLASH a EEPROM. Stejné rozložení vývodů i stejné periferie mají také mikrokontroléry ATmega88 a Atmega168, pouze větší paměť programu. Zatímco ATmega48 je vybavena pamětí o velikosti 4 KB, ATmega88 má paměť programu o velikosti 8 KB a ATmega168 16 KB. Je tedy možné použít libovolný z těchto mikrokontrolérů, avšak v novější verzi označené písmenem „A“, která může pracovat na frekvencí 16 MHz i při nižším napětí než 5 V.

ATmega48A je navíc vybavena generátorem přerušení při změně libovolného pinu, tzv. Pin Change Interrupt. Pro každý z portů má mikrokontrolér jeden řadič přerušení a je možné každému portu přiřadit vlastní obslužnou rutinu. Řadič je vybaven dvěma řídícími registry a jedním stavovým

registrem. První řídící registr má každý generátor vlastní a slouží pro maskování vstupů, které mohou vyvolat přerušení. Nastavením log. 1 určitému bitu v registru PCMSK se povoluje přerušení při změně vstupní logické úrovně. Přerušení se vygeneruje i v okamžiku, kdy změna bude způsobena zapsáním nové hodnoty na výstupní pin softwarově. Stavový registr PCIFR indikuje, který z generátorů detekoval změnu na povolených vstupních pinech. Pokud je pro daný generátor nastavena log. 1 i v registru PCICR a zároveň je nastaven bit globálně povolující přerušení, bude vygenerováno přerušení. Přesné namapování vstupních pinů mikrokontroléru do jednotlivých generátorů je uveden v datasheetu ke konkrétnímu typu mikrokontroléru.

5.5 Schéma zapojení

Řídící systém je rozdělen do dvou částí. Na první z nich je hlavní řídící mikrokontrolér, který komunikuje se senzory. Druhá část obsahuje koprocesor, který se stará převod signálu z RC přijímače na digitální hodnoty.



Obrázek 5.1: Výřez schématu desky s hlavním mikrokontrolérem.

Schéma zapojení řídící desky je zobrazeno na obrázku 5.1. Drobným problémem při návrhu se stal požadavek na dvě různé napěťové úrovně. Zatímco senzory mohou pracovat maximálně s napětím 3,6 V, modelářské vybavení vyžaduje napětí ve výši 5 V. Z tohoto důvodu jsem se rozhodl použít pro napájení mikrokontroléru stejné napětí jako je na senzorech a napěťové úrovně pro řízení motorů a serva převádět. Pro převod na nižší hladinu stačí použít odporový dělič s vhodným poměrem, pro převod na vyšší hladinu pak buďto specializovaný převodník úrovní nebo budič s otevřeným kolektorem (například ULN2003A) a na výstupu pull-up rezistory. Třetí a mnou použité

řešení vychází z principu budiče s otevřeným kolektorem, ale namísto specializovaného integrovaného obvodu jsem použil obyčejné tranzistory NPN s omezovacím rezistorem v bázi a pull-up rezistorem na výstup. Toto řešení je nejjednodušší a nejlevnější, ale neguje výstupní signál. Právě proto je u časovačů použit invertující režim, který výstupní signál také neguje, takže ve výsledku je na výstupu korektní signál.

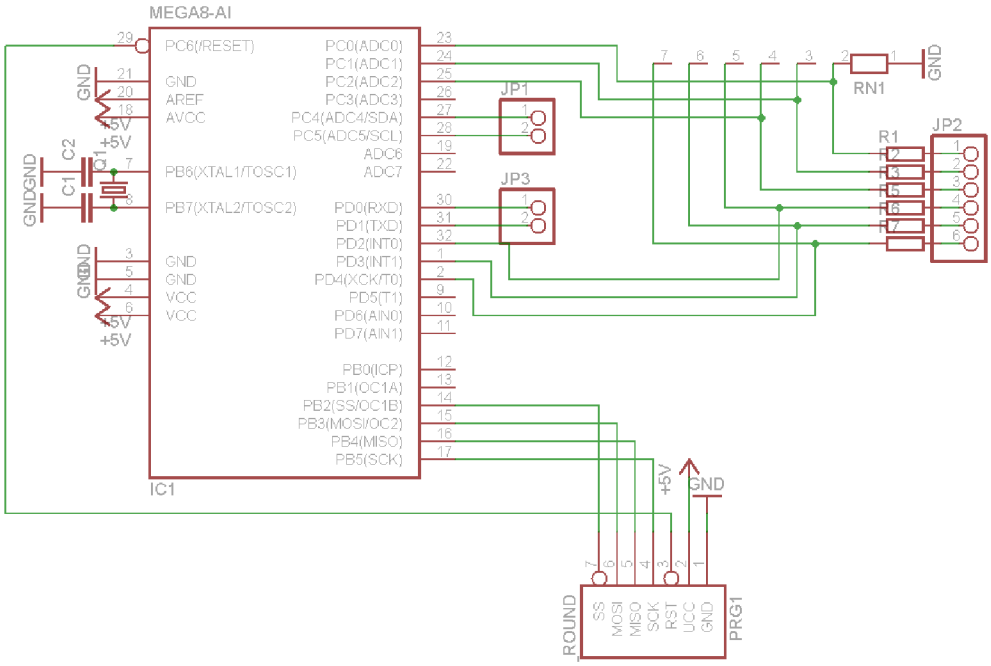
Pro napájení jsou použita dvě napětí: 5 V pro součástky spolupracující s modelářskými komponentami a 3,0 V (může být i 3,3 V) pro hlavní elektroniku. Napětí 5 V lze použít buďto přímo z jednoho z regulátorů, které jsou vybaveny BEC stabilizátorem (*Battery Eliminator Circuit*) nebo přímo z hlavního napájecího napětí baterie. Stejně tak lze napětí pro senzory získávat stabilizací a snížením napětí 5 V nebo použít přímo napětí baterie. Nakonec jsem použil napětí baterie a stabilizoval jej pomocí stabilizátoru 7805 na napětí 5 V a pomocí LM317 jsem stabilizoval napětí pro senzory. Toto řešení má podle mého názoru tu výhodu, že při poruše některého z regulátorů nedojde ke spálení řídící elektroniky přepětím. Na druhou stranu model při výpadku kteréhokoliv z řídících prvků havaruje v každém případě.

Pro připojení programátoru slouží u většiny mikrokontrolérů řady AVR sériové rozhraní SPI. Zvolený mikrokontrolér je výjimkou. Pro hodinový signál sice využívá pin z rozhraní SPI, ale datové vodiče MOSI a MISO jsou nahrazeny piny PDI a PDO, které sdílí výstupní piny s Tx a Rx sériového asynchronního kanálu 0. Během vývoje jsou tedy obě rozhraní SPI i USART0 nepoužitelné, protože data přivedená na tyto piny během programování mikrokontroléru většinou způsobí zaseknutí připojené periferie. Jediným řešením je až kompletní restart celého systému odpojením napájecího napětí.

Pro měření napětí baterií, které napájí regulátory motorů, je možné použít vestavěný analogově-digitální převodník. Zvolený mikrokontrolér je vybaven osmi vstupními kanály, které jsou vyvedeny na port F a mezi kterými je možné softwarově přepínat. Na nejnižší tři piny je přes odporový dělič přivedeno napájecí napětí každého z regulátorů. I když v testovacím zapojení jsou připojeny všechny motory na stejný zdroj, je možné za účelem zvýšení letového času připojit každý z motorů na vlastní akumulátor. Dalších pět dostupných kanálů je vyvedeno na kolíkovou lištu přes neosazené odporové děliče pro budoucí využití. Je možné na ně připojit například záložní analogový gyroskop pro případy chyby komunikace po sběrnici.

Jelikož zvolený mikrokontrolér je vybaven rozhraním pro připojení externí paměti dat, nejsou na porty PA a PC vyvedeny žádné alternativní funkce pinů. Tyto piny je možné využít libovolným způsobem. Pro zjednodušení debugování a testování je na port PA připojena osmice DIP přepínačů pro nastavování parametrů – konkrétně výběr použitého stabilizačního algoritmu – a na port C je připojeno osm LED pro signalizaci nastavených parametrů nebo chybových stavů.

Schéma zapojení desky s koprocesorem je na obrázku 5.2. Jádrem je koprocesor ATmega48A nebo některý z alternativních klonů ATmega88A či ATmega168A. Stejně jako hlavní procesor i koprocesor pracuje s hodinovým signálem 16 MHz. Napájecí napětí je 3 V a je přivedeno pomocí konektoru, kterým jsou přivedeny signály sběrnice I2C. Protože přijímač RC signálu pracuje s napětím 5 V, je nutné toto napětí snížit na hodnotu 3 V. Kvůli tomu je každý vstupní pin vybaven odporovým děličem, který zajišťuje správnou úroveň. Další výhodou je, že nepřipojené piny jsou tímto děličem připojeny na 0 V a nehrozí nečekané chování.



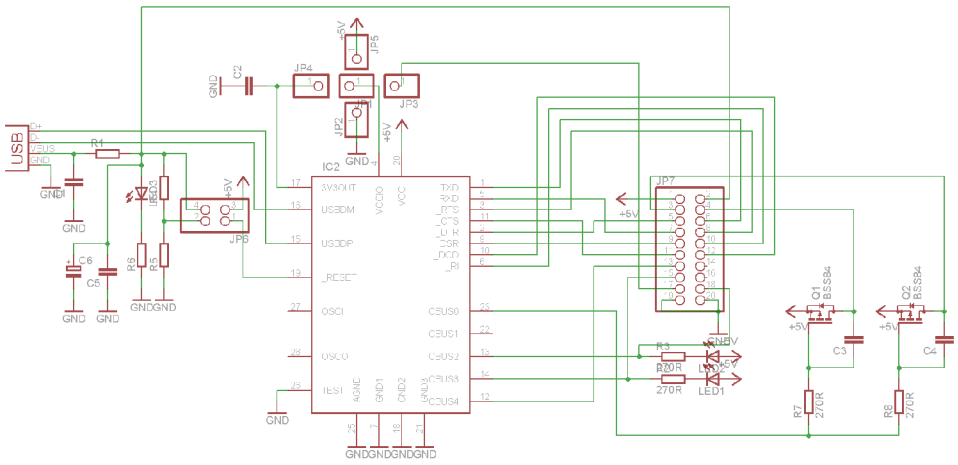
Obrázek 5.2: Výřez schématu desky s koprocesorem.

K mikrokontroléru je také připojen ultrazvukový dálkoměr SFR02, který pracuje s napětím 5 V. Dle specifikace je tento senzor založen na PIC16F687. Minimální napětí, které je tímto obvodem spolehlivě detekováno jako log. 1, je podle datsheetu 3.0 V. Vzhledem k toleranci odporů, které určují výstupní napětí stabilizátoru LM317 může dojít ke stavu, kdy by napětí na výstupu při vyšší zátěži mohlo klesnout pod hladinu 3 V a SRF02 by nemusel komunikovat korektně. Proto jsem pomocí dvou NPN tranzistorů sestavil jednoduchý převodník napětí. V opačném směru je napětí sníženo pomocí dvou tranzistorů namísto napěťového děliče z jednoduchého důvodu. Senzor SRF02 má na svém výstupu definovanou napěťovou úroveň pouze v okamžik, kdy vysílá výstupní hodnotu. V okamžiku, kdy vysílání neprobíhá, je na vstupu nedefinovaná hodnota, což by ve spojení s napěťovým děličem způsobilo, že by na vstupu RX byla log. 0. Kvůli této situaci se bude jednotka USART neustále snažit přijmout nový byte a s velkou pravděpodobností by nedošlo ke správnému zasynchronizování na start-bit.

Koprocesor také generuje signál pro řízení až šesti serv. Zapojení budičů je realizováno stejným způsobem jako na hlavní řídící desce. Stejný port, který ovládá tyto výstupy, také slouží k programování mikrokontroléru. Z tohoto důvodu je vhodné, aby po dobu připojení kabelu od programátoru byly všechny výstupy odpojeny. Frekvence signálu, kterým je mikrokontrolér programován, je v řádech kilohertzů, což je mnohonásobně vyšší než signál pro řízení serv. Tuto skutečnost by řídící elektronika serva vyhodnotila jako požadavek na minimální výchylku, tj. maximální natočení směrem doleva. Protože ovšem elektronika nehlídá meze vstupního signálu, snažil by se motorek serva pohnout až za přípustné mechanické meze a nejspíše by došlo k poškození plastových ozubených koleček.

5.5.1 Převodník USB – USART

Pro komunikaci systému s počítačem za účelem nastavování konstant nebo zaznamenávání letových údajů je možné použít dvě rozhraní. Buďto moderní rozhraní USB, jehož řadičem není mikrokontrolér vybaven, ale je možné jej emulovat například projektem V-USB, nebo starší USART. Použití USB rozhraní by s sebou přinášelo další úskalí. Každé zařízení vyžaduje VID a PID číslo, za jehož vydání se platí. Navíc by nadřazené zařízení vždy muselo být vybaveno USB Host řadičem. Na druhou stranu sériovou linkou již není vybaven žádný nový počítač. Je tedy nutné zvolit kompromisní řešení. Na trhu existují obvody různých firem, které emulují sériovou linku v počítači přes rozhraní USB. Takto emulované porty mají sice určitá omezení, která pramení v principu komunikace po USB, ale pro účely běžné komunikace jsou zanedbatelná.



Obrázek 5.3: Schéma programátoru.

Schéma použitého převodníku je na obrázku 5.3. Převodník je založen na obvodu firmy FTDI Chip FT232RL. Zapojení vychází z doporučení výrobce [25] a již existujících převodníků [24].

Tento obvod umožňuje emulaci kompletní sériové linky včetně všech řídících signálů. Dále je vyvedeno pět univerzálních výstupů, kterým je možné pomocí konfigurační utility nastavit speciální funkce. V počítači se převodník identifikuje jako běžný sériový port s nastavitelným číslem portu, je tedy možné s ním pracovat jako s normálním sériovým portem. Obvod pro svojí činnost vyžaduje minimální počet externích součástek. Dle minimálního zapojení stačí pouze kondenzátor na výstupu děliče napětí. Doporučené zapojení vyžaduje tlumivku na přívodu napájecího napětí a dodatečné blokovací kondenzátory. Jelikož specifikace USB vyžaduje, aby během konfigurace odběr připojeného zařízení nepřesahoval 100 mA, jsou ostatní obvody odděleny mosfet tranzistory. Obvod umožňuje na libovolný z výstupů vyvést signál, který signalizuje ukončení konfigurace, a spínat tyto tranzistory. Dále jsou zde vyvedeny dvě signalizační LED, jedna signalizuje příjem dat po USART, druhá vysílání. Napěťové úrovně komunikačních signálů je možné nastavit v rozmezí 1,8 až 5,25 V. Nastavení konkrétní úrovně se provádí připojením pinu VCCIO na napětí odpovídající logické úrovni 1. Na navrženém převodníku se napětí VCCIO volí pomocí zkratovací propojky na jednu z variant: 5 V, 3.3 V nebo z připojeného obvodu. Při návrhu jsem se zaměřil také na univerzální použitelnost převodníku, takže jsou všechny dostupné piny vyvedeny na výstupní konektor.

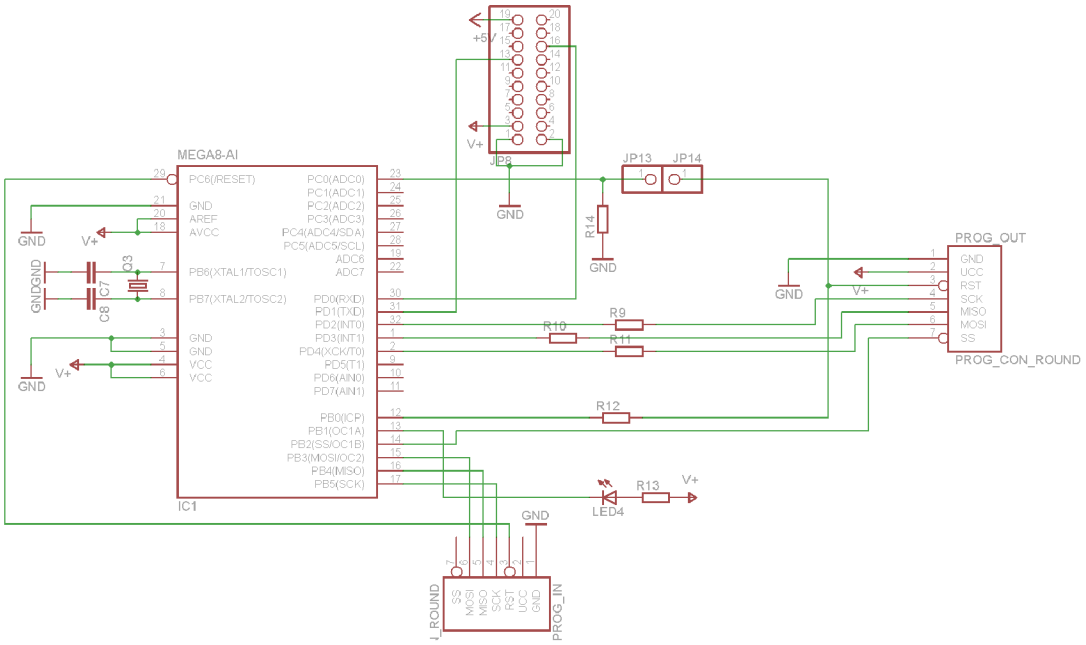
5.5.2 Programátor

Programování mikrokontrolérů rodiny AVR se provádí za sériovým rozhraním ISP – in system programming [18][19]. Toto rozhraní je velice podobné rozhraní SPI. Lze jej velice jednoduše emulovat pomocí paralelního nebo sériového portu přímým nastavováním jednotlivých pinů. Bohužel těmito rozhraními se již počítače neosazují a při jejich emulaci programátory nepracují správně, protože nejsou dodrženy předepsané časové intervaly. Výjimkou jsou obvody FT232RL a podobné, které umožňují režim Bit-Bang, ve kterém je možné přesně ovládat jednotlivé piny, avšak za cenu mnohonásobného snížení přenosové rychlosti. Navíc tyto programátory nelze ovládat z integrovaného vývojového prostředí AVR Studio, které je volně dostupné ze stránek výrobce mikrokontrolérů.

Hodně rozšířené řešení uvádí [26][27]. Jedná se emulátor vývojového kitu STK500v2. Tento vývojový kit se připojoval k počítači pomocí běžné sériové linky a umožňoval programování mikrokontrolérů a debugování vyvíjených aplikací pomocí JTAG. Uvedený programátor emuluje pouze nejnutnější část příkazů určených k programování mikrokontrolérů.

Programátor řídí mikrokontrolér Atmega8A. Ta pomocí sériové linky komunikuje s vývojovým studiem a emuluje rozhraní ISP pro programování cílového mikrokontroléru. Toto řešení ovšem vytváří problém zvaný „vejce a slepice“ – pro vytvoření programátoru potřebujeme naprogramovat mikrokontrolér. Řešením je použití Bit-Bang režimu v kombinaci s softwarem AVR Dude. Takto lze jednorázově naprogramovat mikrokontrolér programátoru pouze za pomoci výše popsaného převodníku.

Schéma zapojení programátoru je na obrázku 5.4. Oproti literatuře jsou zde drobné změny. První je odstranění obvodu FT232RL. Programátor bude připojován k výše uvedenému převodníku, a proto je tento obvod zbytečný. Druhou úpravou je velikost napěťového děliče. Software programátoru detekuje, zda-li je připojen programovaný mikrokontrolér měřením napětí na vývodu reset za pomoci ADC. Pokud je naměřené napětí mimo povolený rozsah, programátor odmítá mikrokontrolér naprogramovat. Protože je programátor navrhován na napětí 5 V, jeho připojení na 3 V logiku by způsobilo nedostatečné napětí na ADC a programovaný mikrokontrolér by nebyl detekován. Stejný problém by vznikl i v okamžiku, kdy by programátor nastavený na 3 V hodnoty byl připojen k napětí 5 V. Proto je jeden z odporů vyveden mimo desku aby bylo možné jeho hodnotu měnit podle aktuální potřeby.



Obrázek 5.4: Schéma zapojení programátoru.

6 Řídící software

Řídící algoritmus tricopteru vykonává několik funkcí. Základní funkcí je stabilizace letových parametrů tricopteru. K tomu je nutné v pravidelných intervalech načítat data ze senzorů i řídící signál z vysílačky a upravovat výstupní signál. Další činností řídícího programu je nastavování parametrů stabilizačního algoritmu.

6.1 Stabilizační algoritmus

Cílem stabilizačního algoritmu je minimalizovat odchylky naměřených nebo vypočítaných hodnot od hodnot přijatých po řídícím kanále. Způsob realizace stabilizace záleží na principu ovládání tricopteru. Navržený algoritmus interpretuje přijaté řídící signály stejným způsobem jako vrtulník. Pozice plynové páky odpovídá pozici kolektivního řízení, ovládání křidélek a výškového kormidla slouží pro ovládání cyklického řízení a směrové kormidlo nahrazuje pedály. Jinak řečeno, plynová přípusť určuje průměrný tah všech tří motorů. Zbylé tři ovládací prvky určují velikost úhlových rychlostí okolo příslušných os. Pro tuto základní stabilizaci stačí pouze údaje z gyroskopu.

Vstupní hodnoty, které jsou naměřeny gyroskopem jsou před zpracováním filtrovány. V systému jsou implementovány dva filtry z nichž je možné zvolit právě jeden. Volba se provádí pomocí DIP přepínačů. První filtr využívá prostý aritmetický průměr několika po sobě jdoucích vzorků. Množství vzorků, které filtr používá, je nastavitelné pomocí sériové linky. Od tohoto počtu se také odvíjí frekvence, s jakou stabilizačním algoritmus aktualizuje výstupní hodnoty. Druhý filtr vychází z projektu FreeScale Race Challange [37]. Tento filtr je převzat z volně dostupné šablon zdrojových kódů.

Samotný regulační algoritmus je rozdělen do dvou částí. Zatímco stabilizace okolo podélné a příčné osy je ovládána změnou tahu motorů, rotace okolo svislé osy je kompenzována náklonem zadního rotoru, který ovládá servo. Servo má jinou dobu i rychlost odezvy než regulátory motorů, proto musí být každý z regulátoru navržen jinak. Stabilizaci letové polohy zajišťují dva PID regulátory, každý pro jednu osu. Regulátor čte řídící signál a údaje naměřené gyroskopem. Každá z těchto hodnot je vynásobena přednastavenou konstantou. Výsledné hodnoty jsou porovnány a je vypočten jejich rozdíl. Tento rozdíl tvoří proporcionální výstup regulátoru. Zároveň je přičten k integrační složce a rozdíl posledních dvou odchylek tvoří diferenciální složku. Výstupní hodnota regulátoru je vytvořena součtem všech tří hodnot vynásobených příslušnými konstantami. Stabilizace svislé rotace se provádí pouze PI regulátorem. Výstupní hodnota je vypočtena pouze na základě integrační složky, která je pravidelně zvyšována o vypočtený rozdíl hodnot.

Rovnice každého regulátoru vychází z [33][34]. Regulátor je implementován přesně podle těchto rovnic. Drobnou výjimkou je implementace operace dělení časovou konstantou. Jelikož procesor nemá hw podporu této operace a její emulace je časově náročná, je operace dělení nahrazena násobením převrácenou hodnotou. Dělení tak probíhá pouze jednou a to v okamžiku, kdy je tato časová konstanta poprvé načtena v průběhu inicializace. Inicializace není časově kritická a delší doba výpočtu nepředstavuje žádnou zásadní potíž.

Výše uvedený algoritmus je snadné rozšířit i o další funkcionalitu. Za pomoci výškoměru a údajů z akcelerometru je možné doplnit stabilizaci výšky. Případně je možné upravit ovládání tak, že by cyklické řízení neurčovalo úhlovou rychlost ale náklon v jednotlivých osách. Toho se dá dosáhnout změnou výpočtu proporcionální složky. Namísto použití hodnot z gyroskopu bude použit jejich součet. Protože hodnoty gyroskopu jsou načítány v pravidelných okamžicích, jejich numerickou integrací lze vypočítat úhel náklonu. Ten pak bude použit jako vstupní hodnota pro výpočet rozdílu, který je proporcionální složkou.

Další možné rozšíření stabilizačního algoritmu již vyžaduje přesnější zpracování vstupních dat ze senzorů.

6.2 Komunikace se senzory

Mikrokontrolér je vybaven řadičem kompatibilním se sběrnicí I2C, takže veškeré řízení komunikace se omezuje pouze na čtení a zápis do řídících registrů jednotky a čekání, až jednotka dokončí operaci. Pro čekání na dokončení lze použít buďto pooling nebo jednotka umožňuje generovat přerušení po dokončení operace. Dokončení jedné operace přenosu dat při frekvenci hodinového signálu 400 Khz trvá přibližně 500 hodinových taktů, během kterých by procesor pouze čekal. Během jednoho čtení je nutné po sběrnici přenést přibližně 70 bytů, takže by procesor čekáním zbytečně ztratil cca 35 000 hodinových taktů. Z tohoto důvodu se použití poolingu jeví jako velmi nevhodné.

Alternativou k poolingu je použití přerušení, během kterého by byly nastaveny hodnoty řídících registrů a spuštěna operace. Během jejího provádění by mikrokotrolér mohl provádět výpočty potřebné ke stabilizaci tricopteru. Podmínkou pro použití přerušení je vytvoření stavového automatu, který bude řídit komunikaci.

6.2.1 Návrh stavového automatu

Stavový automat pro řízení komunikace po sběrnici je navržen pouze pro čtení dat ze senzorů, pro zápis konfigurace do jednotlivých registrů připojených zařízení na sběrnici je nutné použít funkce, které využívají pooling. Toto omezení jsem zvolil z důvodu vysoké složitosti celkového automatu,

který by zvládal všechny tyto možnosti. Navíc zápis konfigurace probíhá pouze jednou a to během inicializace senzorů, následně již probíhá periodické čtení.

Ve výchozím stavu čeká automat na externí podnět, který by spustil komunikaci vygenerováním počátečního Start stavu na sběrnici zapsáním 1 do bitu TWSTA v registru TWCR zároveň s nastaveným bitem TWIE, který povoluje přerušení, a bitem TWINT, což odstartuje operaci. Po dokončení operace jednotka vygeneruje přerušení. Obsluha přerušení následně rozhodne o následující operaci v závislosti na stavu, ve kterém se automat nachází.

Stavový automat prochází jednotlivé senzory v předem zadaném pořadí a čte datové registry. Pro každý senzor je nutné provést následující sekvenci: Adresovat zařízení pro zápis a zapsat adresu prvního datového registru (tj. s nejnižší adresou). Následně vyslat restart stav a adresovat zařízení pro čtení přečíst příslušný počet bytů. Každý senzor automaticky provádí post-inkrement adresy, takže není potřeba znovu odesílat adresu. Po dokončení čtení automat vybere další zařízení v pořadí a celý proces čtení dat se opakuje. Pokud automat přečte údaje ze všech senzorů, přejde do stavu IDDLE a čeká na další odstartování přenosu.

6.2.2 Implementace stavového automatu

Pro určení aktuálního stavu slouží proměnná FSM_STATUS. Tato proměnná obsahuje ve třech nejvyšších bitech číslo právě obsluhovaného zařízení, zbylých 5 bitů slouží pro identifikaci operace. Čísla zařízení a čísla operací zobrazuje tabulka v [18].

Další proměnná FSM_COUNTER slouží jako počítadlo načítaných bytů. Po odeslání adresy zařízení pro čtení se tato proměnná v závislosti na typu zařízení nastaví na počet bytů a postupně je snižována. Před přijetím posledního bytu je nutné vynulovat bit TWEA, který povoluje potvrzování přijatého bytu vynulováním bitu ACK. Jednotka TWI neumožňuje ukončit komunikaci vysláním STOP signálu, pokud byl předcházející přijatý bit potvrzen.

Protože u některých senzorů je možné, že dojde ke změně hodnot datových registrů přímo během komunikace a tedy přijatý dolní a horní byte by nebyl součástí jedné korektní hodnoty, je nutné provádět u těchto senzorů více čtení a následně rozhodnout, která hodnota je korektní. Pro určování, kolik průběhu čtení je potřeba provést, slouží proměnná FSM_REPEAT. Tato proměnná je nastavená vždy po změně čísla senzoru na následující. Pokud je tato proměnná různá od 1 po přijetí posledního bytu dat, vygeneruje stavový automat nový start signál na sběrnici a bude pokračovat znovu s načítáním dat od daného senzoru. Po dokončení všech iterací pro určitý senzor se z pomocného bufferu vybere, které hodnoty jsou korektní a tyto budou zapsány na výstup. Pokud je první vzorek stejný jako druhý, použije se první, v opačném případě se použije třetí. Vzhledem

k poměru obnovovací rychlosti senzorů a přenosové rychlosti na sběrnici není možné, aby vzniklo než jak jedna změna během čtení a tedy není nutné provádět žádná další porovnání.

Pro vyfiltrování drobného šumu ve vstupních datech se provádí více načítání během jedné iterace a data jsou následně zprůměrována. Jelikož je operace dělení softwarově emulována, což by způsobilo značné zdržení ve výpočtu, je počet iterací volen jako mocnina čísla dvě. Pak je možné místo dělení použít bitový posun, což je operace, pro kterou má mikrokontrolér hardwarovou podporu.

Aby nedocházelo k ovlivňování dat, která byla načtena v předchozí iteraci a která právě zpracovává hlavní smyčka řídícího programu, s daty, která jsou nově načítána, jsou všechny používané proměnné a pole zdvojeny. Namísto proměnné je použito pole, namísto jednorozměrného pole je použito dvourozměrné. První dimenze rozhoduje, zda-li data používá stavový automat pro načítání nebo zda-li obsahují platná data pro řídící smyčku. Aby nebylo nutné data vždy přepisovat mezi jednotlivými dimenzemi, právě upravovaná data označuje proměnné FSM_BUFFER_ACTIVE, která nabývá hodnot 0 nebo 1.

Načítání dat je periodicky spouštěno po 2,5 ms. K přesnému načasování slouží čítač / časovač 2 v režimu časovače. Jelikož čítač je pouze 8 bitový, není možné dosáhnout ani po použití předděličky požadované frekvence přetečení. Při vhodném nastavení předděličky vychází potřebný počet přetečení na zaokrouhleně 2,4. Toto je nutné vyřešit použitím tří přetečení a po každém třetím přetečení upravit hodnotu volně běžícího čítače tak, aby hodnota byla přímo úměrná desetinné části počtu přetečení.

U jednotky TWI občas dochází k uváznutí především kvůli chybě při přenosu potvrzovacího bitu. V takovém případě jednotka signalizuje pomocí stavového registru TWSR chybovou hlášku a je nutné korektně chybu ošetřit. Ve výjimečných situacích se jednotka zasekne a jediné řešení je provést restart této jednotky vynulováním bitu TWEN a jeho opětovným nastavením. Je nutné mít vhodný zdroj, který po uplynutí maximální doby vygeneruje přerušení, jehož obsluha provede zmiňovaný restart jednotky, uloží informaci o chybě do stavového registru a vynutí start přenosu pro další zařízení. K tomu jsem použil komparační jednotku časovače 2, který generuje začátky přenosu. Vždy po zahájení přenosu uloží stavový automat do registru komparátoru aktuální hodnotu volně běžícího čítače zvýšenou o maximální timeout. Pokud je operace dokončena dřív než nastane shoda na komparátoru, stavový automat opět posune hodnotu registru komparátoru. Pokud by bylo vygenerováno přerušení v průběhu zpracovávání dokončené operace stavovým automatem, jeho obsluha neproběhne, protože obsluha dokončení operace TWI stavovým automatem probíhá v přerušení, které je blokující a tedy nedovolí vygenerování žádné další obsluhy přerušení. Při zápisu nové hodnoty do registru komparátoru je zároveň programově vynulován požadavek na přerušení. Ke shodě na komparátoru také může dojít i v okamžiku, kdy žádná operace na sběrnici I2C

neprobíhá. V tomto případě je absolutně zbytečné provádět restart jednotky TWI. Proto je v obsluze přidána podmínka, která zabraňuje provádění restartu v případě, že stavový automat je ve stavu IDLE. Po zahájení přenosu uloží stavový automat do registru komparátoru aktuální hodnotu volně běžícího čítače zvýšenou o maximální timeout. Pokud je operace dokončena dřív než nastane shoda na komparátoru, stavový automat opět posune hodnotu registru komparátoru. Pokud by bylo vygenerováno přerušení v průběhu zpracovávání dokončené operace stavovým automatem, jeho obsluha neproběhne, protože obsluha dokončení operace TWI stavovým automatem probíhá v přerušení, které je blokující a tedy nedovolí vygenerování žádné další obsluhy přerušení. Při zápisu nové hodnoty do registru komparátoru je zároveň programově vynulován požadavek na přerušení. Ke shodě na komparátoru také může dojít i v okamžiku, kdy žádná operace na sběrnici I2C neprobíhá. V tomto případě je absolutně zbytečné provádět restart jednotky TWI. Proto je v obsluze přidána podmínka, která zabraňuje provádění restartu v případě, že stavový automat je ve stavu IDLE.

6.2.3 Reprezentace naměřených hodnot

Při svém letu tricopter neustále mění svůj náklon. Pro korektní zpracování a řízení je nutné tyto hodnoty pravidelně přepočítávat do referenční vztažné soustavy. Jako referenční soustavu je vhodné použít soustavu, která bude vhodná pro operátora a tedy nebude nutný další přepočet. Pro řízení vrtulníků a vznášedel se obecně předpokládá, že směr podélné a příčné osy stroje je schodný se směrem podélné a svislé osy souřadného systému. Jinak řečeno, úhel natočení podél svislé osy je nulový. V případě rotace podél svislé osy je nutné přepočítávat rychlosti a ostatní proměnné z referenční soustavy.

Řídící systém si musí uchovávat aktuální úhly náklonu. Pomocí těchto úhlů je nutné transformovat hodnoty naměřené akcelerometrem a gyroskopem před jejich dalším zpracováním. Po přepočtu je nutné od hodnot odečíst klidové hodnoty a výsledek již lze považovat za korektně naměřené odchylky. Následně je nutné ještě určit změnu úhlů a o tuto odchylku jednotlivé úhly upravit. V případě zjištěné rotace podle svislé osy provést přepočet proměnných z referenční soustavy.

6.3 Koprocesor

Pro dekódování signálu z přijímače RC a komunikaci s ultrazvukovým dálkoměrem jsem použil druhý mikrokontrolér, který pracuje jako koprocesor. Důvodů pro toto řešení je několik. První je překrývající se namapování speciálních funkcí na vývody mikrokontroléru ATmega128A. Pro měření vstupního signálu je nutné měřit dobu mezi náběžnou a sestupnou hranou každého kanálu. Proto je ideální mít alespoň čtyři zdroje přerušení – pokud se bude používat pouze čtyřkanálová vysílačka pro

ovládání základních veličin. Ačkoliv mikrokontrolér má osm nezávislých zdrojů přerušení, po připojení sběrnic I2C a UART a výstupního PWM signálu pro ovládání regulátorů a serva se tento počet sníží na pouhé dva zdroje přerušení. Bylo by tedy nutné použít kombinační logiku, která by převáděla změnu libovolného počtu vstupů na změnu v jednom výstupu, který by byl připojen k mikrokontroléru.

Dalším důvodem je připojení ultrazvukového dálkoměru. Ten sice podporuje sběrnici I2C ale implementace nedodržuje striktně protokol a je tedy nemožné jej připojit na sběrnici spolu s ostatními senzory. Připojení přes sériovou linku zase není vhodné z hlediska debugování a měření. Mikrokontrolér má pouze dva sériové kanály, přičemž jeden slouží pro připojení programátoru a druhý pro debugování nebo připojení zapisovače dat pro ukládání letových údajů a údajů o stabilizaci.

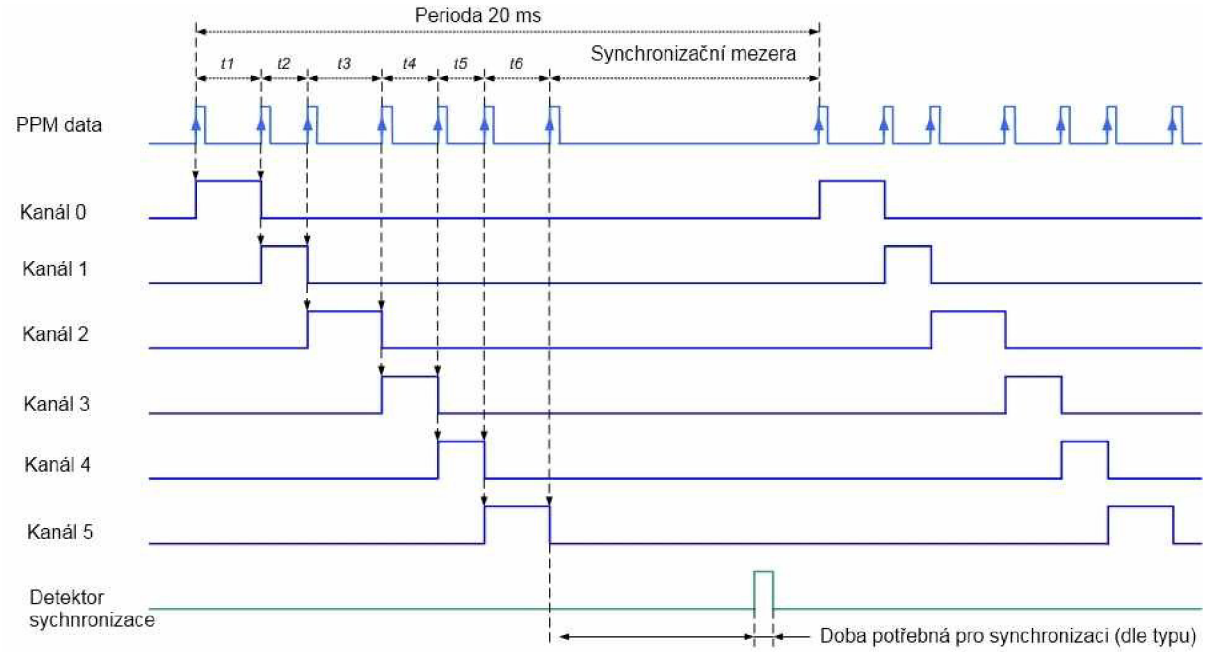
6.3.1 Čtení signálu z RC

Koprocesor dokáže v použitém zapojení číst až šest různých kanálů. Jednoduchým rozšířením plošného spoje by bylo možné tento počet zvýšit až na 10 kanálů. Vstupní signál z přijímače je připojen na port C a D. Hardware mikrokontroléru umožňuje vygenerovat přerušení při změně libovolného pinu. Každý port má svojí obslužnou rutinu, které je předáno řízení, pokud došlo ke změně. Určení, který z pinů přerušení vyvolal, zůstává v režii obslužné rutiny.

Obslužná rutina si uchovává hodnotu vstupních pinů z minulé obsluhy přerušení. Při dalším přerušení provede porovnání hodnot a u pinů, kde došlo ke změně, vypočte novou délku impulsu. Porovnání se provádí pomocí exkluzivního součtu. Bity, které mají ve výsledku log. 1, označují piny, kde došlo ke změně. Rozlišení mezi náběžnou a sestupnou hranou se provádí čtením hodnoty vstupního pinu. Pokud je hodnota pinu rovna log. 1, pak se jednalo o náběžnou hranu, jinak byla hrana sestupná. Při obou hranách se ukládá hodnota volně běžícího čítače, při náběžné hraně do proměnné označující začátek, při sestupné do proměnné, která označuje konec impulsu.

Obslužnou rutinu lze rozdělit na dvě části. První je časově kritická, proto je vykonána okamžitě po vstupu. Jedná se o načtení hodnoty portu, její uložení do proměnné uchovávající minulou hodnotu. Druhu kritickou operací je uložení hodnoty volně běžícího čítače. Zbylé operace, kterými jsou určení bitu, který přerušení vyvolal, a výpočet délky impulsu již nemusí proběhnout ihned. Proto před jejich vykonáním je znovu povoleno přerušení, aby případné ostatní změny byly zachyceny s co největší přesností. Výpočet délky impulsu je prováděn v hlavní kontrolní smyčce mimo obslužnou rutinu přerušení. Výsledek, který reprezentuje délku impulsu v mikrosekundách, je uložen v šestnáctibitovém formátu. Protože ovšem procesor pracuje pouze s osmibitovými registry a také přenos po sběrnici I2C je osmibitový, mohlo by v určitých případech dojít ke zkreslení hodnoty tím, že by byl obsah proměnné přepsán v průběhu přenosu.

Kromě propojení jednotlivých kanálů existuje také možnost použití signálu, kterým jsou data mezi vysílačem a přijímačem přenášena [36]. Tento signál, označovaný jako PPM, přenáší obsah všech kanálů serializovaný do jednoho vodiče v rámcích o délce 20 ms. Časový průběh tohoto signálu ukazuje obrázek 6.1. Vzdálenost mezi jednotlivými náběžnými hranami reprezentuje délku trvání pulsu na jednotlivých kanálech. Tento signál lze použít pouze pro vysílačky, které mají počet kanálů osm nebo nižší. Při vyšším počtu kanálů již není možné zaručit, že by se součet délek vešel do časového limitu pro celý rámec.



Obrázek 6.1: Princip kódování dat v PPM [36].

Pro dekódování signálu PPM je nutné připojit zdroj signálu na pin externího přerušení 0 nebo 1, tj. na piny PIND 2 nebo PIND 3. Vnitřní hardware mikrokontroléru je nakonfigurován tak, aby generoval přerušení pouze od náběžné hrany signálu. Pro zasynchronizování přijímače je třeba vyhledat synchronizační mezeru – tedy dva pulsy, které jsou od sebe vzdáleny o více jak 2.5 ms. Synchronizace může být poněkud problematická u osmikanálové vysílačky, kde při velkých délkách impulsů na jednotlivých kanálech nemusí být synchronizační mezera jednoduše detekovatelná. Jedinou možností, jak vynutit synchronizaci, je přesunout všechny řídící prvky do minimální polohy, čímž dojde ke zkrácení délky celého rámce a mezera by již měla být detekovatelná. Pro vysílačky s více jak osmi kanály není tento způsob vůbec použitelný, protože tyto vysílačky již používají digitální přenos dat.

Po zasynchronizování jsou synchronizační mezery používány pouze pro určení konce rámce u přijímačů, které mají počet kanálů nižší než osm. V obsluze přerušení jsou jednak počítány pulsy pro určení indexu kanálu a také probíhá kontrola délky impulsu. Pokud počítadlo napočítá devátý impuls nebo délka impulsu je větší než limit, dojde k vynulování počítadla. Stejně jako v předchozím

případě jsou ukládány začátky a konce a výpočet probíhá v hlavní programové smyčce se zakázaným přerušením.

Kromě měření hodnot jednotlivých kanálů musí koprocesor také kontrolovat, zda-li je stále přijímán platný signál od vysílače. Pokud je signál ztracen, přistane přijímač generovat výstupní signál. Horší situace je, pokud signál je příliš zašuměný a přijímač dává na výstup chybná data. Obě situace je nutné korektně ošetřit. Ošetření kompletní ztráty signálu spočívá v kontrole doby mezi dvěma změnami na jednom kanále. Volně běžící čítač, který slouží jako časová základna pro měření signálu, přeteče jednou za 32,268 ms. Obslužná rutina přerušení při změně pinu inkrementuje počítadlo změn, které je kontrolováno a nulováno při obsluze přetečení časovače. Pokud mezi dvěma přetečeními nenastaly alespoň 4 změny, pak není vstupní signál korektní. Druhou kontrolou je ověřování velikosti naměřené hodnoty. Pokud naměřená hodnota leží mimo interval, není možné považovat naměřenou hodnotu za korektní a použije se předchozí správná hodnota.

6.3.2 Měření výšky letu

Kromě měření vstupního signálu z vysílačky se koprocesor stará o komunikaci s ultrazvukovým dálkoměrem SFR02. Senzor je připojen pomocí asynchronní sériové linky a komunikuje rychlostí 9 200 Bd. Na rozdíl od ostatních senzorů, SRF02 nezahajuje měření sám, ale je nutné měření zahájit zasláním příkazu. Odesláním příkazu 0x54 zahájí senzor měření a sám zašle naměřený údaj po dokončení měření. Naměřený výsledek je předán zpět jako šestnáctibitová hodnota udávající vzdálenost od podložky v centimetrech. Po přijetí naměřené hodnoty je ihned zahájeno nové měření opětovným odesláním příkazu. Pro případ, že by během přenosu dat do nebo ze senzoru došlo k chybě a data nebyla přijata, je vynuceno nové zahájení měření po vypršení timeoutu. Podle [17] měření trvá maximálně 70 ms, takže jsem zvolil timeout 126 ms – čtyři přetečení časovače pro měření signálu z přijímače.

6.3.3 Generování signálu pro řízení serva

Jelikož je koprocesor vybaven schodným časovačem, jakým je vybaven i řídící mikrokontrolér. Proto se logicky nabízí možnost použít i koprocesor pro generování signálu pro ovládání dodatečných serv, například pro stabilizaci plošiny pro kameru. Jelikož časovač je používán i pro měření délky impulsu a je požadován vyšší počet kanálů než kolik umožňuje samotný HW generátor, je použita metoda, kdy namísto generování výstupu komparátorem je vyvoláno přerušení. Princip generování je podobný způsobu, jakým jsou přenášena data zakódovaná PPM. Přerušení komparátoru B se využívá pro generování rámců o délce 20 ms. Při přerušení se nastaví nejnižší bit, do registru komparátoru B se nastaví hodnota volně běžícího čítače po uplynutí dalších 20 ms a do registru komparátoru A se uloží hodnota čítače v době počátku přerušení zvýšená o délku impulsu na kanálu 0. Po uplynutí této doby

dojde k vygenerování přerušení komparátorem A, který vynuluje výstupní pin aktivního kanálu, nastaví výstupní pin následujícího kanálu a zvýší hodnotu registru komparátoru o délku následujícího kanálu. Tento proces proběhne pro všechny kanály. U posledního kanálu se nemění obsah registru komparátoru, čímž se zajistí vyčkání na přerušení od komparátoru B.

Teoreticky je možné tímto způsobem generovat až 8 kanálů, pokud součet délky jednotlivých kanálů je nižší než 20ms. Protože výstupní piny jsou přiřazeny portu B, je možné použít pouze šest kanálů, jelikož nejvyšší dva piny slouží pro připojení krystalu, který používá vestavěný oscilátor pro generování hodinového signálu mikrokontroléru. Výše uvedený postup trpí jednou nevýhodou. Jelikož systém používá více přerušovacích kanálů, bude docházet k mírnému prodloužení a zkrácení některých úseků, nicméně odchylka by měla být zanedbatelná. Výstupní průběh bude vypadat stejně jako přenos v PPM, tj. kanály 0 až 5 v obrázku 6.1

6.3.4 Komunikace po I2C

Koprocesor se na sběrnici chová jako externí paměť o velikosti 256 Bytů. Do tohoto adresového prostoru jsou namapovány registry jednotlivých zařízení. Adresy registrů jsou zobrazeny v tabulce 6.1. Komunikační protokol je shodný s protokoly, které jsou použity pro komunikaci se senzory. Pro zápis je odeslána adresa zařízení, následně adresa nejnižšího registru, ke kterému bude přistupováno. Následně může být obsah tohoto registru zapsán a adresa se automaticky inkrementuje. Pro čtení je nutné vygenerovat stop + start nebo restart a odeslat I2C adresu zařízení. Po té bude koprocesor odesílat postupně obsah jednotlivých bytů. Stejně jako při zápisu je po každém přečteném bytu hodnota adresy inkrementována. Při zápisu mimo adresový prostor registrů se nikam zapisovaná hodnota neuloží, ale přenos bude potvrzen ACK. Při pokusu o čtení mimo uvedené registry bude vrácena hodnota globálního stavového registru, který zobrazuje stav připojených zařízení.

Adresa (hex)	**Název**	**Popis**
0x00	Dev_ID	Identifikace zařízení (0xAA)
0x10	Slave_Addr_0	První byte pro změnu slave adresy
0x11	RC_Config	Konfigurační registr dekodéru RC signálu
0x12	RC_Status	Stavový registr dekodéru RC signálu
0x13 – 0x1E	RC_Data	Dekódovaná data kanálů RC, little endian
0x30	Range_Config	Konfigurační registr obsluhy dálkoměru
0x31	Range_Status	Stavový registr obsluhy dálkoměru
0x32	Range_Low	Dolní byte naměřené vzdálenosti
0x33	Range_High	Horní byte naměřené vzdálenosti
0x50	RC_Out_Config	Konfigurační registr generátoru PWM
0x51 – 0x5C	RC_Out_Data	Vstupní data pro generátor PWM
0xAA	Slave_Addr_1	Druhý byte pro změnu slave adresy

Tabulka 6.1: Mapování registrů do paměťového prostoru.

Všechna zařízení, která jsou namapována do paměťového prostoru koprocesoru, mají stejnou adresu na sběrnici I2C. Tato adresa je uložena v paměti EEPROM a je možné ji změnit. Změna se provádí tak, že se do registrů SLA_ADDR1 a SLA_ADDR2 zapíše nová adresa. Oba zápisy musí následovat po sobě a nesmí mezi nimi být přistupováno k žádnému jinému registru. Po zapsání nové adresy je doporučeno provést reset mikrokontroléru odpojením napájení. V případě zapomenutí slave adresy je možné propojením pinu 0 portu B se zemí a následným připojením napájecího napětí vymazat nastavenou adresu. Po rozpojení propojky bude zařízení opět komunikovat na defaultní adrese.

Koprocesor je připojen na sběrnici I2C pomocí modulu TWI, který obstarává kompletní fyzickou obsluhu sběrnice. Jednotka monitoruje komunikaci na sběrnici a v okamžiku, kdy rozpozná svojí adresu a má povoleno potvrzování, signalizuje nastavením bitu TWINT žádost o obsluhu. Aktuální stav budiče signalizují bity v registru TWSR. Podle těchto bitů lze určit, jaká akce má následovat. Během doby, kdy je nastaven bit TWINT, drží řadič sběrnice hodinový signál SCL v log. 0, čímž masterovi signalizuje, že potřebuje více času na zpracování operace. Během této doby je sběrnice obsazena a není možné po ní komunikovat. Je tedy nutné, aby obsluha žádosti jednotky TWI byla co nejrychlejší a nebrzdila další možnou komunikaci po sběrnici. Dále je nutné pomocí vhodného timeoutu a watchdogu zajistit, aby v případě zaseknutí obslužného programu byl procesor s jednotkou včas resetován a nebylo bráněno v komunikaci po sběrnici.

6.4 Nastavení parametrů stabilizace

Stabilizační algoritmus vyžaduje nastavení několika konstant. Bohužel pro každý copter je nastavení jiné, protože mají různé motory, senzory i mechanickou konstrukci. Tyto konstanty je nutné nastavovat až po sestavení systému ideálně při prvním letu. Pro nastavování je možné použít dva přístupy. Buďto se nastavování bude provádět za pomoci sériové linky nebo jiného komunikačního kanálu nebo za pomoci přepínačů a potenciometrů. Nastavování pomocí potenciometrů a přepínačů není dostatečně přesné a je omezeno pouze na několik základních konstant. Na druhou stranu použití sériové linky vyžaduje nutnost nosit s sebou notebook nebo jiné zařízení, kterým by se nastavení provedlo.

Po zvážení pro i proti obou řešení jsem se rozhodl použít sériovou linku. Nastavení konstant se provede pouze jednou a následně se hodnoty uloží do eeprom a jejich další změna nebude potřeba. Pro komunikaci s mikrokontrolérem je nutné připojit sériovou linku s napěťovými úrovněmi 3 V na sériové rozhraní číslo 1. Komunikace probíhá rychlostí 19 200 Bd v osmibitovém komunikačním režimu bez parity a s jedním stop-bitem. Pro zahájení komunikace je třeba odeslat za sebou dvakrát znak konce řádku (znak ENTER). Aby bylo možné používat OS Windows i OS Linux, používá

systém pro označení konce řádku znak LF (0x0A). Znak CR, který používá Windows, je ignorován. Aby systém spolupracoval i s nástrojem Hyperterminál, je nutné v tomto programu v nastavení vynutit odesílání znaků LF.

Během letu jdou data přijatá po sériové lince ignorována. Až po té, co je stažena plynová páka na minimální hodnotu, přejde systém do stavu, kdy je možné měnit parametry stabilizačního algoritmu po sériové lince i za pomoci DIP přepínačů. Minimální hodnota plynové páky se určuje při připojení napájecího napětí nebo při resetu. K této hodnotě ji přičten minimální offset, který zajišťuje určitou rezervu. Při zvýšení výchylky plynové páky dojde k opuštění programovacího režimu a přechodu do řídícího algoritmu nastaveného pomocí PID přepínačů. Zároveň dojde ke kalibraci středních poloh vysílačky na základě posledních hodnot. Před zvýšením výkonu je tedy nutné mít všechny kontrolní prvky v neutrální pozici. Pokud není nastavena korektní konfigurace, dojde pouze k opuštění programovacího režimu, nicméně nebude spuštěn žádný řídící proces a tedy motory zůstanou v nulovém stavu, dokud nebude opět stažena plynová páka do volnoběžného stavu.

Nastavování konstant probíhá za pomocí textového rozhraní. Každá z konstant je ve formátu s plovoucí desetinnou čárkou. Pro zadávání a výpis čísel jsem implementoval vlastní funkce, jelikož funkce *sscanf* a *sprintf*, které by načítání a výpis umožňovaly, vyžadují import knihoven, které zabírají příliš velké místo v paměti programu. Jejich import v procesorech s menší pamětí programu omezuje zbylé místo pro vlastní stabilizační a testovací algoritmy. Vstupní hodnoty musí být zadávány ve tvaru, který začíná celočíselnou částí následovanou znakem *e* a po něm exponent, vše bez mezer. Číslo *-0.25* by tedy vypadalo následovně: *-25e-2*. Vypisovaná čísla dodržují matematický tvar desetinného čísla s exponentem, tj. *2,5e-1*. Při zadávání i při výpisu jsou čísla zpracovávána s přesností na maximálně deset platných číslic. Zadání konkrétního parametru se provádí odesláním znaků, které konstantu identifikují a následně vlastní hodnota parametru. Jednotlivé identifikátory je

Parametr	**Význam**
Ki	Koeficient integrální složky
Kp	Koeficient proporcionální složky
Kd	Koeficient derivační složky
Ks	Koeficient citlivosti senzorů
Kt	Časový koeficient – 2,5 ms
Kr	Koeficient zisku přijímače
Pe	Celkový podíl regulátoru ve výstupním signálu
R	Vypsání nastavených hodnot

Tabulka 6.2: Přepínače pro nastavování parametrů stabilizace

možné zobrazit odesláním *-h* a jsou zobrazeny v 6.2. Po odeslání znaků *CR+LF* nebo *LF* je hodnota uložena. Mezery mezi jednotlivými částmi řetězce jsou ignorovány. Pokud bylo načtení hodnoty provedeno úspěšně, odpoví program znaky *OK* a následně zopakuje načtenou hodnotu. Pokud

uživatel uprostřed zadávání udělal chybu nebo chce z jakýchkoliv důvodů zadanou konstantu zrušit, provede tak pomocí klávesy *Backspace*. Po přijetí znaku *0x08* se provede přechod na nový řádek a je zahájeno nové načítání.

6.5 Testování a měření účinnosti stabilizace

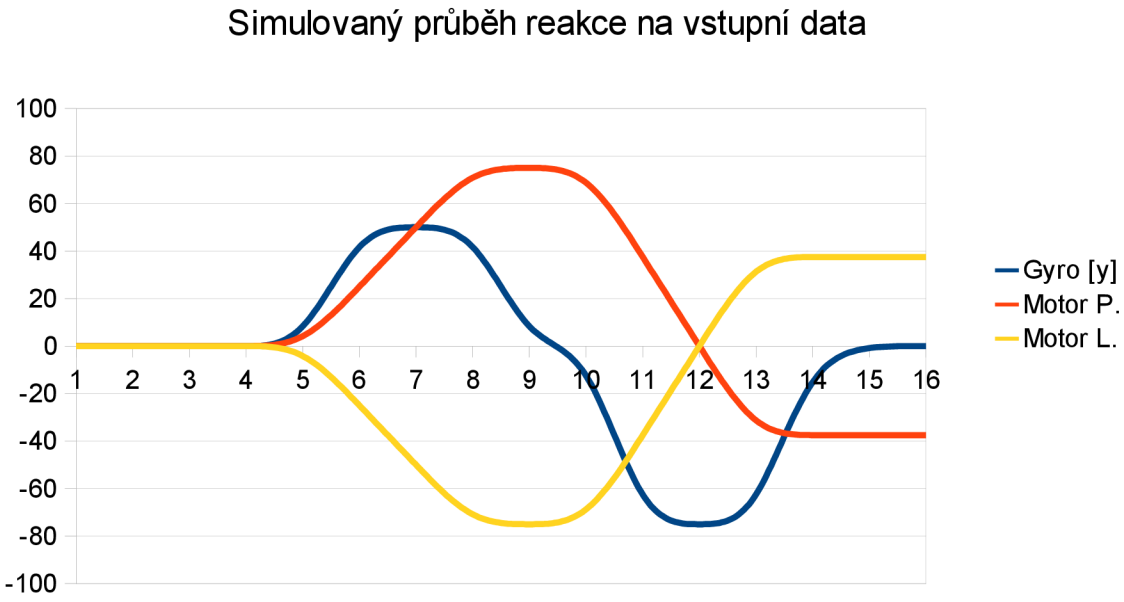
Při vývoji algoritmů stabilizace a během ladění je třeba kontrolovat jejich korektnost, stabilitu a účinnost. Obecně vývoj řídícího software pro létající roboty lze označit za komplikovaný. Před testováním softwarové implementace je nutné ji nejprve bezpečně ozkoušet a odstranit hrubé chyby, které by mohly způsobit poškození modelu při testování na skutečném modelu.

6.5.1 Zkoušky s testovacími daty

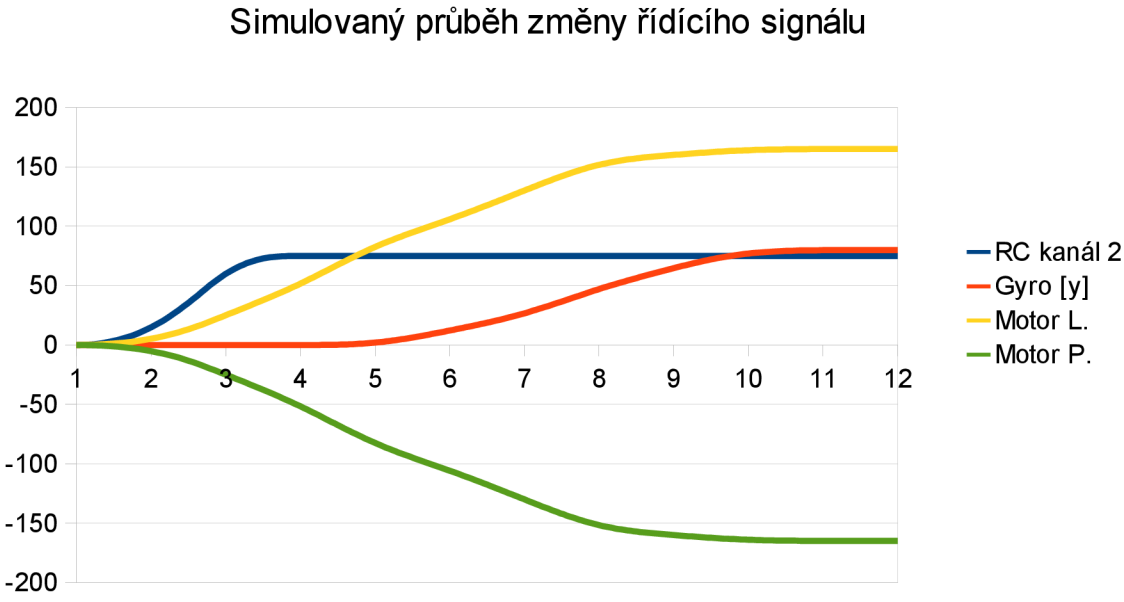
Pomocí předefinovaných testovacích dat je možné ověřit základní nedostatky algoritmu, nicméně bez věrného matematického modelu nelze z tohoto způsobu testování nic přesnějšího zjistit. Pokud by byl model k dispozici, bylo by možné ověřit kompletní činnost. Vzhledem k faktu, že pro sestavený tricopter není matematický model k dispozici, použil jsem testovací data k určení, zda-li odezvy řídícího systému, především jejich smysl, jsou korektní.

Graf 49 zobrazuje vstupní hodnotu úhlové rychlosti podél osy *y* naměřené na gyroskopu, vstupní signál vyjadřující požadovanou úhlovou rychlost a tah motorů a výstupní hodnotu pro přední dva motory, které rotaci okolo podélné osy ovlivňují. Předpokládaná reakce je postupné zvyšování rozdílu tahů mezi motorem 1 a 2, dokud se naměřená úhlová rychlost nevyrovná požadované úhlové rychlosti. Jelikož jsou vstupní data před-generovaná, bude se rozdíl tahu neustále zvětšovat, dokud hodnota nedosáhne maxima a poté zůstane konstantní. Drobné odchylky v grafu oproti předpokládaném stavu pravděpodobně vznikly vyhlazením výsledných hodnot při vytváření grafu.

Graf 6.2 ukazuje chování při krátkodobé změně hodnoty řídícího signálu. Opět je pro zjednodušení zobrazena pouze jedna osa, a to osa podélná. V tomto případě je hodnota naměřená gyroskopem prvně konstantní a po chvíli se začne blížit požadovanému signálu. Rozdíl hodnot tahu jednotlivých motorů by se měl zvětšovat. Jakmile vyrovná hodnota naměřená gyroskopem hodnotu ovládacího prvku, měl by se nárůst korekčních hodnot zastavit.



Graf 6.1: Reakce řídícího systému na simulovaný průběh úhlové rychlosti.

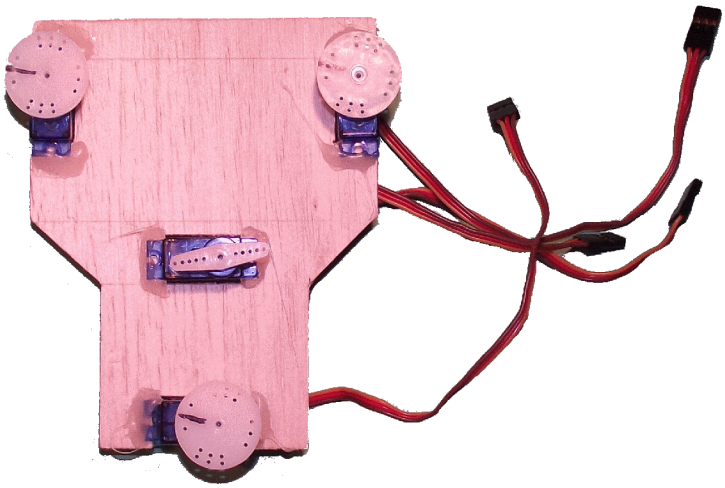


Graf 6.2: Reakce řídícího systému na simulovanou změnu řídícího signálu.

Další příklady testů včetně textového popisu jsou uloženy ve složce *tests* na přiloženém disku.

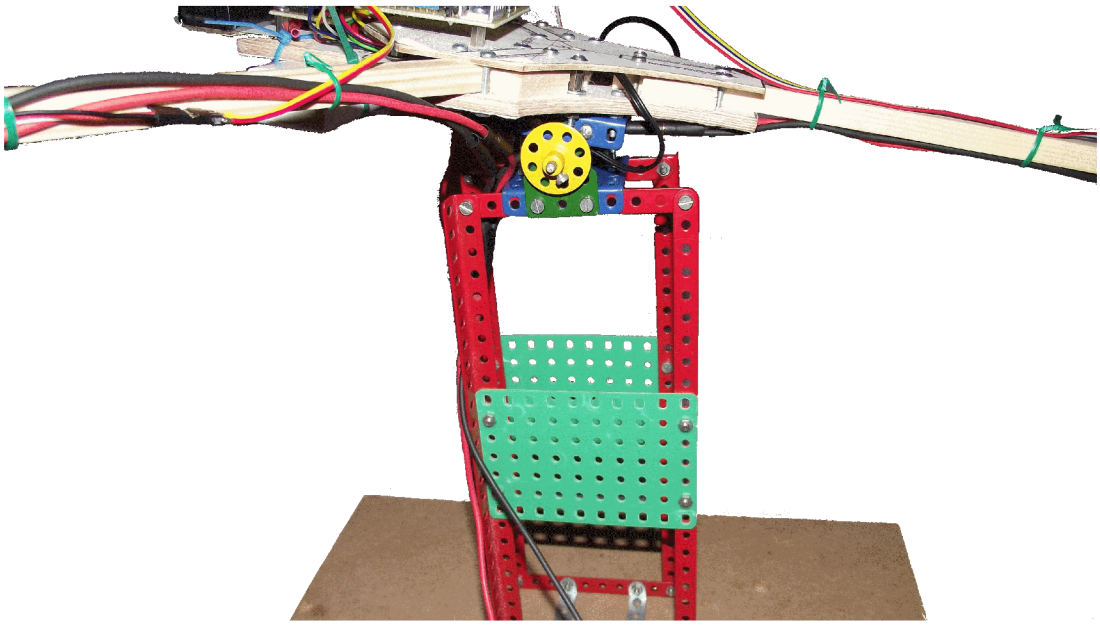
6.5.2 Testovací přípravky

Bohužel testování pomocí předem vygenerovaných dat není dostatečné. Testováním je možné ověřit reakce softwaru na jednotlivé změny vstupních signálů, ale ne celkové chování modelu. Pro důkladnější testování jsem vytvořil dva přípravky, které umožňují testování řídícího systému bez nebezpečí poškození. První z nich umožňuje zobrazovat přibližný výkon každého z motorů a úhel náklonu zadního rotoru. Jsou to čtyři modelářská serva umístěná na maketě těla tricopteru vyříznutého z balsy. Jelikož serva i motory jsou řízeny stejným signálem, bude natočení serva odpovídat otáčkám motoru. Jediným rozdílem mezi servem a regulátorem otáček motoru je jejich citlivost na změny vstupního signálu. Zatímco regulátor využívá mikrokontroléru pro měření délky impulsu, u analogových serv, která byla v přípravku použita, používají k buzení motoru rozdíl délek impulsu přijatého po řídícím kanálu a vygenerovaného monostabilním klopným obvodem. Délka impulsu vygenerovaného tímto obvodem záleží na pozici potenciometru, který je mechanicky připevněn k výstupní osičce serva. Aby byla reakce patrnější, pro účely debugování je vhodné zvýšit citlivost výstupu na zásahy regulátoru pětkrát až desetkrát.



Obrázek 6.2: Přípravek pro ověřování korektnosti reakcí.

Druhý testovací přípravek umožňuje testování kompletně sestaveného tricopteru. Je to jednoduchá konstrukce z Merkuru, která umožňuje rotaci copteru okolo příčné nebo podélné osy. Zároveň však nedovoluje tricopteru vzlétnout. Tento přípravek slouží především k ověření, zda-li reakce software jsou v únosných mezích a zda-li nedochází ke snaze oscilovat.

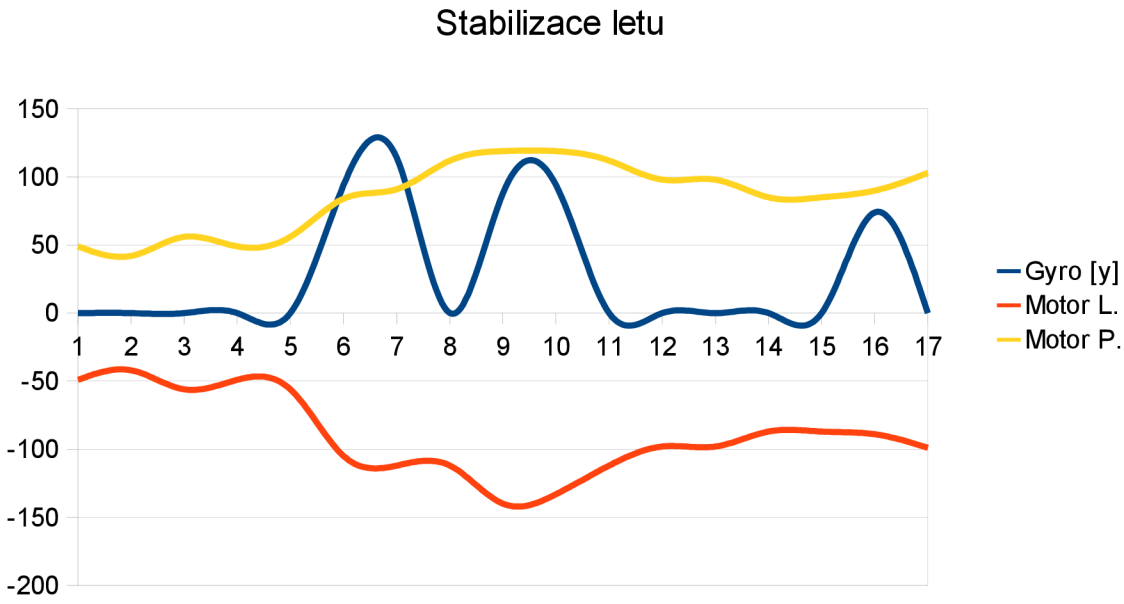


Obrázek 6.3: Testovací přípravek pro ověření reakcí stabilizace.

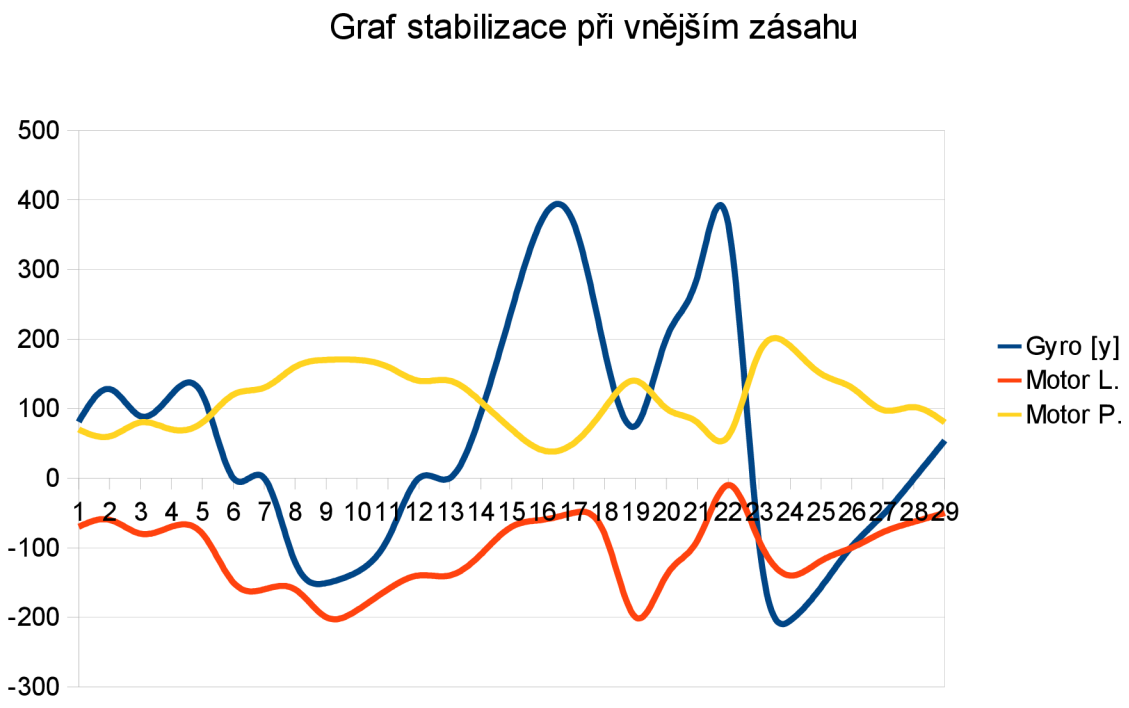
6.5.3 Měření za letu

Měření účinnosti stabilizace bylo také prováděno během letu. Tato měření byla prováděna uvnitř uzavřených prostor, aby byl minimalizováno zkreslení vnějšími vlivy. Simulace větru či jiných jevů, které by mohly rušit let copteru byla provedena pomocí drobných postrčení letícího stroje rukou. I přes veškerou snahu jsou však některé výsledky ovlivněny efekty, které nebylo možné vyloučit. Jednak během letu vzniká pod copterem vzduchový polštář, který napomáhá v nízkých výškách copteru udržet stabilní polohu. I když bylo testování prováděno ve výšce cca 1 m, je možné, že částečně tento efekt let ovlivnil. Druhým vlivem je hmotnost kabelu, kterým byl copter připojen k napájecímu zdroji a k počítači pro záznam letových údajů. Ačkoliv je copter vybaven baterií, pro účely testování je používán počítačový zdroj, protože baterie by se příliš rychle vybila. Také naměřené letové údaje není možné zaznamenávat přímo v řídícím mikrokontroléru. Nejjednodušší způsob jejich záznamu je v počítači.

Následující dva grafy zobrazují vybrané situace během letu. Graf 6.3 ukazuje stabilní let bez vnějších vlivů. V grafu 6.4 byl copter úmyslně vychýlen rukou. Pro přehlednost ukazují grafy pouze naměřené údaje v jedné ose, konkrétně ose y a výstup předních dvou motorů. Výsledky dalších měření jsou uloženy na přiloženém CD. Modrá čára zobrazuje údaj naměřený gyroskopem. Žlutá a oranžová čára zobrazuje regulační odchylku vypočtenou pro přední dva motory. Žlutá čára reprezentuje odchylku motoru, který je na pravém předním nosníku. Tento motor zvyšuje své otáčky při kladné Regulační odchylka závisí také na naměřené hodnotě v ose x, která ovšem v grafu není zobrazena. Ve 13 vzorku byl tricopter vyveden jemným vnějším zásahem z rovnováhy a následuje pokus o utlumení změny úhlové rychlosti.



Graf 6.3: Průběh regulačních hodnot během letu bez vnějších zásahů.



Graf 6.4: Průběh naměřených úhlových rychlostí a regulačních odchylek při vnějším zásahu.

Z obou grafů je patrné, že hodnoty z gyroskopu nejsou plně vyfiltrovány a stroj tedy neustále mírně osciluje. Tento efekt není okem operátora znatelný. Důvod oscilace pravděpodobně způsobují

nosníky motorů, které jsou až příliš pružné. U obou grafů jsou patrné zásahy řídícího systému na nenulovou hodnotu naměřenou gyroskopem. Řídící systém upravuje hodnoty obou motorů současně. Zároveň ovšem dochází i ke stabilizaci polohy v příčném směru. Tento zásah je možné v grafu vypozorovat mezi 18 a 19 vzorkem. Je zde patrné, že korekce pravého motoru stoupá pomaleji než klesá o levého motoru. Současně s tím totiž stoupá výkon zadního motoru, který vyvažuje zápornou rotaci kolem příčné osy. Drobné zkreslení dat je způsobeno průměrováním vzorků řídícím systémem na přípustnou přenosovou rychlost, aby bylo možné je odeslat.

6.6 Komunikační protokol s nadřazeným MCU

Pro účely stabilizace vyšších letových funkcí jako rychlost nebo pozice nejsou možnosti řídícího mikrokontroléru dostatečné. Hlavním nedostatkem je absence jednotky pro výpočet s čísly ve formátu s plovoucí desetinnou čárkou. Druhým nedostatkem je nízká maximální frekvence hodinového signálu. Z těchto důvodů jsem do mikrokontoléru implementoval komunikační rozhraní, které umožňuje připojit nadřazený systém, který by zpracovával data.

Hodnota (hex)	Popis
0xA0	Konec přenášeného pakteu
0xA1	Zrušení speciální funkce následujícího paketu
0xA2	Začátek paketu + přenos dat pro motory a servo
0xA3	Začátek paketu + přenos údajů z akcelerometru
0xA4	Začátek paketu + přenos údajů z gyroskopu
0xA5	Začátek paketu + přenos údajů z magnetometru
0xA6	Začátek paketu + přenos údajů z ultrazvukového dálkoměru
0xA7	Vyhrazeno pro budoucí použití
0xA8	Začátek paketu + přenos dat pro rozšiřující PWM kanály
0xA9	Vyhrazeno pro budoucí použití
0xAA	Vyhrazeno pro budoucí použití
0xAB	Vyhrazeno pro budoucí použití
0xAC	Volné pro libovolné využití
0xAD	Volné pro libovolné využití
0xAE	Volné pro libovolné využití
0xAF	Volné pro libovolné využití

Tabulka 6.3: Seznam bytů se speciálním významem.

Fyzická vrstva komunikace využívá asynchronní sériovou linku. Toto rozhraní velmi rozšířeno a je jím vybavena většina mikrokontrolérů. Přenos probíhá v osmibitovém režimu po paketech. Pro označení začátku paketu, typu dat a konce paketu je vyhrazeno 15 hodnot. Šestnáctý byte slouží k označení zrušení významu následujícího bytu. Význam a hodnoty těchto bytů jsou zobrazeny v tabulce 6.3. Pro zajištění spolehlivého přenosu je každý paket vybaven CRC kontrolním součtem. Kontrolní součet je vypočten jako výlučný součet přenášených bytů, tedy i vložených speciálních hodnot. Při kontrole musí platit, že výlučný součet všech bytů včetně CRC musí být roven nule. Každý paket začíná startovacím bytem z výše uvedené tabulky, který v sobě zároveň uchovává

informaci o typu přenášených dat. Následuje kontrolní byte a za ním jsou přeneseny datové byty. Paket je poté ukončen stop bytem.

Vytvoření a odeslání paketu umožňuje funkce *sendDataPacket*. Tato funkce přijímá tři parametry. První je ukazatel na pole bezznaménkových znaků, která mají být odeslána a druhý parametr obsahuje počet a třetí označuje typ přenášených dat. Funkce vrací počet znaků, které je možné maximálně odeslat. Pokud je tato hodnota nižší požadovaný počet znaků, nebudou data odeslána a software musí zavolat tuto funkci později, až dojde k částečnému vyprázdnění bufferů. Pokud je ve výstupním bufferu dostatek volného místa, funkce přidá k datům počáteční byte, který zároveň označuje typ přenášených dat. Dále je vypočtena hodnota kontrolního součtu a je každému výskytu bytu uvnitř přenášených dat předřazena hodnota, která tento speciální význam ruší. Na konec paketu je přidán stop byte a paket je vložen do výstupní fronty. Pokud právě neprobíhá odesílání, je vyjmut první byte z fronty a vložen do datového registru jednoty USART. Další byty jsou odeslány v rámci obsluhy přerušení.

Přijaté pakety detekuje stavový automat. Při příjmu nových dat je vygenerováno přerušení, které přečte přijatý byte z jednotky USART a zpracuje jej na základě aktuálního stavu. Pokud byl celý paket dekódován bezchybně a pokud je v příchozí frontě volné místo, dojde k jeho uložení. Fronta příchozích paketů je vytvořena jako rotační pole o kapacitě čtyř položek. Každá položka obsahuje ukazatel na paměťový blok s vlastním paketem. První byte paketu udává typ, druhý počet bytů a další pakety obsahují data. Počet dat je přesně specifikován velikostí paketu, maximálně však může být v paketu přenášeno 16 bytů uživatelských dat. Pro vyčtení dat z bufferu slouží funkce *revDataPacket*, která vrací nulu, pokud nejsou žádná data k dispozici. V opačném případě funkce zkopíruje data do paměťového prostoru, který byl funkci předán jako ukazatel, nastaví korektní hodnotu proměnné indikující typ paketu a vrátí počet bytů, které byly překopírovány. Funkce nekontroluje obsah mezí a povinností uživatele zajistit, aby na předané adrese bylo k dispozici minimálně šestnáct bytů místa.

7 Závěr

V této práci byla popsána konstrukce třírotorového copteru. Mechanická konstrukce tricopteru, která byla v kapitole 4 popsána, se v praxi ukázala jako dostatečná, ale během testů jsem objevil několik možných úprav. Původně použité uhlíkové trubičky nejsou bez přilepení použitelné, protože mají tendenci se protáčet a pohybovat. Vhodným řešením tohoto problému by mohlo být jejich přilepení do speciálně připravených úchytů s čtvercovým nebo obdélníkovým průřezem. Až tento úchyt se následně připevní k jednotlivým dílům copteru. Dřevěné nosníky, u kterých se problémy s uchycením nevyskytují, mají tendenci až příliš vibrovat. Ačkoliv má toto tlumení rázů částečně prospěšný účinek, velká část vibrací se přenáší až k senzorům a je zaznamenána. Proto je vhodné tyto vibrace minimalizovat v maximální možné míře. Řešení existuje několik, avšak jako nejlepší lze označit nahrazení jednoho nosníku několika menšími, která by byly navzájem slepeny. Pro ohnutí lepeného nosníku je zapotřebí větší síla než je tomu u nosníku z jednoho kusu.

Použité senzory jsou dostatečné pro stabilizace letu. Všechny senzory bohužel snímají i vibrace těla modelu, protože jsou k němu přilepeny. Zlepšení kvality naměřených údajů lze dosáhnout změnou uchycení senzorů. Namísto přímého přilepení k tělu modelu by pomohlo vložit mezi tělo a senzory speciální tlumící podložky. Druhým řešením je vylepšení filtrů, které odstraňují tyto vibrace. Použití magnetometru uvnitř tricopteru není vhodné. Motory a regulátory vytvářejí příliš velký magnetický šum a údaje ze senzoru nejsou použitelné.

Hardware řídícího systému je pro základní stabilizaci pro rekreační používání tricopteru dostatečný. Protože se jedná o platformu pro vývoj a testování, je řídící mikrokontrolér značně naddimenzován. Pro účely základní stabilizace by bylo možné namísto dvou mikrokontrolérů použít pouze jeden. Kvůli převodu signálu z RC soupravy je nutné, aby řídící mikrokontrolér byl vybaven řadičem PCINT. Proto v úvahu připadají pouze mikrokontroléry ATmega88A nebo ATmega168A. Konkrétní volba záleží pouze na velikosti stabilizačního a řídícího algoritmu. Oba uvedené mikrokontroléry mají omezený počet vývodů, což nutně znamená, že je třeba omezit počet připojených periférií. Nicméně i s omezeným počtem pinů bude možné implementovat základní vlastnosti řídícího systému.

Navržený řídící a stabilizační systém je schopen tricopter za letu dostatečně stabilizovat. Jeho největší komplikací je ovšem velké množství parametrů PID regulátoru, které je třeba před prvním letem správně nastavit.

Dalším rozšířením řídícího systému tricopteru je použití výkonnějšího mikrokontroléru, který bude zpracovávat údaje ze senzorů například pomocí Kalmanova filtru. S přesnějšími údaji ze senzorů bude možné implementovat podrobnější metody stabilizace a řízení. Systém by tedy mohl umožňovat plně autonomní řízení i bez zásahů operátora. Také je možné vybavit systém přijímačem

a dekodérem GPS signálu. Dalším rozšířením je vybavení tricopteru stabilizovanou plošinou, na kterou by bylo možné uchytit kameru nebo fotoaparát.

V modelářských diskuzních fórech jsem objevil informaci, že ESC jsou vybaveny mikrokontroléry AVR a DPS těchto ESC umožňuje vyvedení a připojení sběrnice I2C. Také existuje vhodný firmware, který po nahrání do ESC umožňuje jejich řízení přímo pomocí sběrnice. Odpadá tím nutnost generovat PWM signál pro řízení motorů a také dochází ke zpřesnění komunikace. Mimo to je také komunikace a tedy i odezva zrychlena, protože nedochází k aktualizaci pouze jednou za 20 ms, ale vždy, když bude nová hodnota vypočtena. Na druhou stranu tímto zásahem dojde k porušení záruky na ESC.

Literatura

[1] KOLÁČEK, Ludvík. *Učebnice pilota vrtulníku*. 1. vyd. Olomučany: Akademické nakladatelství CERM, 2009. ISBN 978-80-7204-627-0.

[2] NEŠTRÁK, Dušan a Ján PIĽA. Aerodynamika, konstrukce a systémy vrtulníků: studijní modul 12. Vyd. 1. Brno: Akademické nakladatelství CERM, 2006. ISBN 80-720-4484-2.

[3] How helicopters fly and are controlled. In: *RC Airplane World*: *your Guide to flying radio control airplanes & aircraft* [online]. 2002 [cit. 2012-03-22]. Dostupné z: http://www.rc-airplane-world.com/how-helicopters-fly.html

[4] BORENSTEIN, J., H. EVERET., L. FENG,. Where am I? Sensors and Methods for Mobile Robot Positioning [online]. 1996 [cit. 2011-12-27].

[5] RIPKA, P., ĎAĎO, S.:Senzory a převodníky. 1. vyd. Praha: Vydavatelství ČVUT, 2005, 136 s. ISBN 80-010-3123-3.

[6] Integrované MEMS gyroskopy. HW.cz [online]. 11.10.2009 [cit. 2012-01-08]. Dostupné z: http://automatizace.hw.cz/integrovane-mems-gyroskopy

[7] Altimeter. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-01-08]. Dostupné z: http://en.wikipedia.org/wiki/Altimeter

[8] WINDESTÅL, David. RCExplorer [online]. © 2011 - 2012 [cit. 2012-01-08]. Dostupné z: http://rcexplorer.se

[9] Tricopter. In: *RC-Mania*: *modelářské diskuzní fórum* [online]. [2010] [cit. 2012-02-10]. Dostupné z: http://www.rcmania.cz/viewtopic.php?f=106&t=30838&sid=24f5d4e5a40a468f3e5e0d2c86e747a1

[10] Building a tricopter. In: *Radial Mind* [online]. 2011-05-26 [cit. 2012-05-18]. Dostupné z: http://www.radialmind.org/projects/trycopter

[11] Turnigy 2217-20. In: Hobby King [online]. 1999-2011 [cit. 2012-01-02]. Dostupné z: http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=5691

[12] Turnigy Plush 25amp Speed Controller. In: Hobby King [online]. 1999-2011 [cit. 2012-01-02]. Dostupné z: http://www.hobbyking.com/hobbyking/store/__2163__TURNIGY_Plush_25amp_Speed_Controller.html

[13] Teorie řízení serva. In: *Serva.cz* [online]. © 2008 [cit. 2012-01-18]. Dostupné z: http://www.serva.cz/rizeni-serva-teorie/

[14] *ADXL345*: *Digital Accelerometer* [online]. Honeywell, 2009, 24 s. [cit. 2011-02-01]. Dostupné z: http://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[15] *ITG-3200*: *Product Specification* [online]. Honeywell, 2009, 39 s. [cit. 2011-02-01]. Dostupné z: http://www.sparkfun.com/datasheets/Sensors/Gyro/PS-ITG-3200-00-01.4.pdf

[16] *HMC5883L*: *3-Axis Digital Compass IC* [online]. Honeywell, 2011-03-01, 20 s. [cit. 2011-02-01]. Dostupné z: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf

[17] SRF02 Ultrasonic range finder: Technical Specification. *Robot electronics* [online]. 2012 [cit. 2012-02-01]. Dostupné z: http://www.robot-electronics.co.uk/htm/srf02tech.htm

[18] *Datasheet: Atmega128A*. Atmel, 2011-02-01, 378 s. Dostupné z: http://www.atmel.com/Images/doc8151.pdf

[19] *Datasheet: Atmega48A*. Atmel, 2011-05-01, 34 s. Dostupné z: http://www.atmel.com/Images/8271S.pdf

[20] *7805*: *3-Terminal 1A Positive Voltage Regulator* [online]. FairChild Semiconductor, 2001, 29 s. [cit. 2012-02-10]. Dostupné z: http://www.datasheetcatalog.com/datasheets_pdf/7/8/0/5/7805.shtml

[21] *LM317*: *3-Terminal Positive Adjustable Regulator* [online]. FairChild Semiconductor, 2001, 9 s. [cit. 2012-02-10]. Dostupné z: http://www.datasheetcatalog.com/datasheets_pdf/L/M/3/1/LM317.shtml

[22] Stabilizovaný zdroj s LM317. In: *Elektronika*: *zapojení, návody a konstrukce* [online]. 20.5.1999 [cit. 2012-02-10]. Dostupné z: http://belza.cz/pwrsply/317zdroj.htm

[23] GOOK, Michael. *Hardwarová rozhraní*: *průvodce programátora*. Vyd. 1. Brno: Computer Press, 2006, 463 s. ISBN 80-251-1019-2.

[24] USB to RS232 Adapter with FT232. In: *Electronics DIY* [online]. 2005-02-23 [cit. 2012-01-10]. Dostupné z: http://electronics-diy.com/electronic_schematic.php?id=717

[25] *FT232R*: *USB UART IC* [online]. FTDI Chip, 2011-05-01 [cit. 2012-02-10]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[26] AVRUSB500v2: programátor AVR na USB. In: *Wagnerovi.cz* [online]. 2009-01-27 [cit. 2012-02-10]. Dostupné z: http://wagnerovi.cz/view.php?nazevclanku=avrusb500v2-programator-avr-na-usb&cisloclanku=2009010001

[27] AvrUsb500v2: an open source Atmel AVR Programmer, stk500 V2 compatible, with USB interface. In: *Tuxgraphics.org*: *the place for avr microcontroller, DIY electronics and science fun* [online]. 2007-05-31 [cit. 2012-02-10]. Dostupné z: http://tuxgraphics.org/electronics/200705/article07052.shtml

[28] Autonomous quadrocopter. LAOROSA [online]. 31.5.2010 [cit. 2011-12-27]. Dostupné z: http://www.design-laorosa.com/2010/05/autonomous-quadrocopter.html

[29] Understanding The RC Quadrocopter. RC Heli Fun [online]. [2010] [cit. 2012-01-02]. Dostupné z: http://www.rchelicopterfun.com/quadrocopter.html

[30] SMITH C. L., *Digital Computer Process Control,* International Textbook Company, 1972.

[31] OGATA, Katsuhiko, Modern control engineering /Boston :Prentice Hall,c2010. 5th ed. x, 894 s. ISBN 978-0-13-615673-4

[32] BLAHA, P., VAVŘÍN. P. Řízení a regulace 1. Brno: VUT Brno: 2005. 214 s.

[33] PID contrller. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2012-05-16 [cit. 2012-05-17]. Dostupné z: http://en.wikipedia.org/wiki/PID_controller

[34] PID process control, Cruise Control example. In: Th Code Project: Your Development Resource [online]. 2009-05-14 [cit. 2012-03-15]. Dostupné z: http://www.codeproject.com/Articles/36459/PID-process-control-a-Cruise-Control-example

[35] KLÁN, P.: Ziegler-Nicholsovo nastavení PID regulátoru – retrospektiva. Automa, 2000, č. 4, s. 53.

[36] How to hack the PPM signal from any receiver (Futaba) with Arduino. In: *DIY Drones* [online]. 2008-6-8 [cit. 2012-03-17]. Dostupné z: http://diydrones.com/profiles/blogs/705844:BlogPost:38393#

[37] Freescale Race Challenge 2012: Soutěž samořídících autíček na autodráhu pokračuje. In: *HW.cz* [online]. 2011-10-01 [cit. 2012-04-08]. Dostupné z: http://www.hw.cz/teorie-a-praxe/mimochodem/freescale-race-challenge-2012-soutez-samoridicich-auticek-na-autodrahu

Seznam obrázků

Seznam tabulek

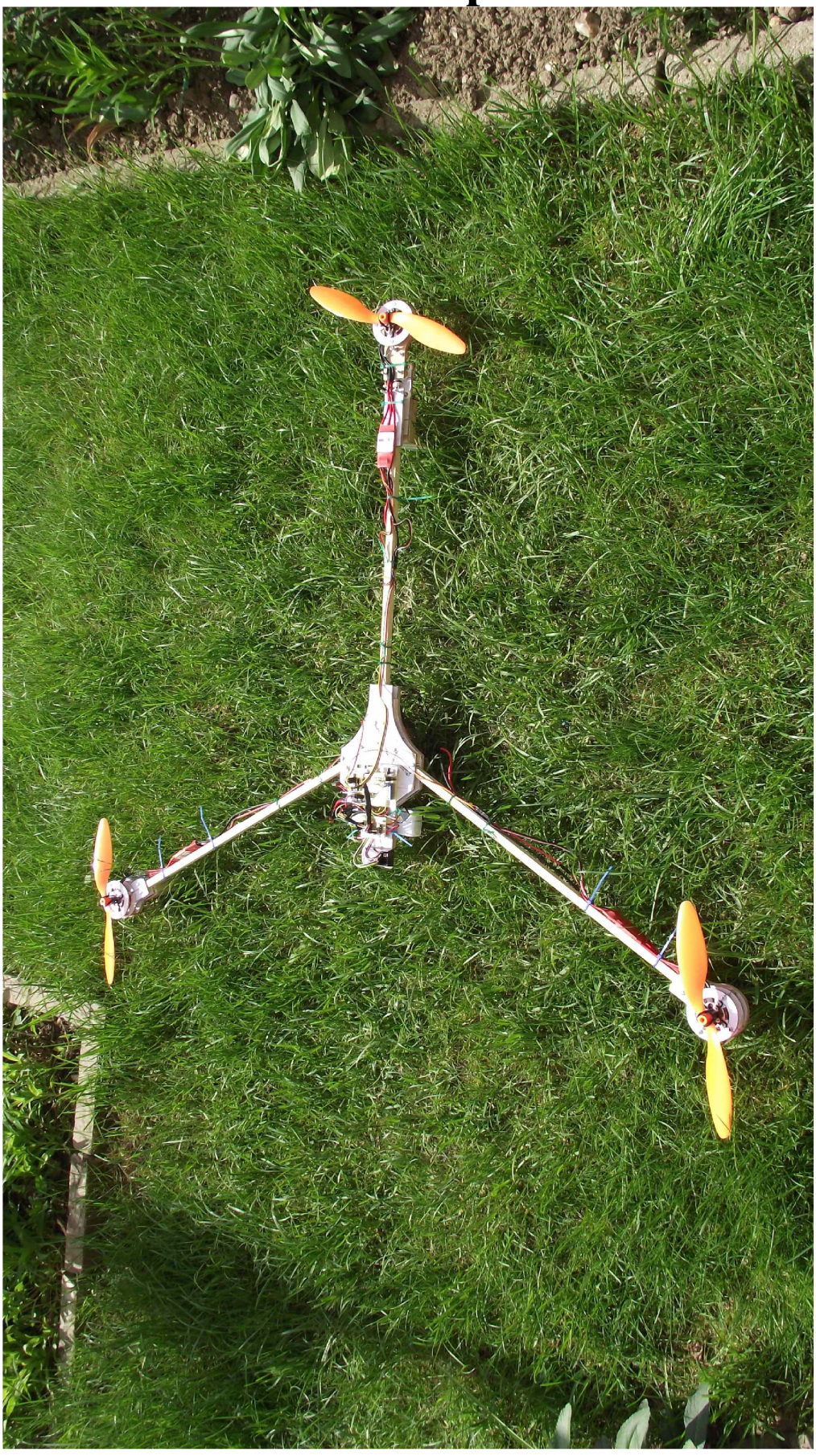
Seznam grafů

Seznam použitých zkratek

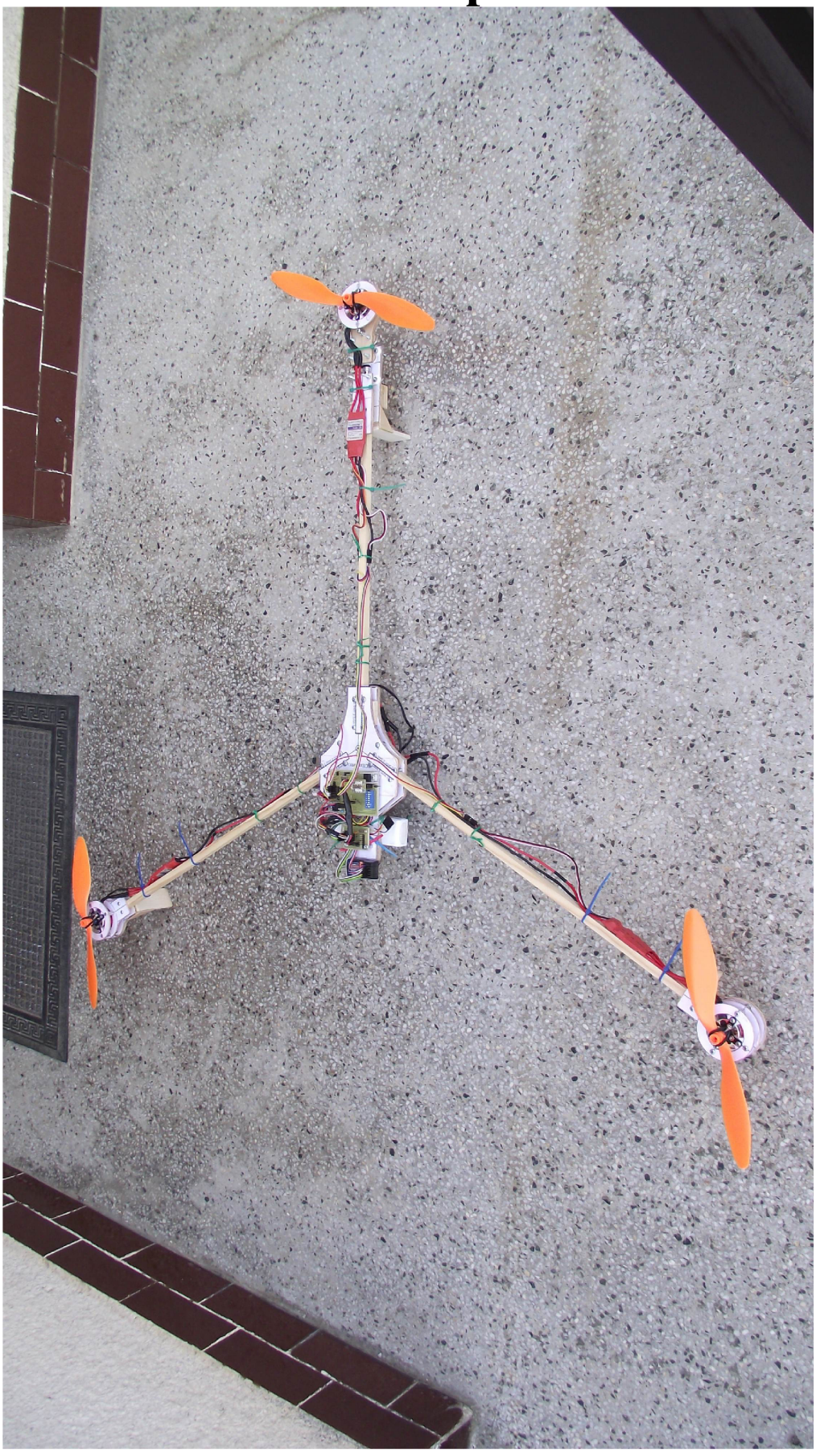
USB	univerzální sériová sběrnice (universal serial bus)
UART	univerzální asynchronní přijímač/vysílač (universal asynchronnous receiver/transmitter)
PID regulátor	proporcionální, integrační a derivační regulátor (proportional–integral–derivative)
OCR	registr výstupního porovnání (output compare register)
ICR	registr zachycení události (input capture register)
BEC	obvod pro emulaci baterie (battery emulation circuit)
ESC	elektronický regulátor otáček (electronic speed controller)
PWM	pulsně šířková modulace (pulse-width modulation)
RC	dálkové ovládání (remote controller)
TWI	dvou-vodičové rozhraní (two wire interface)
I2C	sběrnice pro propojení integrovaných obvodů (inter integrated circuit)

Seznam příloh

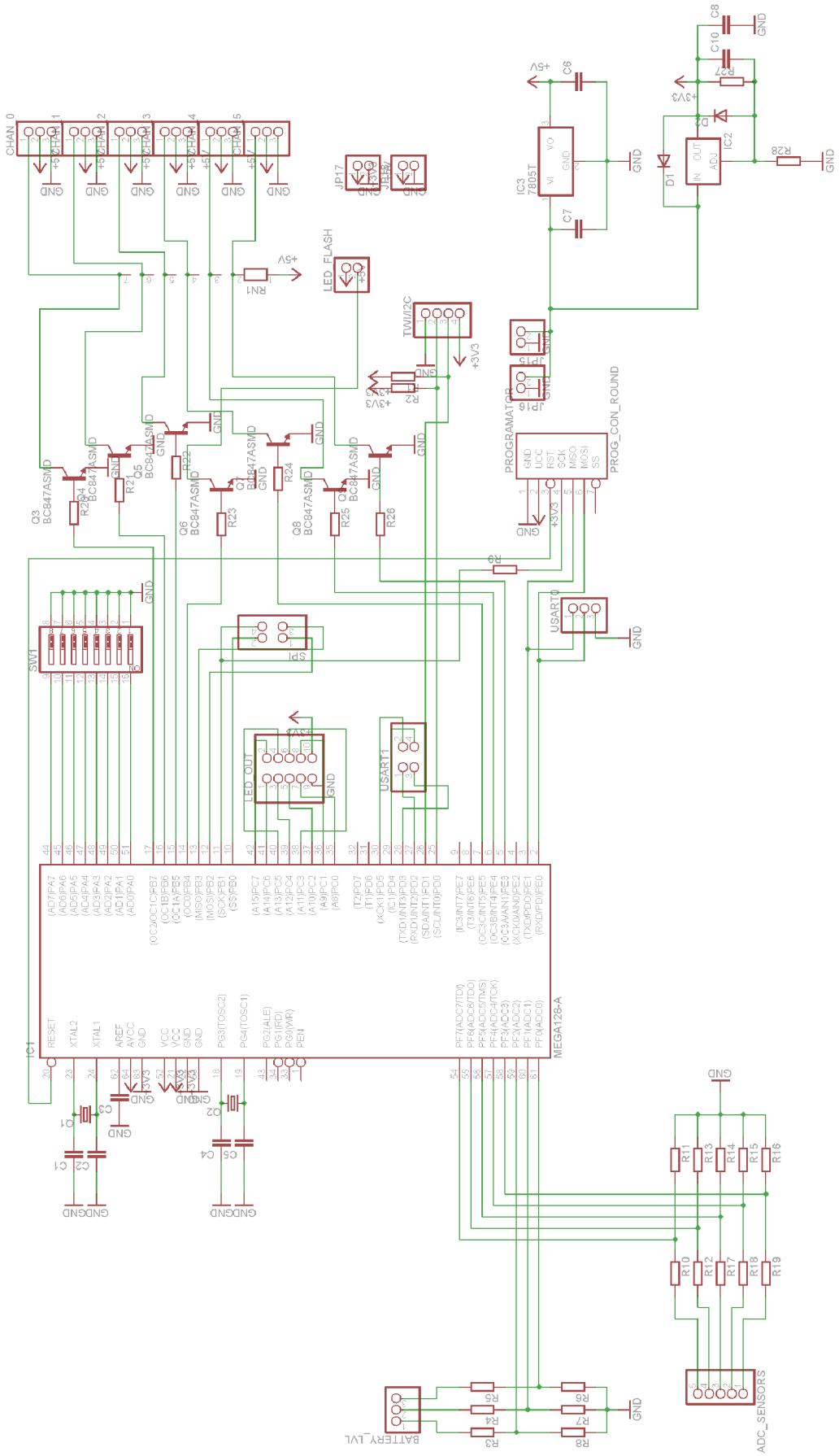
Příloha 1: Obrázek tricopteru 1



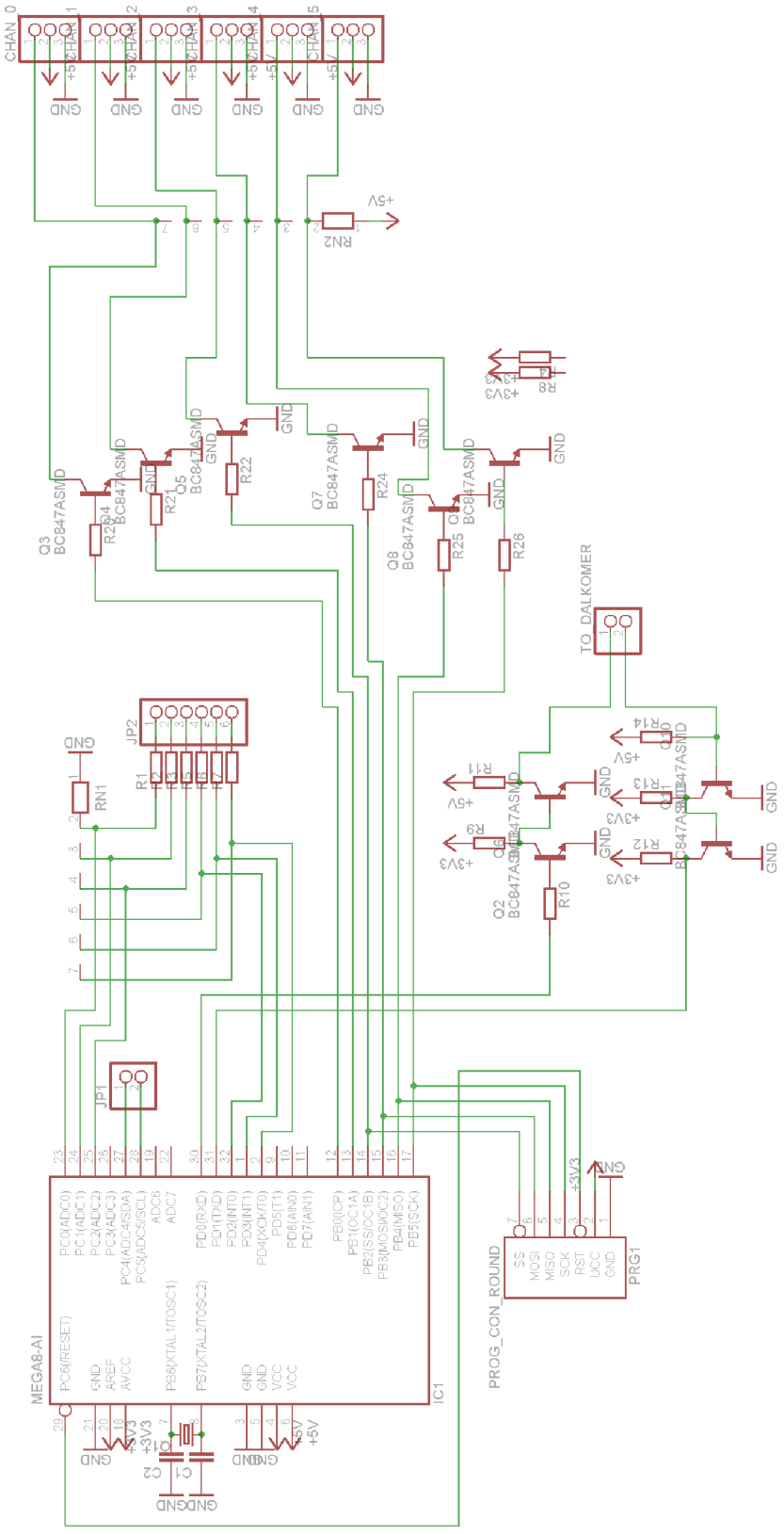
Příloha 2: Obrázek tricopteru 2



Příloha 3: Schéma zapojení – hlavní MCU



Příloha 4: Schéma zapojení – koprocesor



Příloha 5: Obsah CD

- thesis
  - Složka se zdrojové texty diplomové práce
- sources
  - Složka se zdrojové kódy pro AVR studio pro jednotlivé mikrokontroléry
- images
  - Složka s obrázky tricopteru
- board
  - Složka s schématy a návrhy DPS