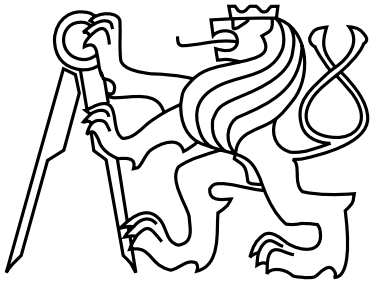
Na tomto místě bude oficiální zadání vaší práce

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu



Diplomová práce

Konstrukce vícerotorového dronu pro průzkum terénu

Bc Jakub Halák

Vedoucí práce: Ing. Pavel Kubalík, Ph.D

Studijní program: Informatika, magisterský

Obor: Projektování číslicových systémů

7. května 2014

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Kněžmostě dne 7.05.2014. .....

Poděkování

Rád bych zde poděkoval vedoucímu práce Ing. Pavlu Kubalíkovi PhD. za jeho ochotu okamžitě se mnou řešit problémy, které nastaly během tvorby této práce, jeho bleskovou odezvu při komunikaci a celkovou vstřícnost.

Zvláštní poděkování patří mým rodičům za jejich podporu po celou dobu mého studia.

Abstract

This master thesis is focused on design and construction of quadrotor drone for terrain surveillance. Drone can be controlled by RC radio transmitter. Dron is equipped with sensors for measurement of motion (gyroscope, accelerometer, magnetometer, GPS and sonar) and barometer for measurement of pressure. Drone is able to communicate with Raspberry PI and receive commands from it. As part of this work was to design a construction and control board to control all peripherals.

Abstrakt

Práce se zabývá návrhem a realizací konstrukce vícerotorového dronu pro průzkum terénu. Dron je možné ovládat pomocí RC vysílačky. Dron je vybaven senzory pro určení jeho aktuální polohy, náklonu a rychlosti v prostoru (gyroskop, akcelerometr, magnetometr, GPS a sonar) a dále je přidán barometr pro měření tlaku vzduchu. Dále je implementováno rozhraní pro komunikaci s Raspberry PI, kterým může být ovládán. V rámci této práce byla navrhnuta konstrukce a řídící deska pro ovládání všech periferii.

Obsah

Seznam obrázků

Seznam tabulek

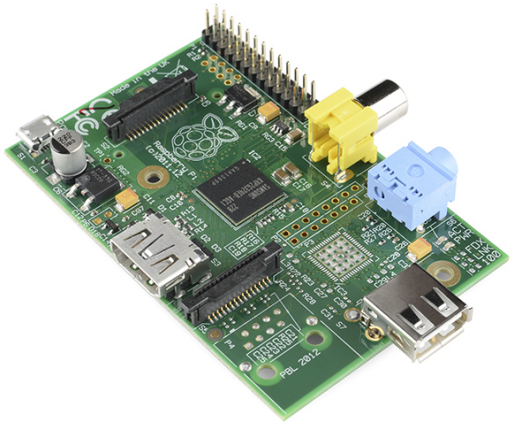
Kapitola 1

Úvod

Úkolem této práce je navrhnout a zrealizovat konstrukci vícerotorového dronu pro průzkum terénu, který je možné ovládat rádiovým vysílačem. Dron, neboli bezpilotní letoun, také označován jako UAV, je létající zařízení, které dokáže létat bez pilota. První bezpilotní letoun byl sestrojen již v roce 1916. UAV se využívaly a dodnes využívají při pořizování fotografií ze vzduchu.

V rámci práce byla navržena řídící deska pro vícerotorový dron. Řídící jednotka je navržena tak, aby se dala použít i na jiné projekty než je vícerotorový dron, jako jsou vozidla, letouny a další. Dron je snadno rozšiřitelný o další periferie.

Jednou z hlavních možností rozšíření je komunikace s Raspberry PI [5], od kterého dron může přijímat příkazy. Raspberry PI (obr. 1.1) je miniaturní počítač velikosti kreditní karty, který vyvíjí britská společnost Raspberry PI Foundantion s cílem podpořit výuku informatiky ve školách. Jeho základem je SoC (System on Chip) obsahující procesor založený na architektuře ARM s taktem 700MHz, grafický procesor a paměť RAM o kapacitě 512MB RAM. Dron je možné dále rozšířit o jiné druhy ovládání, například pomocí UART, nebo sběrnice SPI.



Obrázek 1.1: Raspberry PI.

Stabilizace dronu je prováděna pomocí PID regulátorů (sekce 5.2.1.2). Dron by měl být schopný unést závaží o maximální hmotnosti 500 g.

1.1 Rozvržení práce

Diplomová práce je rozdělena do sedmi hlavních kapitol. Každá kapitola se zabývá různými tématy týkající se konstrukce dronu. První kapitola se zabývá definicí parametrů dronu. Druhá kapitola rozebírá různá řešení, která existují na trhu a popisuje řešení vlastní. Následující kapitola vysvětluje jak vypadá typická konstrukce vícerotorového dronu a jak dron funguje. Kapitola čtvrtá popisuje realizaci řídící desky, zatímco pátá kapitola popisuje realizaci softwaru. Předposlední kapitola se zabývá testováním základních komponent, které jsou potřeba pro stabilní let a poslední kapitola shrnuje poznatky celé práce.

1.2 Požadavký práce

Hlavní požadavky diplomové práce jsou

- Navrhnout a sestavit mechanickou konstrukci.
- Navrhnout a zrealizovat řídicí desku pro ovládání periférií.
- Naprogramovat knihovny pro ovládání periférií.
- Naprogramovat základní program pro létání.

Kapitola 2

Analýza a návrh řešení

Na trhu se dnes pohybuje ohromné množství různých dronů, některé spadají mezi komerční produkty, jiné zase do OpenSource. Mezi konstrukcemi převládají čtyřmotorové drony, tzv. quadrocoptery. V menší četností se pak objevují tripletocoptery a hexacoptéry. V ojedinělých případech můžeme spatřit a octocoptéry. Quadrokoptéry dominují nad ostatními konstrukcemi z důvodu jednoduššího řízení. Obliba používání dronů je zapříčiněna jejich schopností dobře manévrovat a jejich velikost jim umožňuje se dostat do hůře dostupných míst. Nejmenší drony jsou jen 15 cm veliké. V profesionální sféře je využívají hlavně hasiči, policie, armáda a v poslední době i televize k natáčení reportáží z nepřístupného terénu. K vývoji a popularizaci dronů velmi přispěli modeláři.

2.1 Nabídka trhu

Po prozkoumání na trhu dostupných řešení, byla vybrána dvě nejpoužívanější. Nejznámějšími řešeními jsou ArduPilot [6] a AeroQuad [7]. Existuji však další, jako například BlueCopter [8], ArduQuad [9]. Zaměřím se však pouze na řešení ArduPilot, jelikož má za sebou dlouhý vývoj a jedná se o nejpoužívanější řešení.

2.1.1 ArduPilot

Na platformě ArduPilot dnes pracuje velké množství nadšenců z celého světa. Platforma za dobu své existence prodělala nemalé změny a mnoho vylepšení, které ji přivedly do stavu v jakém je dnes. Jedná se o velmi variabilnířešení ,které dovoluje ovládat velmi široké spektrum zařízení, počínaje quadrokoptérami, triplecoptery, hexacoptery a octacoptery, přes letadla a vrtulníky a vozidly konče. Součástí platformy je řídící deska označena APM, která je osazena mikrořadičem od firmy ATmel ATmega2560 [10]. Jedná se o RISC mikrořadič, který pracuje na frekvenci až 16MHz. Mikrořadič je založen na Harwardské architektuře a obsahuje flash paměť pro program 256KB, 8KB SRAM a 4KB EEPROM. ArduPilot je založen na platformě Arduino, která je velmi rozšířena mezi hardwarovými vývojáři.

Obrázek 2.1: Řídící deska ArduPilot APM 2.0.

2.1.2 Vlastní řešení

Pro vlastní řešení bylo rozhoduto použít také platformu Arduino, jelikož jedním z požadavků bylo snadné rozšiřování softwarové výbavy. Platforma Arduino používá mikrořadiče od firmy ATmel z rodiny ATmega. První prototyp byl postaven na mikrořadiči ATmega328p [12], který stačil na ovládání základních funkcí. S dalším vývojem se zvětšovaly nároky na paměťový prostor a bylo nutné použít mikrořadič s větší pamětí. Dalším mikrořadičem, který je kompatibilní s platformou Arduino a jeho paměťový prostor je dostačující je právě již zmíněná ATmega2560. Ve finálním návrhu řídící desky byl zvolen právě tento model mikrořadiče. Řídící deska je navržena hlavně k řízení quadrocoptery, je však možné pomocí ni řídit i triplecoptery, letadla, vozidla a vrtulníky. V tabulce (tab.2.1) je znázorněno porovnání specifikací jednotlých řešení.

	Vlastní řešení	**ArduPilot**
Mikrořadič	ATmega2560	ATmega2560
PWM vstupy	6x	8x
PWM výstupy	4x	8x
Senzory	gyroskop, akcelerometr, barometr, magnetometr	gyroskop, akcelerometr, barometr
Rozšiřitelnost	GPS, telemetrie, Rapberry Pi ready, sonar	GPS, telemetrie
Rozměry	50 x 50 mm	66.5 x 40.5 mm

Tabulka 2.1: Specifikace jednotlivých řešení

2.1.3 Cenová analýza

Na trhu existuje nemalé množství quadrocopter. Jejich cena se velice různí. Od hraček za 1 500 Kč až po profesionální zařízení, jejichž cena se může vyšplhat až na stovky tisíc korun. Běžné modelářské quadrocoptery se pohybují od 5 000 Kč do 10 000 Kč. Pokud ale chceme kvalitnější zařízení, musíme sáhnout po dražší variantě.

Tabulka 2.2 ukazuje cenu výroby jednoho kusu vlastního řešení řídící desky. Pokud by se výroba změnila na sériovou, můžeme očekávat významný pokles ceny. Cena je u vlastního řešení jednou z výhod, které nabízí oproti ostatním stejně vybaveným řešením. Další výhodou je možnost úpravy dle vlastních potřeb a uživatel proto neni odkázán na nákup drahých komponent.

Název součástky	**Cena [Kč]**
ATmega2560	240
MPU-6000	700
HMC5883L	60
MPL3115A2	70
TPS62112R	120
AMS1117	15
Pasivní součástky	175
Polovodiče	120
DPS	500
Celkem	2000

Tabulka 2.2: Cena výroby vlastního řešení při výrobě jednoho kusu.

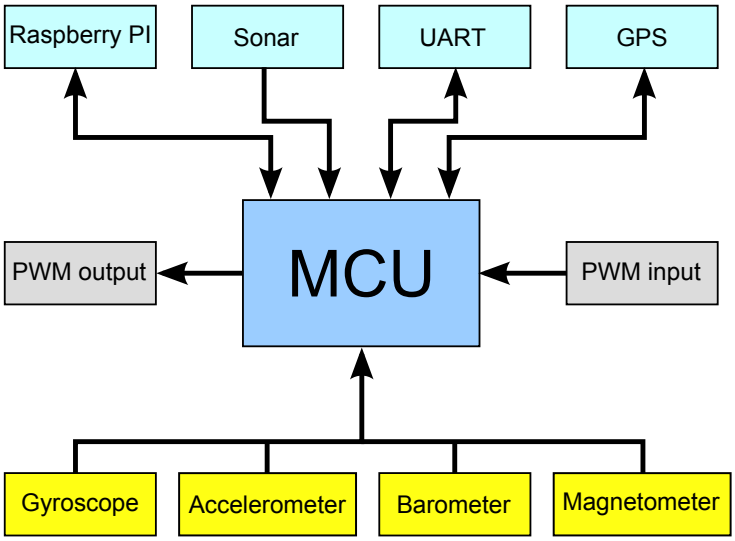
V tabulce 2.3 je vidět, že majoritu ceny tvoří konstrukce quadrocoptery, která je stále stejná nezávisle na řešení řídící desky, kterou použijeme. Veškeré komponenty byly vybrány s ohledem na co nejnižší konečnou cenu, ale aby se stále zachovali požadované vlastnosti quadrocoptery.

Společné komponenty	
Název komponenty	**Cena [Kč]**
4 x Elektromotor	3600
4 x Vrtule	1000
4 x Regulátor otáček	1100
Baterie	1400
Radiový vysílač a přijímač	2100
Konstrukce	600
Celkem	9800
Řídící desky	
ArduPilot	3200
Vlastní řešení	2000

Tabulka 2.3: Cena výroby quadrocoptery.

2.2 Návrh desky

Řídící deska je hlavní součástí celé quadrocoptery. Na desce jsou vyvedeny základní komunikační a rozšiřující porty. Blokové schéma desky je na obrázku 2.2.



Obrázek 2.2: Blokové schéma zapojení řídící desky

Mezi základní a nejdůležitější porty patří PWM vstupy pro čtení informací z rádiového vysílače a PWM výstupy pro řízení otáček motorů.Se senzory, které jsou osazeny na desce komunikuje procesor prostřednictvím I^2C sběrnice.

Pro rozšíření budou na desce umístěny speciální konektory pro připojení až šesti sonarů, které umožní sledovat okolí quadrocoptery ve všech směrech. Další rozšíření je realizováno prostřednictvím tří sériových linek, kde první je určena pro připojení GPS modulu, druhá pro Raspberry PI a třetí pro telemetrii. Je samozřejmě možné všechny tři linky použít i k jiným účelům.

Dále je na desce implementována sběrnice SPI, která je využita pouze k nahrání bootloaderu platformy Arduino do řídící desky. Je jí dále pak možno využít pro případné další rozšíření.

Jelikož se na desce nachází obvody, které pracují s napětím 5V i 3,3V, bude mít deska dva zdroje. Zdroj s napětím 5V bude napájet CPU a obvody s PWM společně s převodníky logických úrovní. Napětí bude dále sníženo stabilizátorem na 3,3V, kterým se budou napájet senzory, převodníky logických úrovní a GPS modul.

2.3 Výběr obvodů

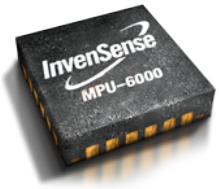
Jak už bylo řečeno výše, základem desky je mikrořadič ATmega2560, který umožní, krom řízení základních funkcí letu mnoho dalších možností týkajících se případných rozšíření. Tento mikrořadič byl vybrán na základě poměru ceny k výkonu a je jedním z mikrořadičů

podporovaných platformou Arduino. Existuje proto veliké množství knihoven, které je možné s tímto mikrořadičem použít.

2.3.1 Gyroskop a akcelerometr

Mezi základní senzory quadrocoptery patří gyroskop spolu s akcelerometrem, s jejichž pomocí dokáže řídící deska udržet zařízení jak ve stabilní poloze, tak dovede zajistit, že quadrocoptera zůstane ve stejné poloze i přes nevhodné okolní podmínky jako je například vítr.

Na trhuje k dispozici velké množství těchto senzorů. Senzory lze rozdělit do dvou skupin, v první skupině jsou senzory, které komunikují s procesorem po sběrnicích (například I^2C a SPI), kde je jíž hodnota výstupního signálu převedena na číslicový signál. A druhá skupina, kde výstupní signál je spojitý a je nutné jej zdigitalizovat (změřená hodnota je úměrná napětí na výstupu senzoru). . Ačkoli by byl odečet analogových hodnot diky přítomnosti A/D převodníků přímo na desce jednoduchý, není pro toto řešení ideální. Jelikož gyroskop a akcelerometr jsou tříosé senzory, bylo by potřeba šesti A/D převodníků, které je možné využít pro jiné aplikace. Z tohoto důvodu bylo vybráno řešení digitální. Nakonec byl vybrán obvod InvenSense MPU-6000 (obr.2.3), který navíc umožňuje snížení nároků na místo, protože se jedná o kombinované řešení gyroskopu a akcelerometru na jednom čipu.



Obrázek 2.3: MEMS obvod gyroskopu a akcelerometru MPU-6000 [13]

2.3.2 Barometr

Jedním ze senzorů, který umožní řízení quadrocoptery pomocí autopilota je barometr. Tento senzor se v našem případě spíše než k měření tlaku použije k měření výšky, ve které se zařízení nachází. Jelikož se nejedná o obvod, na který by se kladly, krom dobrého rozlišení, veliké nároky, byl vybrán jednoduchý obvod od firmy Freescale Semiconductor MPL3115A2 (obr.2.4). Tentoz obvod je schopen přímo poskytnout nadmořskou výšku, ve kterém se nachází s rozlišením kolem 10 cm.

Obrázek 2.4: MEMS obvod barometru MPL3115A2 [14]

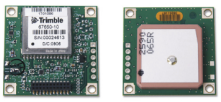
2.3.3 Magnetometr

Pro snadné rozpoznání směru, ve kterém je natočena quadrokoptéra je na desce implementován magnetometr. Magnetometr je elektronický kompas, s jehož pomocí určíme na kterou světovou stranu quadrocoptera směřuje. Na tento obvod není žádný speciální požadavek. Proto byl vybrán obvod co nejednoduší a nejlevnější. Obvod HMC5883L (obr.2.5) od firmy Honeywell MPS je pro tuto úlohu dokonalý a je zároveň velice používaný mezi internetovou komunitou.

Obrázek 2.5: MEMS obvod magnetometru HMC5883L [15]

2.3.4 GPS

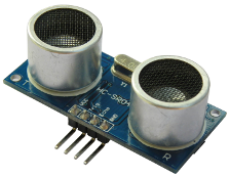
GPS je jedna z možných rozšíření, které deska umožňuje. Modul je zde, aby poskytl případnému autopilotovy představu o tom, kde se Quadrocoptera nachází. V případě, že by měl autopilot přístup k mapám, je možné pomocí dat z GPS vytvořit základní program, který bude schopen vyhýbat se budovám, nebo následovat předem vytyčenou trasu. Většina GPS modulů komunikuje pomocí rozhraní UART, na desce je proto pro modul připraven konektor. Byl zvolen modul od firmy Trimble (obr.2.6), který je již hotovým řešením a usnadní tak práci s modulem.



Obrázek 2.6: MEMS obvod gyroskopu a akcelerometru [16]

2.3.5 Sonar

Sonar je další rozšiřující modul, který deska podporuje. Tento modul je opět spíše pro autopilota a umožňuje mu detekovat blízké objekty. K sledování okolí je zapotřebí až šest samostatných ultrazvukových senzorů. Senzory snímají všechny směry, kterými se může quadrocoptera pohybovat a zaznamenávají vzdálenosti od objektů, které jsou v cestě. To umožňuje autopilotovy reagovat na náhlé a nečekané změny v terénu, nebo ho navigovat uvnitř budovy. Pro tuto činnost byly zvoleny velice jednoduché ultrazvukové moduly od firmy ElecFreaks HC-SR04 (obr.2.7). Modul funguje velice jednoduše, je však potřeba pro určení správné vzdálenosti počítat se změnou rychlosti šíření zvuku s měnící se teplotou.



Obrázek 2.7: Obrázek použitého sonaru[17]

2.3.6 Napájecí zdroje

Napájení desky je rozděleno mezi dva zdroje, 5V a 3,3V. V následující tabulce 2.4 je zaznamenána spotřeba jednotlivých součástek.

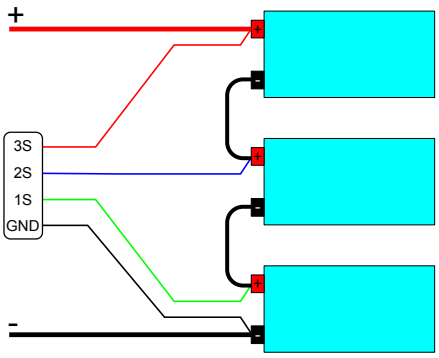
Součástka	**Proud**
ATmega2560	100 mA
MPU-6000	5 mA
MPL3115A2	10 μA
HMC5883L	100 μA
GPS	37 mA
HC-SR04	15 mA

Tabulka 2.4: Spotřeba jednotlivých obvodů

Jelikož se jedná o bateriově napájené zařízení byl zvolen jako hlavní zdroj 5V DC-DC měnič TPS62112, který má větší učinnost než obyčejný lineární stabilizátor s nízkým úbytkem v propustném směru. Následně je 5V sníženo na 3,3V pomocí lineárního regulátoru AMS1117. Maximální vstupní napětí DC-DC měniče je 17V a minimální 3,1V. Použitá baterie dodává 12,6V při maximálním nabití.

2.4 Baterie

Typická modelářská baterie se skládá z více menších článků zapojených v sérii. Jeden takový článek se označuje 1S a jeho napětí je 3,7V (4,2 V plně nabitý). Zároveň se jeden článek skládá z paralelního zapojení menších článků. Je tedy možné vidět označení 1S3P, což znamená jeden článek složený ze tří paralelně zapojených článků. Typické zapojení 3S baterie je vidět na obrázku 2.8.



Obrázek 2.8: Schéma zapojení pro měření kapacity baterie.

Byla zvolena baterie od firmy Pelikán Daniel FOXY G2 Li-Pol 5000mAh (obr. 2.9). Jedná se o baterii 3S1P, tedy tři články v sérii, s celkovým napětím 11,1 V (12,6 V plně nabité).

Obrázek 2.9: Fotografie baterie [18].

2.5 Rádiový vysílač a přijímač

V dnešní době je radiové řízení velice jednoduché a dostupné pro každého. V modelářském průmyslu existuje obrovské množství vysílačů a přijímačů za poměrně nízké ceny. Pro ovládání quadrocoptery je zapotřebí minimálně čtyř kanálů. Jeden kanál pro řízení rychlosti točení motorů, jeden pro rotaci, a další dva pro náklony do stran a vpřed, nebo vzad. Přiřazení kanálů pro ovládání quadrocoptery je znázorněno v kapitole 3 na obrázku 3.5. Aby však bylo možné pomocí vysílače předávat i jiné příkazy, byl zakoupen šesti kanálový přijímač Spectrum AR600 (obr. 2.10), který plně využívá možnosti vysilače Spectrum DX5e (obr. 2.10).

Obrázek 2.10: Vlevo vysílač [19], vpravo přijímač [20]

2.6 Elektromotory a regulátory otáček

Elektromotory dělíme do dvou skupin, tzv. brushed[21] a brushless[22], neboli stejnosměrné a střídavé elektromotory.

2.6.1 Stejnosměrné elektromotory

Stejnosměrný elektromotor je nejjednodušším typem elektromotoru využívající stejnosměrný proud. Stator je tvořen permanentním magnetem. Rotor je tvořen formou elektromagnetu s dvěma póly. Přepínač polarity elektrického proudu (tzv. komutátor) každou půlotáčku obrátí polaritu protékajícího proudu skrz elektromagnet. To umožní motoru neustále pokračovat ve směru rotace. Pokud by k tomuto přepnutí nedošlo, elektromotor by se zastavil. Regulace otáček je prováděna změnou napájecího napětí, nebo pomocí PWM modulace, kdy se záměrně mění střední hodnota tekoucího proudu. Směr otáčení je možné změnit pouhým přepólováním zdroje.

Obrázek 2.11: Stejnosměrný elektromotor

Stejnosměrné motory se zdají být ideální volbou, ale jen z hlediska jednoduchosti řízení. Jejich nízká efektivita se pro tuto aplikaci nehodí.

2.6.2 Střídavé elektromotory

Střídavé elektromotory patří mezi synchroní motory. To znamená, že elektrocké pole generované statorem a elektrické pole rotoru se shodují. Rotor je tvořen permanentními magnety a stator obvykle bývá složen ze tří vinutí zapojených do trojúhelníku, nebo do hvězdy.

Hlavní výhodou střídavých elektromotorů oproti stejnosměrným je absence mechanického komutátoru, který je zde nahrazen elektrickým komutátorem. Díky tomu mají střídavé elektromotory delší životnost, větší účinnost a neprodukují vysokofrekvenční rušení, které vzniká při jiskření mechanického komutátoru. Tato výhoda je zároveň i jejich nevýhodou. Stejnosměrné elektromotory jsou náročnější na výrobu a jsou proto mnohem dražší. Navíc je potřeba pro jejich funkci použít elektroniku, která zajistí komutaci jednotlivých vinutí. Rotující magnetické pole potřebné k roztočení elektromotoru je vytvářeno postupným připojováním jednotlivých vinutí ke zdroji stejnosměrného napětí.

Obrázek 2.12: Střídavý elektromotor NX-4006-530kv

Díky vlastnostem, kterými střídavé motory disponují byl zvolen motor NX-4006-530kv (obr. 2.12), který je přímo navržen pro použití u quadrocopter. Výrobce udává, jak se motor chová s různými velikostmi vrtulí (tab. 2.5) a jakou má spotřebu či efektivitu.

Vrtule	**Napětí [V]**	**Proud [A]**	**Tah [g]**	**Účinnost [g/W]**
GWS 9050	11,5	3	288	8,348
	15,3	4,7	490	6,814
GWS 1060	15,3	6,4	622	6,352
GWS 1047	11,5	5,2	475	7,943
	15,3	8	785	6,413
GWS Q1280	11,5	8,6	669	6,824
	15,3	13,3	1037	5,096

Tabulka 2.5: Spotřeba motorů v závislosti na velikosti vrtule a napětí

2.6.3 Regulátory otáček

Regulátory otáček jsou elektronická zařízení, která se používají k řízení otáček modelářských i jiných typů elektrických motorů. Stejně jako motory se i regulátory dělí na stejnosměrné a střídavé. Jedním s hlavních parametrů každého regulátoru je jeho proudová zatížitelnost. Jedná se o velikost trvalého proudu, který může regulátor dodávat, aniž by došlo k jeho poškození. Výrobce elektromotoru NX-4006-530kv doporučuje pro jeho řízení regulátor schopný dodávat maximální proud 20A, proto byl zvolen cenově dostupný RAY 20B (obr. 2.13).

Obrázek 2.13: Střídavý regulátor otáček ESC

Kapitola 3

Funkce a konstrukce

V této části je popsáno, jakým způsobem quadrocoptera funguje, jak se ovládá a z čeho se skládá.

3.1 Konstrukce quadrocoptery

Základem quadrocoptery jsou, jak již název napovídá, čryři motory zarovnané do čtverce. Toto rozložení nám dává pár zajímavých vlastností.

- Každý motor nese pouze jednu čtvrtinu celé váhy, což nám umožňuje použít méně výkonné motory
- Rotace jednoho páru motorů ruší rotaci druhého páru, který se otáčí v opačném směru. Tuto záležitost dále rozebereme v sekci 4.2

3.1.1 Konstrukce

Konstrukce quadrocoptery se skládá ze čtyř ramen sestavených do kříže. První verze konstrukce byla sestavena z hliníkových profilů a plexiskla. Snaha byla vytvořit co nejlehčí konstrukci, protože u létajících zařízení je váha velice důležitá z hlediska doby letu. Dle tabulky 2.5 bylo počítáno s tím, že jeden motor s vybranou vrtulí GWS 1047 má tah 475 g. Všechny čtyři motory by tedy měli unést až 1,9 kg s celkovou spotřebou 20,8 A. Zvolená baterie má kapacitu 5 Ah, což znamená, že by motory byli schopni nést 1,9 kg po dobu 14,5 min. Dle tabulky 3.1 určíme, jak těžká by měla konstrukce maximálně být, abychom byli ještě schopní nést alespoň 500g závaží.

Součástka	**Váha[g]**
Baterie	394
Elektromotor	67
Regulátor	26
Řídící deska	12

Tabulka 3.1: Váhy jednotlivých částí quadrocoptery bez konstrukce

Požadovanou váhu konstrukce spočteme dle vztahu

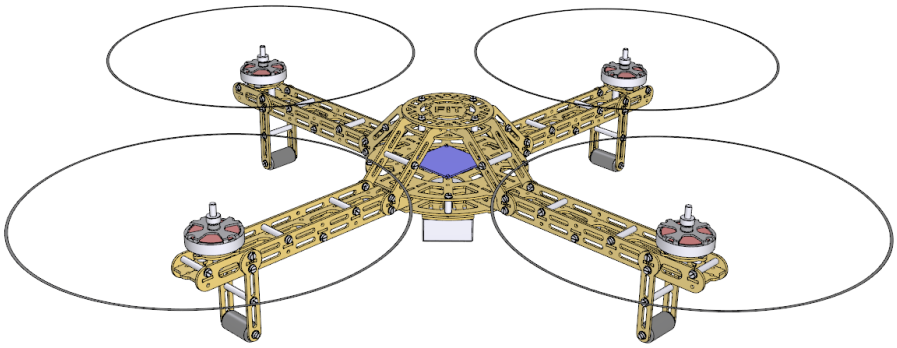
$$M = N - Z - B - 4 \cdot E - 4 \cdot R - D$$

kde M je váha konstrukce, N nosnost, Z požadovaná zátěž, B baterie, E elektromotor, R regulátor a D řídící deska. Odhadovaná váha konstrukce je tedy 622g. Jedná se však pouze o teoretický výpočet, stanovíme proto maximální váhu konstrukce na 400g. Pro dosažení takto nízké váhy je zapotřebí menší rozměr konstrukce. Podle jiných konstrukcí, které se dají na internetu najít, byla stanovena velikost na 50 cm od motoru k motoru v rámci jednoho ramene. Podařilo se sestavit konstrukci vážící 342g (obr. 3.1).

Obrázek 3.1: První verze rámu quadrocoptery s deskou Arduino Nano.

Nejčastějším problémem u létajících zařízení jsou vibrace. U quadrokoptery vznikají vibrace hlavně od nevyvážených vrtulí. Tyto vibrace se prostřednictvím konstrukce přenáší na řídící desku, čímž ovlivňují senzory. Detailněni je tato problematika popsána v kapitole 6 v sekci 6.1.

Druhou verzi konstrukce se během vytváření této práce nepodařilo dokončit. Její plány jsou však k práci přiloženy. Na obrázku 3.2 je zobrazen 3D model této verze, která by měla být lehčí a stabilnější než první verze. Hlavní předností této konstrukce by měla být nižší váha.

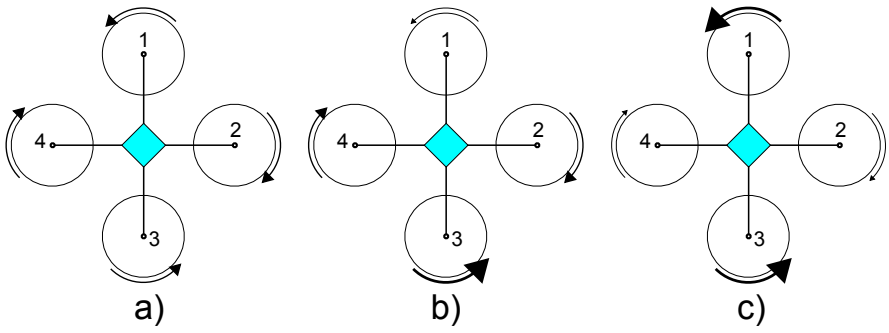


Obrázek 3.2: 3D model druhé verze kontrukce pro quadrocopteru.

3.2 Řízení quadrocoptery

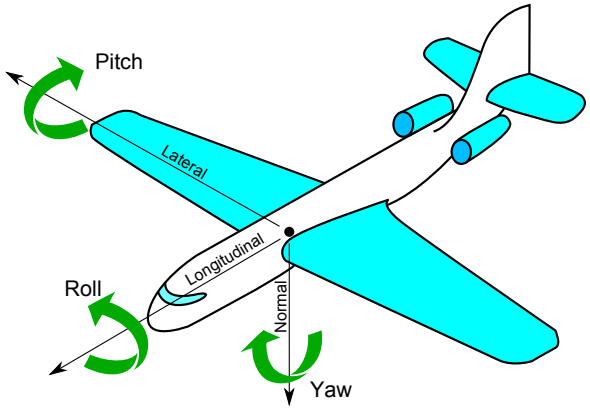
U běžného vrtulníku máme jednu velkou vrtuli, která dodává sílu potřebnou k vznesení a jednu malou, ocasní, která kompenzuje rotaci, kterou velká vrtule přenese na kontrukci vrtulníku. Bez ocasní vrtule by začal vrtulník rotovat až na rychlost točení vrtule.

Pokud by se u quadrocoptery točili všechny vrtule na stejnou stranu, tak bychom se dostali k stejnému problému, jako kdyby vrtulník neměl ocasní vrtuli. Celá konstrukce by začala rotovat ve směru otáčení vrtulí. Tohoto jevu se snadno zbavíme, pokud rozdělíme motory do dvou párů. Motory v jednom páru musí sdílet společnou osu i směr rotace. To znamená, že druhý pár musí sdílet druhou osu a opačný směr rotace než pár první. Oba rotační momenty se navzájem vyruší a quadrocoptera nebude rotovat. Schéma tohoto řešení je na obrázku 3.3 části a).



Obrázek 3.3: Řízení quadrocoptery a) Stabilní poloha; b) Náklon vpřed; c) Rotace

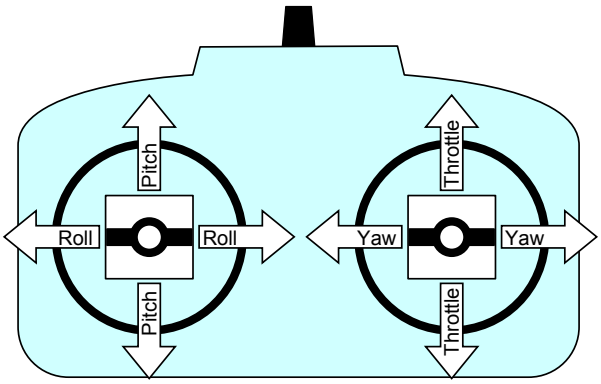
Nyní se již quadrocoptera ovládá pouhou změnou rychlosti otáčení jednotlivých motorů. V leteckém průmyslu máme takzvanou leteckou souřadnicovou soustavu. Jedná se o soustavu pevně spojenou s letadlem, jejíž počátek se nachází v referenčním bodě letadla (zpravidla v jeho těžišti) a jejíž osy (podélná, bočná a normálná) tvoří pravoúhlou a pravotočivou soustavu. V anglickém jazyce se jedná o osy Longitudinal, Lateral a Normal, nebo více známé Roll, Pitch a Yaw jež jsou znázorněny na obrázku 3.4.



Obrázek 3.4: Pozice všech tří os

Pokud bychom chtěly, aby se quadrocoptera otáčela v Normálové ose (stejné jako otáčet hlavou doprava a doleva) je zapotřebí zvýšit rychlost na motorech v jednom páru, ale zároveň snížit na páru druhém, jak je vidět na obrázku 3.3 části c), kde se zvyšuje rychlost otáčení na motoru číslo 1 a 3 a snižuje na motoru 2 a 4. V tomto případě se začne quadrocoptera otáčet doprava. Pokud bychom chtěli, aby se otáčela doleva, musíme zvýšit rychlost otáčení na motorech 2 a 4 a snížit na motorech 1 a 3. Snížení a zvýšení se musí provést o stejnou hodnotu, aby se zachoval stále stejný tah a quadrocoptera nezačala klesat, nebo stoupat.

Stejným způsobem se provádí rotace v ostatních osách. Tentokrát však zůstává stále stejná rychlost otáčení na jednom páru a na druhém se rychlost mění jak je vidět na obrázku 3.3 části b). Snížením rychlosti otáčení na motoru číslo 1 a zvýšením rychlosti otáčení na motoru číslo 3 o stejnou hodnotu dosáhneme náklonu v Bočné ose a quadrocoptera se posune směrem dopředu, dokud opět nesrovnáme rychlosti všech motoru. Důležité je opět zachovat stejnou rychlost otáčení na obou párech jinak by quadrocoptera klesala, nebo stoupala. Úplně stejným způsobem se postupuje, pokud chceme vytvořit pohyb do stran, jen místo motorů 1 a 3 měníme rychlost na motorech 2 a 4. Hlavní výhodou quadrocoptery, jak bylo řečeno výše, je její výborná manévrovatelnost díky možnosti pracovat ve všech třech osách v jednom čase. To umožňuje rychlé reakce na změnu směru.



Obrázek 3.5: Znázornění mapování os na vysílač.

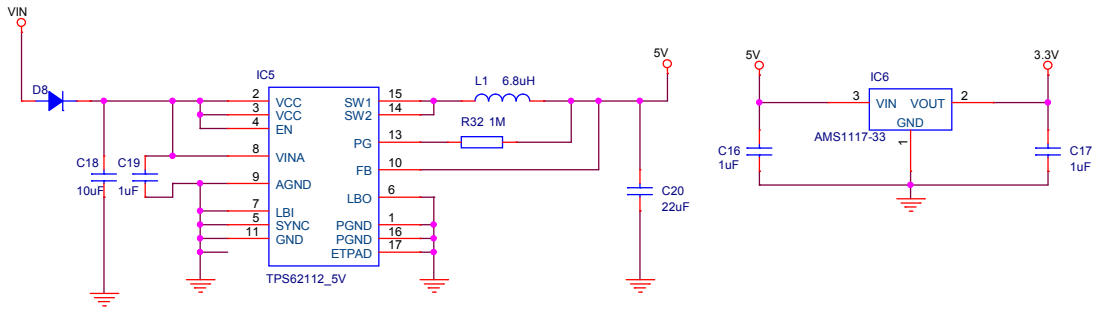
Možnosti jak ovládat quadrocopteru jsou celkem dvě, manuální a automatická stabilizace. U manuální stabilizace je vysílačka propojená přímo s motory a uživatel sám kompenzuje možné výchylky, které mohou při letu nastat. Tento způsob ovládání je však velice náročný z důvodu velké nepředvídatelností okolních jevů, jako je například vítr. Automatická stabilizace využívá k stabilnímu letu data z gyroskopu a akcelerometru a neustále udržuje quadrocopteru ve stabilní poloze čímž pomáhá vzdorovat okolním vlivům. Uživatel již jen udává, kterým směrem si přeje, aby quadrocoptera letěla, nebo tuto funkci může převezmout autopilot.

Kapitola 4

Realizace řídící desky

4.1 Napájecí zdroj

Deska obsahuje dva zdroje napájení, jejichž zapojení je na obrázku 4.1. Pro jejich realizaci byly použity obvody TPS62112, který mění vstupní napětí na 5V a AMS1117, který redukuje 5V na 3,3V. Oba obvody jsou zapojeny dle doporučení výrobce, nebylo zde třeba žádných dodatečných úprav.



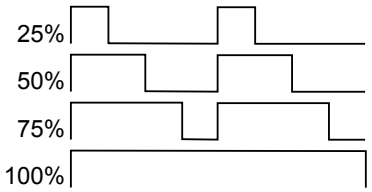
Obrázek 4.1: Napájecí zdroje desky, nalevo DC-DC měnič, napravo LDO regulátor

Schottkyho dioda D8 slouží jako ochrana proti přepólování, které je signalizováno LED diodou D1 ve schématu (B.1). Dioda D2 signalizuje připojené napájení, obě diody jsou však pouze signalizační a nemají na funkci žádný vliv.

DC-DC měnič TPS62112 byl zvolen na základě jeho výborné efektivnosti, která je u zařízení napájeného baterii velice důležitá z hlediska doby provozu. Měnič má zároveň veliký rozsah vstupního napětí 3V až 17V, což pokrývá modelářské baterie od 2S až po 4S, kde 1S je baterie napětí 3,7 V. LDO regulátor AMS1117 jsem zvolil pro jeho efektivnost v rámci malého rozdílu vstupního a výstupního napětí, která je opět důležitá pro provoz na baterii.

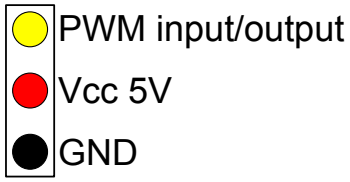
4.2 PWM vstup a výstup

Pulzně šířková modulace, ve zkratce PWM, je modulační technika, která se používá pro přenos analogového signálu pomocí signálu dvoustavového, neboli pomocí digitálního signálu. Ačkoliv je možné PWM modulaci použít pro kódování informací, jejím hlavním použitím je spínání výkonových zařízení, v našem případě elektromotorů. Množství přeneseného signálu se udává střídou, tento princip můžeme vidět na obrázku 4.2, kde například první průběh propustí 25% výkonu, důležitá je však i perioda střídy.



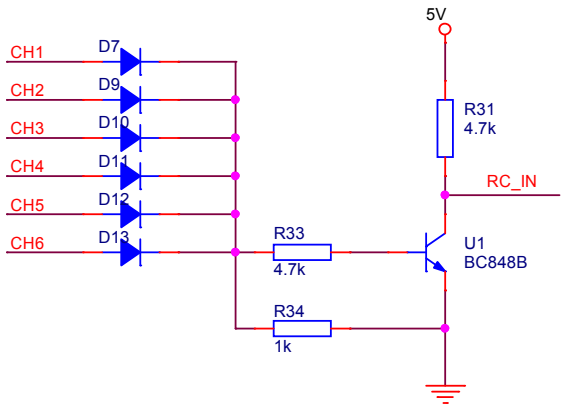
Obrázek 4.2: Popis dvou period PWM.

Doba trvání periody se mění s aplikací, ve které PWM modulaci používáme. Pro tlumení světla zářivky je zapotřebí perioda kolem 120 Hz, pro řízení motorů jednotky až desítky kHz a například pro audio desítky až stovky kHz. U modelářských regulátorů však stačí frekvence 50Hz jak je vidět na obrázku 4.5, kde perioda cyklu je 20ms. Veškeré modelářské vybavení s touto frekvencí pracuje, pokud se jedná o PWM. Veškeré modelářské vybavení dekóduje vstupní PWM tak, že pulz široký 1ms znamená nulová rychlost a 2ms znamená plná rychlost otáčení motoru. Některé regulátory jsou schopny si rozsah zapamatovat v případě, že vysílačka pracuje v jiném rozsahu. Aby se zabránilo možnému poškození zařízení je zapojení u všech modelářských zařízení realizováno třípinovým konektorem (obr. 4.3), kde uspořádání pinů zabraňuje nechtěnému otočení polarity.



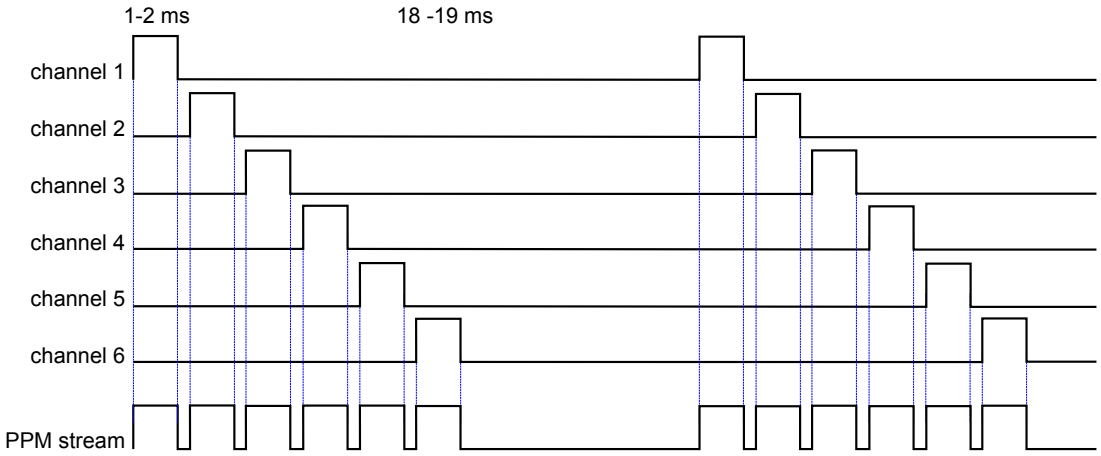
Obrázek 4.3: Popis PWM konektoru.

Zakoupená vysílačka má šest PWM kanálu, které je možné využit. Ideálním řešením je připojit každý kanál zvlášť na mikrořadič, ale bohužel již nezbyli volné piny s přerušením. Byl jsem proto nucen vytvořit jednoduchý obvod, který z šesti PWM kanálů vytvoří takzvaný PPM stream (obr. 4.5).



Obrázek 4.4: Schéma zapojení jednoduchého PPM encoderu.

Obvod na obrázku 4.4 je velice jednoduchý. Využívá toho, že mezi jednotlivými kanály je malý časový rozestup. To je zapříčiněno způsobem jakým přijímač reprodukuje PWM signál. Uvnitř přijímače je shift register, který postupně "vysune"jednotlivé kanály. Diody v obvodu PPM encoderu slouží k zamezení vytvoření zkratu mezi kanály. Transistor zde funguje jako spínač, který generuje posloupnost pulzů jednotlivých kanálů.



Obrázek 4.5: Popis funkce PPM encoderu.

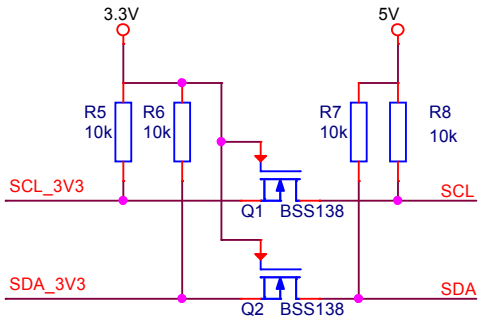
V tabulce 4.1 jsou vypsány přiřazení PWM výstupů na MCU. Pro vstup je použito na platformě Arduino přerušení číslo 0 na pinu 2, fyziky na mikrořadiči je použito přerušení číslo 4 na pinu 6.

PWM výstupní	MCU pin Arduino	MCU pin fyzicky
motor 1	6	15
motor 2	7	16
motor 3	8	17
motor 4	9	5

Tabulka 4.1: Přiřazení PWM výstupu na piny MCU.

4.3 Senzory a Raspberry PI

Na desce se nachází celkem tři hlavní senzory, gyroskop s akcelerometrem, barometr a magnetometr. Všechny tři používají pracovní napětí 3,3V. ATmega2560 používá 5V, je však na vstupu tolerantní vůči 3,3V. Bohužel I^2C je sběrnice, po které probíhá komunikace v obou směrech. Je proto potřeba použít obvod k přizpůsobení napěťových hladin logických úrovní (obr. 4.6).



Obrázek 4.6: Schéma zapojení obousměrného převodníku logických úrovní mezi 5V a 3,3V

Princip obvodu je docela jednoduchý. 3,3V označuje nižší napájecí napětí a 5V zase vyšší. SDA_3V3 je signál s nižší logikou a SDA zase vyšší. Pokud ani jedna strana nekomunikuje, pull-up rezistory způsobí, že je linka na obou stranách v logické jedničce. Tedy SDA_3V3 = 3,3V a SDA = 5V. Rozdíl napětí mezi vývody source a gate tranzistory je blízké nule a tranzistor je zavřen.

Jakmile levá strana (ta s nižším napětím) spojí linku se zemí (logická nula), rozdíl napětí mezi source a gate tranzistoru stoupne a tím se otevře. Logická nula se objeví i na pravé straně.

Pokud pravá strana spojí linku se zemí, dioda mezi source a drain tranzistoru způsobí, že se zvětší rozdíl napětí mezi source a gate a tranzistor se otevře. Tím se na levé straně objeví logická nula.

Na desce se vyskytuje i nutnost převádět jednosměrnou komunikaci. Konkrétně s Raspberry PI a GPS. Zde již díky možnosti mikrokontroleru rozpoznat na vstupu napětí od 3V do 5V jako logickou 1, není třeba použít tento převodník, ale pouze odporový dělič.

Tabulka 4.2 obsahuje přiřazení důležitých pinů připojených na MCU.

Název pinu	**MCU pin Arduino**	**MCU pin fyzicky**
I^2C SCL/SDA	21/20	43/44
Gyroskop a akcelerometr přerušení	3	7
Barometr přerušení	18 int 5	46 int 3
Magnetometr přerušení	19 int 4	45 int 2
GPS RX/TX	Serial 3	63/64
GPS Mode	29	71
GPS Antena	28	72
Sonar Trigger	37	53
Sonar Echo 1	ADC8	89
Sonar Echo 2	ADC9	88
Sonar Echo 3	ADC10	87
Sonar Echo 4	ADC11	86
Sonar Echo 5	ADC12	85
Sonar Echo 6	ADC13	84

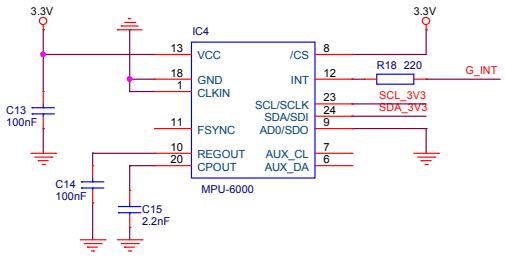
Tabulka 4.2: Přiřazení pinu senzorů na MCU.

4.3.1 Gyroskop a akcelerometr

Jako senzor gyroskopu a akcelerometru je použit obvod MPU-6000. Tento obvod je na trhu jediný, který v sobe kombinuje funkci obou senzorů. Obvod má vestavěnou takzvanou DMP (Digital Motion Processor) jednotku, která již provádí některé matematické úkony, čímž odebírá výpočetní zátěž z mikrořadiče. Tato jednotka ukládá nově vypočítané hodnoty do 1024 B FIFO paměti a signalizuje jejich vložení pomocí signálu přerušení. Ze senzoru je možné vyčítat i hrubé hodnoty a výpočty provádět v mikrořadiči. Typická spotřeba senzoru je pouhých 3,6mA, kdy po uspání obvodu klesne až na 10μA.

Senzor má proměnlivý rozsah citlivosti ±250, ±500, ±1000 a ±2000°/s pro gyroskop a ±2g, ±4g, ±8g a ±16g pro akcelerometr. Má také vestavěnou dolní propust pro filtraci dat a korekci závislou na teplotě, takže uživatel nemusí tyto problémy řešit. Senzor obsahuje tři 16-bitové AD převodníky pro gyroskop a tři 16-bitové AD převodníky pro akcelerometr pro získání co nejpřesnějších hodnot.

Schéma zapojení obvodu se shoduje s doporučením výrobce (obr. 4.7).



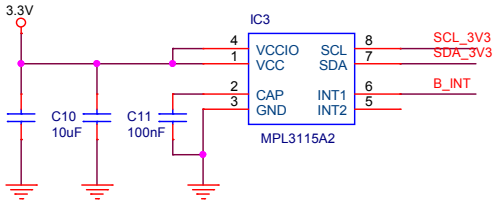
Obrázek 4.7: Schéma zapojení MPU-6000.

Senzor dokáže komunikovat po sběrnicích SPI a I²C. V tomto případě se komunikace provádí po sběrnici I²C signály SDA_3V3 a SCL_3V3. Signál G_INT slouží pro indikaci výpočtu nových hodnot DMP jednotkou.

4.3.2 Barometr a Magnetometr

Obvod MPL3115A2 komunikuje s mikrořadičem po I²C sběrnici na signálech SDA_3V3 a SCL_3V3 a poskytuje data o tlaku a nadmořské výšce, ve které se nachází. Senzor obsahuje 24-bitový AD převodník pro vysoké rozlišení. Barometr přepočítává některé hodnoty a tím odebírá zátěž z mikrořadiče. Rozsah měření tlaku senzoru se pohybuje od 50kPa do 110kPa, což odpovídá až 5 km nad mořem, kde tlak u hladiny moře je 101kPa. Typická spotřeba se pohybuje kolem 40μA na stabilní rozlišení nadmořské výšky 10cm. Senzor má již vestavěný teploměr pro korekci dat, teplotu je možné si samostatně vyčítat.

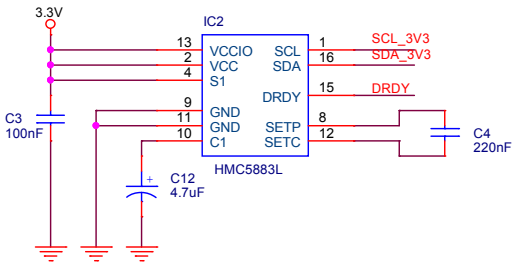
Schéma zapojení obvodu se shoduje s doporučením výrobce (obr. 4.8).



Obrázek 4.8: Schéma zapojení barumetru.

Magnetometr HMC5883L pro měření magnetického pole. Obvod obsahuje 12-bitový AD převodník, který umožňuje detekci polohy v magnetickém poli Země s přesností na 1,5°. Typická spotřeba se pohybuje okolo 100μA. Senzor stejně jako ostatní senzory komunikuje po I²C sběrnici přes signály SDA_3V3 a SCL_3V3.

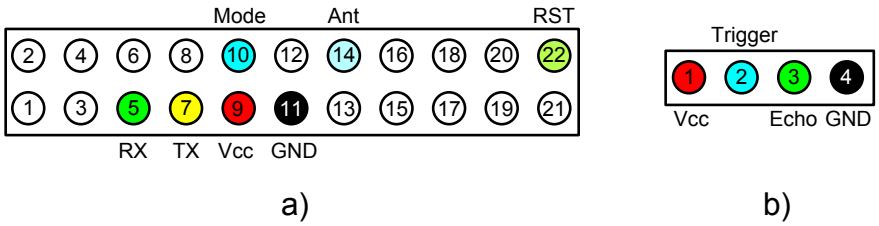
Schéma zapojení obvodu se shoduje s doporučením výrobce (obr. 4.9).



Obrázek 4.9: Schéma zapojení magnetometru.

4.3.3 GPS, Sonar a Raspberry PI

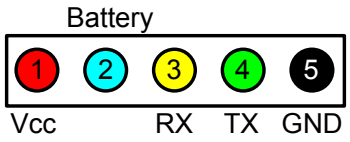
GPS a sonar jsou externí moduly, na které je pouze připraven konektor na řídící desce (obr. 4.10). GPS komunikuje pomocí UART v LVTTL logice. Je proto potřeba provést konverzi, ale pouze na RX pinu na straně GPS. Další vstupy, které je třeba ošetřit převodníkem jsou RESET a MODE, který umožňuje přepínat GPS do režimu STANDBY. Pro převod není potřeba použít žádný složitý převodník, v tomto případě dostačuje použití odporového děliče. GPS modul dále disponuje výstupem, který je nastaven do logické 0 pokud je anténa odpojena, nebo zkratována a pokud je připojena tak se nastaví do logické 1.



Obrázek 4.10: Obrázek a) konektor pro připojení GPS, obrázek b) konektor pro připojení sonaru.

Sonar jako jediný senzor pracuje při napětí 5V, není zde proto potřeba žádná konverze. Tento modul v sobě nemá vestavěný teploměr, který by prováděl korekce změny rychlosti zvuku při změnách teploty. Tuto činnost si musí uživatel ohlídat sám. Sonar má dva komunikační piny, ECHO a TRIGER. Pomocí pinu TRIGGER dostane od mikrořadiče sonar povel k zahájení měření, který vyšle 8 krátkých ultrazvukových pulzů. Pomocí těchto pulzů určí vzdálenost objektu a pošle na pin ECHO pulz s danou šířkou, která přestavuje dobu, kterou zvuku trvalo dorazit k vzdálenému objektu a zpět.

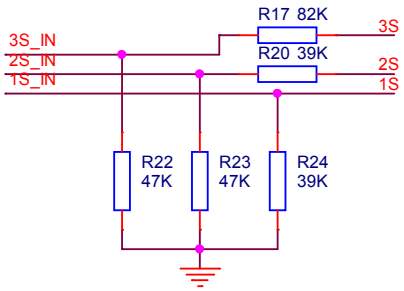
Pro komunikaci s Raspberry PI slouží připravený UART konektor, jehož popis je na obrázku 4.11. Řídící deska pomocí tohoto konektoru dodává Raspberry napájení 5V. Signál battery je překlopí do logické 1 pokud kapacita baterie klesna pod 25%.



Obrázek 4.11: Schéma zapojení konektorů pro připojení Raspberry PI.

4.3.4 Měření baterie

K měření kapacity baterie jsou použity AD převodníky na mikrořadiči. Pomocí odporových děličů (obr. 4.12) je však potřeba upravit vstupní napětí tak, aby na jednotlivém AD převodníku nepřesáhlo 5V, jinak by mohlo dojít k jejich poškození.



Obrázek 4.12: Schéma zapojení měření kapacity baterie.

Tabulka 4.3 obsahuje přiřazení důležitých pinů připojených na MCU.

Název pinu	**MCU pin Arduino**	**MCU pin fyzicky**
1S_IN	ADC0	97
2S_IN	ADC1	96
3S_IN	ADC2	95

Tabulka 4.3: Přiřazení měření baterie na piny MCU.

Kapitola 5

Software

Veškeré použité kódy jsou k dispozici na přiloženém CD. Knihovny využívají funkce platformy Arduino, pro použití bez této platformy je třeba tyto funkce nahradit!

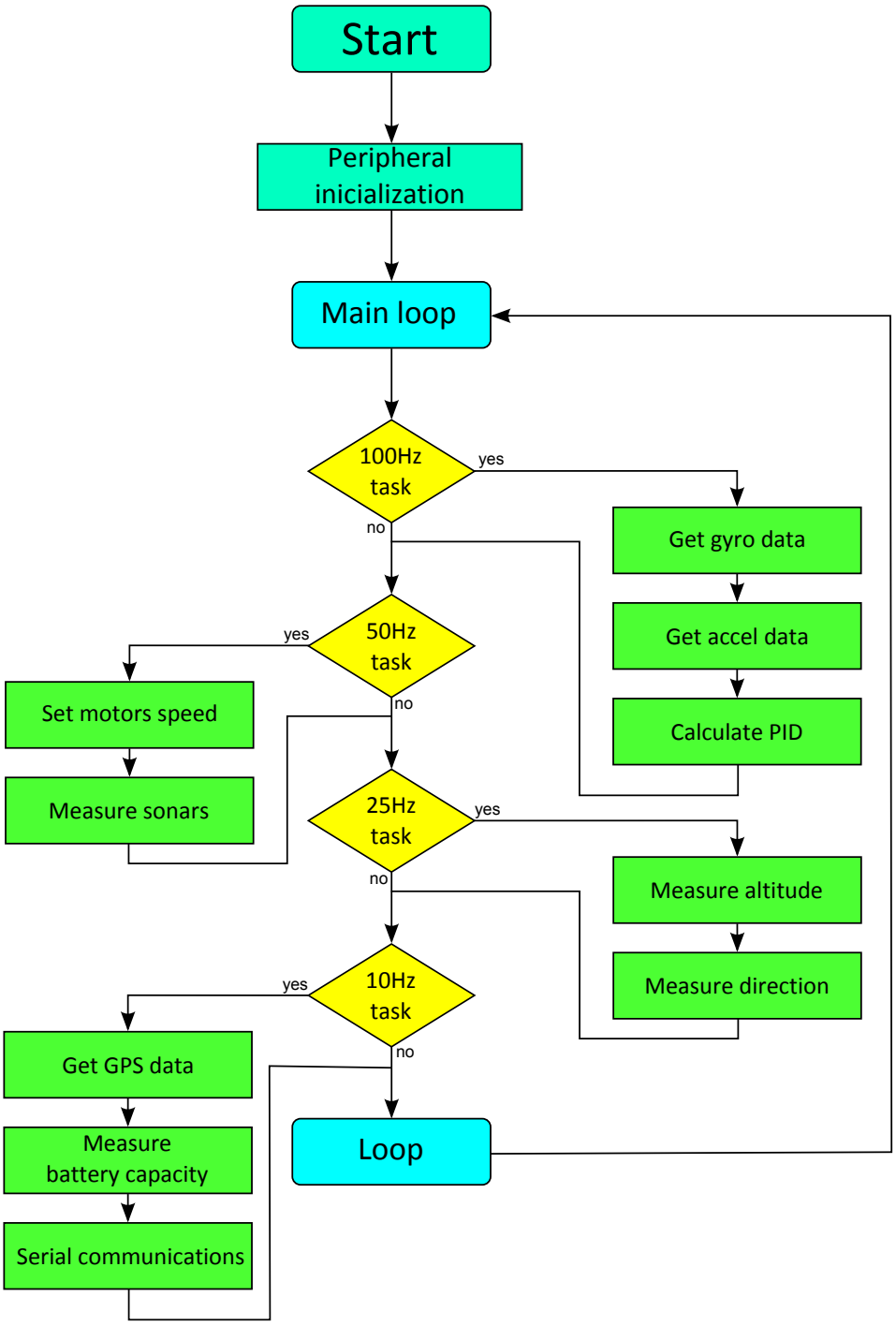
5.1 Jádro

Na obrázku 5.1 je vyobrazen vývojoví diagram softwaru pro quadrocopteru. Při startu programu se spustí inicializace jednotlivých periferii, mezi něž patří gyroskop, akcelerometr, barometr, magnetometr, GPS, sonar a nastaví se PID regulátory. Následně se spustí hlavní smyčka programu, která je rozdělena do čtyř sekcí.

Každá sekce pracuje na jiné frekvenci (100Hz, 50Hz, 25Hz a 10Hz), čímž je určena její priorita v celém programu. Části kódu spuštěné s frekvencí 100Hz mají největší prioritu a patří mezi ně gyroskop, akcelerometr a PID regulace. Jelikož regulátory otáček pro řízení elektromotorů pracují na frekvenci 50Hz nemá proto smysl provádět obnovu dat častěji. Nastavení otáček elektromotoru se proto provádí v sekci s frekvencí 50Hz. Měření detekce vzdálených objektů je důležitá činnost, ale častější obnova nám nepřinese lepší výsledek. Sonar má totiž pracovní frekvenci 40Hz.

Ostatní operace již nejsou tak důležité, jsou proto zařazeny do nižších priorit. Komunikace s přídavnými moduly po UART jsou zařazeny do nejnižší priority a vykonávají se frekvencí 10Hz. Ačkoliv by stačilo obnovovat měření kapacity baterie i s frekvenci 1Hz, tak není potřeba pro jednu operaci zakládat novou sekci. Proto je i měření baterie v sekci 10Hz.

Během jakýchkoliv dodatečných úprav kódu je možno libovolně do sekcí přidávat nové operace, či přídávat nové sekce. Všechny bloky v jednotlivých sekcích se dají před překladem jednoduše vypnout pomocí konfiguračního souboru config.h.



Obrázek 5.1: Vývojový diagram softwaru pro quadrocopteru.

5.2 Bloky

5.2.1 100Hz

5.2.1.1 Gyroskop a akcelerometr

MEMS obvod MPU-6000 je velice populární a také hojně používaný. Existuje proto velice rozsáhlá knihovna, kterou zpracoval Jeff Rowberg ([24]). Knihovna umožňuje vyčítat jak raw data ze senzoru, tak využívat vestavěný DMP procesor a zpřístupňuje další vlastnosti senzoru. V následujícím kódu je zobrazena ukázka jakým způsobem lze s pomocí knihovny vyčítat ze senzoru data.

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t gx, gy, gz;

void setup() {
    // inicializace komunikace po I2C sběrnici
    Wire.begin();
    // inicializace komunikace UART
    Serial.begin(38400);

    // Spustění incializace senzoru
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();

    // ověření spojení
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6000 connection successful" : "MPU6000
        connection failed");
}

void loop() {
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  // v proměnných ax,ay,az a gx,gy,gz
        jsou načteny aktuální raw hodnoty ze senzoru
}
```

Jak je vidět, práce s knihovnou je jednoduchá. Následující kód znázorňuje použití raw hodnot k výpočtu náklonu zařízení za použití Kalmanova filtru ([25]). Kalmanův filtr je speciální algoritmus pro filtraci signálů v časové oblasti. Výhodou tohoto algoritmu je schopnost získat čistý signál a hodnoty ze zašuměného signálu nebo jinak znehodnoceného souboru hodnot, i bez jakéhokoliv poznatku o rušení.

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

MPU6050 accelgyro;

Kalman kalmanX;
Kalman kalmanY;

int16_t accX, accY, accZ;
int16_t gyroX, gyroY, gyroZ;

double gyroXangle, gyroYangle; // vypočet uhlu náklonu pouze pomocí senzoru
double kalAngleX, kalAngleY; // konečné úhly po aplikaci filtru

double Pitch_out, Roll_out;
uint32_t timer;

void setup() {
    // inicializace komunikace po I2C sběrnici
    Wire.begin();

    // Spustění íncializace senzoru
    accelgyro.initialize();

    // ověření spojení
    (accelgyro.testConnection();

    accelgyro.getMotion6(&accX, &accY, &accZ, &gyroX, &gyroY, &gyroZ);

        double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
    double pitch = atan2(-accX, accZ) * RAD_TO_DEG;

    kalmanX.setAngle(roll); // nastaví počáteční úhel
    kalmanY.setAngle(pitch);
    gyroXangle = roll;
    gyroYangle = pitch;
}

void loop() {
    accelgyro.getMotion6(&accX, &accY, &accZ, &gyroX, &gyroY, &gyroZ);

    double dt = (double)(micros() - timer) / 1000000; // Calculate delta time
             timer = micros();

             double roll  = atan2(accY, accZ) * RAD_TO_DEG;
    double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;

    double gyroXrate = gyroX / 131.0; // Convert to deg/s
    double gyroYrate = gyroY / 131.0; // Convert to deg/s

    // korekce přechodu mezi -180 a 180 stupni
    if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 && kalAngleY < -90)) {
       kalmanY.setAngle(pitch);
       kalAngleY = pitch;
       gyroYangle = pitch;
    } else {
       kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt); // výpočet úhlu pomocí Kalmanova
            algoritmu
    }

    if (abs(kalAngleY) > 90) {
       gyroXrate = -gyroXrate;
    }
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // výpočet úhlu pomocí Kalmanova
         algoritmu

    gyroXangle += gyroXrate * dt; // výpočet úhlu bez Kalmanova algoritmu
    gyroYangle += gyroYrate * dt;

    if (gyroXangle < -180 || gyroXangle > 180) {
       gyroXangle = kalAngleX;
    }
    if (gyroYangle < -180 || gyroYangle > 180) {
       gyroYangle = kalAngleY;
    }

    Pitch_out = -1*kalAngleX;
    Roll_out = kalAngleY;
}
```

Protože práce s DMP procesorem způsobovala problémy s časováním bylo nakonec použito toto řešení. Detailněji jsou tyto problémy popsány v kapitole 6

5.2.1.2 PID regulátor

PID regulátor (obr. 5.2) patří mezi spojité regulátory, který se skládá z Proporcionální, Integrační a Derivační části. V systémech řízení se řadí před řízenou soustavu. Do regulátoru vstupuje regulační odchylka e(t) a vystupuje akční veličina u(t), která je definována jako

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$

kde K_p (Proporcionální zisk), K_i (Integrační zisk), K_d (Derivační zisk), e (Error = Setpoint - Value), t (čas), τ (integrační konstanta, 0 - t).

Proporcionální složka, P regulátor, je prostý zesilovač. Regulační odchylka je přímo úměrná akční veličině. Proporcionální odezva se může nastavit pomocí konstanty K_p.

$$P = K_p \cdot e(t)$$

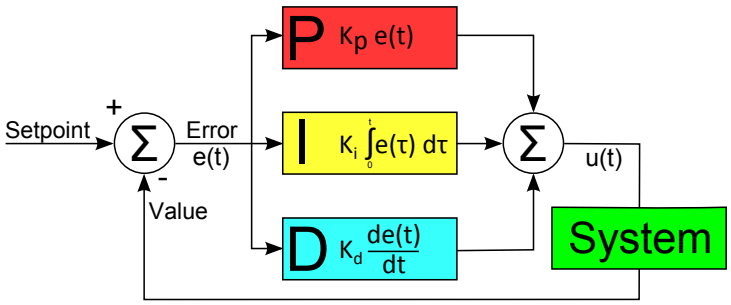
Vysoký proporcionální zisk má za následek rychlou změnu v procesu. Pokud je ale zisk příliš veliký dochází k nestabilním situacím. Pokud je zisk naopak příliš malý, dochází k znecitlivění systému.

Integrační složka, I regulátor, je součtem okamžitých chyb v čase t, který nasčítá chybu, která se měla vykompenzovat již v minulém kroku a tento součet vynásobí konstantou K_i, pro větší či menší užitek. Integrační složka urychluje proces řízení a odstraňuje zbytkové chyby, které se nepovede odstranit v proporcionální složce. Je však důležité si dát pozor, aby naakumulovaná chyba příliš nepřesáhla požadované hodnoty.

$$I = K_i \cdot \int_0^t e(\tau)d\tau$$

Derivační složka, D regulátor, se používá pro zrychlení regulačního děje. Její nevýhodou je, že zesiluje šum, což může v některých případech vést až k nepoužitelnosti celého systému. Samostatně se D-regulátor nikdy nevyskytuje.

$$D = K_d \cdot \frac{de(t)}{dt}$$



Obrázek 5.2: Blokové schéma funkce PID regulátoru.

U quadrocoptery Setpoint nastavuje uživatel pomocí radiového vysílače. Nastavuje tak úhel, který si přeje. Systémem je zde gyroskop, který udává aktuální náklon quadrocoptery. Pro stabilizaci quadrocoptery jsou zapotřebí tři PID regulátory, každý na jednu osu (Yaw, Pitch a Roll). Výstupem jednotlivých regulátorů je tedy offset, který je potřeba přidat, nebo ubrat z daných elektromotorů. Výsledná rychlost pro jednotlivé motory vypadá následovně

motor 1 = Throttle - PitchOffset + YawOffset
motor 2 = Throttle + RollOffset - YawOffset
motor 3 = Throttle + PitchOffset + YawOffset
motor 4 = Throttle - RollOffset - YawOffset

kde Throttle je požadovaný výkon a PitchOffset, RollOffset a YawOffset jsou výstupy PID regulátorů. Tyto výpočty je však možné použít jen v případě, že máme dokonale vyváženou konstrukci a máme jistotu, že všechny motory se budou při stejném nastavení točit stejnou rychlostí. V praxi je potřeba vytvořit korekci takového rozdílu. Korekci provedeme použitím druhého PID regulátoru v jedné ose, který bude kompenzovat možné rozdíly ve výkonu motorů. Výpočet se tedy změní na

motor 1 = Throttle - PitchOffset + YawOffset - PitchCorrection
motor 2 = Throttle + RollOffset - YawOffset + RollCorrection
motor 3 = Throttle + PitchOffset + YawOffset
motor 4 = Throttle - RollOffset - YawOffset

kde PitchCorrection a RollCorrection jsou výstupy korekčních regulátorů.

Knihovna pro realizaci PID regulátoru byla vytvořena svépomocí. Knihovna je velice jednoduchá a obsahuje následující funkce

- void Tune(double Kp, double Ki, double Kd); //nastaví nové hodnoty konstant K_p, K_i a K_d
- void SetOutput(double MinOut, double MaxOut); // nastaví minimlní a maximální hodnotu,kterou může nabívat výstup
- double Compute(double Error); // vypočté výstup PID regulátoru u(t)
- double getP(); // vrátí aktuální hodnotu K_p
- double getI(); // vrátí aktuální hodnotu K_i
- double getD(); // vrátí aktuální hodnotu K_d
- void reset(); // reset integrační složky regulátoru

Základní kód pro výpočet výstupu PID regulátoru je uveden v následujícím kódu

```
double PID::Compute(double Input) {
   unsigned long runTime = millis();
   double dt = (double)(runTime - lastTime);
   double P = (double) kp * Input;
   double D = (double) (kd * (Input - error) * 1000.0/dt);
   error = Input;
   double I = (double) (lastI + ki * Input * dt /1000.0);
   double Output = (double) (P + I + D);
   lastI = I;
   lastTime = runTime;

   if(Output > maxOut){
     Output = maxOut;
```

```
    }
    else if (Output < minOut){
      Output = minOut;
    }
    return Output;
}
```

5.2.2 50Hz

Pro nastavení rychlosti otáčení motorů slouží PWM modulace. Pro generování PWM modulace je nejlepší řešení použít čítače a jejich přerušení. Platforma Arduino již obsahuje knihovnu, která tímto způsobem PWM signál o frekvenci 50Hz generuje. Jedná se o knihovnu Servo.h Následující kód ukazuje, jakým způsobem se knihovna používá

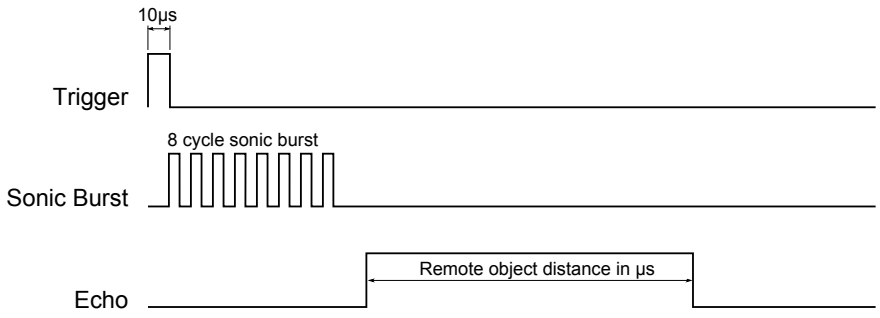
```
#include <Servo.h>

Servo esc1;  // vytvoření instance Servo
Servo esc2;
Servo esc3;
Servo esc4;

void setup()
{
  esc1.attach(6);  // přiřazení pinu pro generování PWM
  esc2.attach(7);
  esc3.attach(8);
  esc4.attach(9);
}

void loop() {
  esc1.writeMicroseconds(1500);  // generování PWM signálu s šířkou pulzu 1,5ms
  esc2.writeMicroseconds(1500);
  esc3.writeMicroseconds(1500);
  esc4.writeMicroseconds(1500);
}
```

Sonar se používá pro měření vzdálenosti mezi senzorem a vzdáleným objektem. Vybraný sonar provádí měření vysláním osmi krátkých 40 kHz pulzů a detekuje příchozí pulzy. Výsledný čas zobrazí na pinu ECHO jako pulz o dané šířce (obr. 5.3). K práci se sonarem byla vytvořena knihovnu, která se snadno používá a dovoluje získávat data z jednotlivých sonarů.



Obrázek 5.3: Průběhy měření vzdálenosti na sonaru HC-SR04.

Knihovna se skládá z následujících funkcí

- double getDistance(byte Sonar); // vrátí vzdálenost změřenou Sonarem 1 až 6, dle volby uživatele

- int getTime(byte Sonar); // vrátí čas změřený Sonarem 1 až 6, dle volby uživatele

Následující kód ukazuje jakým způsobem je možné získat data z předního sonaru

```
#include <Wire.h>
#include "sonar.h"

void setup()
{
    Serial.begin(9600);  // nastaví rychlost komunikace UART
}

void loop()
{
    double sonar = getDistance(1);
      Serial.print("Překážka (cm):");
    Serial.print(sonar);

    Serial.println();
}
```

5.2.3 25Hz

5.2.3.1 Barometr

Pro vyčítání dat z barometru byla vytvořena knihovna, která se snadno používá. Senzor již poskytuje měřená data ve finální podobě, není proto potřeba žádná další úprava. Knihovna obsahuje následující funkce

- void begin(); // inicializace senzoru
- double getAltitude(); // vrátí nadmořskou výšku v metrech
- double getPressure(); // vrátí barometricky tlak v pascalech
- double getTemp(); // vrátí teplotu ve stupních Celsia
- void setMode(byte mode); // nastaví mode, 1 - Barometr, 2 - Altimetr, 3 - Standby, 4 - Active
- void setOversample(byte Oversample); // nastaví počet vzorků na $2^{Oversample}$, vstupní hodnoty 0 až 7
- void enableFlags(); // nastaví důležité příznaky, důležité při inicializaci

Následující kód ukazuje jak vyčítat aktuální výšku v metrech, ve které se senzor nachází

```
#include <Wire.h>
#include "MPL3115A2.h"

// vytvoří instanci senzoru
MPL3115A2 barometer;
float zeroLevel = 0;

void setup()
{
    Wire.begin();         // připojí I$^{2}$C
    Serial.begin(9600);  // nastaví rychlost komunikace UART

    barometer.begin(); // spustí senzor

    barometer.setMode(2); // přepne barometr pro měření nadmořské výšky

    barometer.setOversample(7); // nataví oversample na doporučenou hodnotu výrobcem
    barometer.enableFlags(); //nastavý důležité příznaky
    zeroLevel = barometer.readAltitude();    // zjisti počáteční nadmořskou výšku, v této
        chvíli se nesmí se senzorem pohybovat
}

void loop()
{
    float altitude = barometer.readAltitude();    //  vyčte aktuální nadmořskou výšku.
      Serial.print("Aktuální výška (m):");
```

```
    Serial.print(altitude - zeroLevel);

    Serial.println();
}
```

5.2.3.2 Magnetometr

Knihovna pro použití magnetometru obsahuje pouze dvě funkce

- void begin(); // inicializace senzoru
- double getNorthOffset(); // vrátí úhel, který quadrocoptera svírá s severním pólem Země

Funkce getNorthOffest udává úhel mezi směrovým vektorem quadrocoptery a magnetickým pólem Země. To umožňuje autopilotovy určit jakým směrem je quadrocoptera natočená a následně reagovat. Následující kód ukazuje, jak se knihovna používá

```
#include <Wire.h>
#include "HMC5883L.h"

// vytvoří instanci senzoru
HMC5883L magnetometer;
double northOffset = 0;

void setup()
{
    Wire.begin();         // připojí I$^{2}$C
    Serial.begin(9600);   // nastaví rychlost komunikace UART

    magnetometer.begin(); // spustí senzor
}

void loop()
{
    northOffset = magnetometer.getNorthOffset();    //  vyčte aktuální nadmořskou výšku.
    Serial.print(northOffset);

    Serial.println();
}
```

5.2.4 10Hz

5.2.4.1 GPS a UART

Komunikace pomocí UART je na platformě Arduino velice jednoduch8. Ukázka odesílání a přijímání dat je v následujícím kódu

```
void setup() {
    Serial.begin(9600);  // připravý komunkaci na UART0 s rychlostí 9600 baud
    Serial1.begin(9600); // připravý komunkaci na UART1 s rychlostí 9600 baud
}

void loop() {
    // přečte data z UART0 a přepošle je na UART1
    if (Serial.available()) {
        int inByte = Serial.read();
        Serial1.print(inByte, BYTE);
    }

    // přečte data z UART1 a přepošle je na UART0
    if (Serial1.available()) {
        int inByte = Serial1.read();
        Serial.print(inByte, BYTE);
    }
}
```

Vybraný GPS modul pro komunikaci využívá protokol NMEA 0183 ([26]), který odesílá po UART (9600 baud) tzv. věty. Každá věta začíná znakem $, za kterým následuje kód věty. U modulu je možné si zvolit, jaké věty má posílat. V továrním nastavení posílá modul věty $GPGGA, $GPGSA, $GPGSV a $GPRMC (tab. 5.1), které jsou pro naše potřeby dostačující.

Kód věty	**Funkce věty**
$GPGGA	Věta obsahuje informace o aktuální globální poloze
$GPGSA	Věta obsahuje obsahuje informace o počtu viditelných družic a jejich geometrickém rozmístění
$GPGSV	Věta obsahuje informace o počtu viditelných družic z aktuální polohy a dále informace o každé viditelné družici
$GPRMC	Věta obsahuje základní informace použité při navigaci

Tabulka 5.1: Stručný popis používaných vět protokolu NMEA 0183.

Věta $GPGGA má následující formát

$GSGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh

kde

- hhmmss.ss = UTC dané pozice (např. 170834, neboli 17:08:34)
- llll.ll = zeměpisná šířka dané pozice(např. 4124.8963, neboli 41d 24.8963')
- a = N nebo S
- yyyyy.yy = zeměpisná délka dané pozice (např. 08151.6838, neboli 81d 51.6838')
- a = E nebo W
- x = indikátor kvality (0 = no fix, 1 = GPS fix, 2 = Dif. GPS fix)
- xx =počet využívaných satelitů
- x.x = relativní přesnost horizontální pozice
- x.x = nadmořská výška pozice antény
- M = jednotka nadmořské výšky, metry
- x.x = Geoidal separation
- M = jednotky geoidal separation, metry
- x.x = stáří GPS dat (sekundy)
- xxxx = Differential reference station ID
- *hh = checksum

Věta $GPGSA má následující formát

$GPGSA,Mode1,Mode2,id1,id2,id3,id4,id5,id6,id7,id8,id9,id10,id11,id12,PDOP,HDOP,VDOP*hh

kde

- Mode1 = M (manual), A (automatic)
- Mode2 = [1 = fix; 2 = 2D, 3 = 3D]
- id 1 - 12 = ID satelitů použitých při zaměření polohy(null pro nevyužité místo)
- PDOP = přesnost v 3D pozici
- HDOP = přesnost v 2D pozici
- VDOP = přesnost v 1D pozici
- *hh = checksum

Věta $GPGSV má následující formát

$GPGSV,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19*hh

kde

- 1 = celkový počet zpráv tohoto typu v jednom cyklu
- 2 = číslo zprávy
- 3 = počet viditelných satelitu
- 4 = N nebo S prvního satelitu
- 5 = nadmořská výška ve stupních, maxumim 90
- 6 = Azimut
- 7 = SNR, 00-99 dB
- 8-11 = Informace o druhém satelitu, stejné jako 4-7
- 12-15 = Informace o třetím satelitu, stejné jako 4-7
- 16-19 = Informace o čtvrtém satelitu, stejné jako 4-7
- *hh = checksum

Věta $GPRMC má následující formát

$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh

kde

- hhmmss.ss = časové razítko (např. 225446, neboli 22:54:46 UTC)
- A = validita dat (A = OK, V = warning)
- llll.ll = zeměpisná šířka (např. 5133.82, neboli 51d 33,82)
- a = N nebo S
- yyyy.yy = zeměpisná délka (např. 00042.24, neboli 000d42.24')
- a = E nebo W
- x.x = rychlost v uzlech
- x.x = směr
- ddmmyy = datum
- x.x = variace
- a = E nebo W
- *hh = checksum

5.2.4.2 Měření baterie

Baterie je důležitou součástí quadrocoptery. Je proto důležité vědět, jak dlouho je ještě možné baterii vybíjet, než dosáhne kritického vybití a hrozil by pád quadrocoptery. Pro měření byla napsána knihovna, která obsahuje následující funkce

- int getCapacity(); // vrátí kapacitu baterie v %
- double getVoltage(byte cell); // vrátí napětí na baterii cell(1-3) ve Voltech
- boolean isOK(); // vrací hodnotu TRUE pokud je celková kapacita nad 5% a ani jeden článek neklesl pod 3,8V. Jinak FALSE

Požití AD převodníku, které jsou součástí mikrokontroleru je velice jednoduché. Tento úkol se ještě zjednoduší použitím platformy Arduino. Příkazem analogRead() získáme vzorek naměřené hodnoty převedený do 10bitového binárního čísla(0-1023). Následně je potřeba provést zpětný výpočet na analogovou hodnotu. Mikrokontroler obsahuje pro AD převodníky referenční napětí, které je shodné s napájecím napětím, v našem případě tedy 5V. Pokud bychom chtěli měřit napětí, které dosahuje maximálně 3,3V můžeme na referenční pin přivést právě tuto hodnotu. Následný zpětný výpočet se provede jako

$$U = \frac{V_{ref}}{1024} \cdot V_{in}$$

kde U je naměřená hodnota napětí, V_ref je referenční napětí a V_in je vstupní měřené napětí. Takto jsme schopni změřit napětí v rozsahu 0 - V_ref. Modelářské baterie se však pohybují v rozmezí 3,7V - 4,2V, jak je psáno v sekci 2.4 a zároveň musíme řešit měření více článků dle obrázku 2.8 v sekci 2.4. Na pinu 1S lze tedy přímo naměřit jeho napětí. Na pinech 2S a 3S už jde o součty předchozích článků, které jsou navíc pomocí odporového děliče (sekce 4.3.4, obr. 4.12) zredukovány na napětí pod 5V. Je tedy potřeba provést potřebné korekce pro zpětné získání měřených hodnot.

$$Cell1 = \frac{V_{ref}}{1024} \cdot V1_{in} \cdot \frac{V_{1S}}{V_{ref}}$$
$$Cell2 = \frac{V_{ref}}{1024} \cdot V2_{in} \cdot \frac{V_{2S}}{V_{ref}} - Cell1$$
$$Cell3 = \frac{V_{ref}}{1024} \cdot V3_{in} \cdot \frac{V_{3S}}{V_{ref}} - Cell1 - Cell2$$

kde Cell(1 - 3) je konečná hodnota, kterou jsme naměřili na článku 1 - 3, V_ref je referenční napětí, $V(1-3)_in$ je vstupní měřené napětí a $V_{(1-3)S}$ je maximální napětí na pinech 1S až 3S. Následující kód ukazuje použití knihovny

```
#include <Wire.h>
#include "battery.h"

void setup()
{
    Serial.begin(9600);  // nastaví rychlost komunikace UART
}

void loop()
{
    int Capacity = getCapacity();
    Serial.print("Battery capacity: ");
    Serial.print(Capacity);
    Serial.print("%");

    Serial.println();
}
```

5.2.5 Čtení dat z přijímače

Do MCU jsou informace z přijímače přivedený jako PPM stream na jeden pin přerušení. Abychom určili, který pulz v PPM streamu je který příkaz, tak musíme nejdřív synchronizovat signál. Jediná část signálu, o které víme jistě, co znamená, je mezera mezi posledním a prvním pulzem. Tuto mezeru musíme lokalizovat a synchronizovat se na ní. Dále již stačí načítat jednotlivé pulzy a měřit jejich délku. V PPM streamu jsou kanály poskládány v následujícím pořadí, Yaw, Trainer, Pitch, Switch, Roll, Throttle. Ukázka získání jednotlivých kanálů z PPM streamu je v následujícím kódu

```
#define PPM_Int_Pin 0 // číslo přerušení
#define PPM_Pin 2 // číslo pinu přerušení
unsigned StartPeriod = 0;  // počáteční čas
boolean Sync = false; // informace zda byl signál synchronizován
unsigned channel = 0;  // číslo kanálu
int ppm_in[7] = {1500, 1000, 1500, 2000, 1500, 1000};  // pole pro uložení jednotlivých hodnot
      kanálů

void setup() {
   attachInterrupt(PPM_Int_Pin, calcInput, CHANGE);  // připojení přerušení
}

void calcInput() {
  if (!Sync) {
    if (digitalRead(PPM_Pin) == HIGH) {
      StartPeriod = micros();
    } else {
      if (micros() - StartPeriod > 5000) {  //  detekce mezery mezi pulzy
        Sync = true;
        StartPeriod = micros();
        channel = 0;
      }
    }
  } else {
    if (digitalRead(PPM_Pin) == HIGH) {
      ppm_in[channel] = (int)(micros() - StartPeriod);  // měření délky pulzu
      channel = (channel + 1) % 6;   // další kanál
    } else {
      StartPeriod = micros();
    }
  }
}

void loop() {
   // zde je možné psát vlastní kód, který využije naměřené hodnoty.
}
```

5.2.6 Komunikace s Raspberry PI

Pro komunikaci mezi quadrocopterou a Raspberry PI byl společně s kolegou Jiřím Kukačkou, který v rámci své bakalářské práce připravuje software pro Raspberry PI, vytvořen jednoduchý protokol, který není časově náročný a nezatěžuje tak chod vnitřní struktury quadrokoptery. Pokud pošleme dotaz od Raspberry PI k quadrocopteře, zasílá se vždy jen číslo dotazu, pokud Raspberry PI zasílá quadrocopteře údaje k nastavení (Gyro, Motor Speed), zasílá se číslo příkazu s přičtenou hodnotou 80h. Quadrocoptera vždy odpovídá ve formátu [kód dotazu + data]. Na požadavek k nastavení je odpověď jako kdyby se jednalo o dotaz = akceptování příkazu. V tabulce 5.2 jsou popsány jednotlivé příkazy, které umí quadrocoptera rozeznat.

Název příkazu	Kód příkazu	Zpráva	Popis příkazu
Gyro	0x01	[Roll][Pitch][Yaw]	Žádost o zaslání stavu náklonu int[2B][2B][2B]
Gyro Roll	0x02	[Roll]	Požadovaný úhel roll int[2B]
Gyro Pitch	0x03	[Pitch]	Požadovaný úhel pitch int[2B]
Gyro Yawl	0x04	[Yaw]	Požadovaný úhel yaw int[2B]
Motor Speed	0x0B	[Throttle]	Žádost o rychlost otáčení motorů int[1B]
Příkazy pouze pro čtení			
GPS	0x33	[Lat][Lon][Elev]	Žádost o zaslání souřadnic GPS float[4B][4B][4B]
GPS Latitude	0x34	[Lat]	Žádost o zaslání souřadnice GPS float[4B]
GPS Longitude	0x35	[Lon]	Žádost o zaslání souřadnice GPS float[4B]
GPS Elevation	0x36	[Elev]	Žádost o zaslání souřadnice GPS float[4B]
Battery Actual	0x3D	[Percentage]	Žádost o aktuální stav kapacity baterie v % int[1B]
Altitude	0x3F	[m]	Žádost o nadmořskou výšku float[4B]
Compass	0x40	[Degree]	Žádost o odchylku natočení quadrocoptery vůči severnímu pólu int[1B]
Accel X	0x47	[X]	Žádost o hodnotu akcelerometru v ose x float[4B]
Accel Y	0x48	[Y]	Žádost o hodnotu akcelerometru v ose y float[4B]
Accel Z	0x49	[Z]	Žádost o hodnotu akcelerometru v ose z float[4B]
Příkazy pro čtení sonarů			
Sonar Front	0x65	[Distance]	Žádost o hodnotu předního sonaru v cm int [2B]
Sonat Right	0x66	[Distance]	Žádost o hodnotu pravého sonaru v cm int [2B]
Sonat Back	0x67	[Distance]	Žádost o hodnotu zadního sonaru v cm int [2B]
Sonat Left	0x68	[Distance]	Žádost o hodnotu levého sonaru v cm int [2B]
Sonat Top	0x69	[Distance]	Žádost o hodnotu horního sonaru v cm int [2B]
Sonat Bottom	0x6A	[Distance]	Žádost o hodnotu spodního sonaru v cm int [2B]
Rezervované kódy			
State	0x7F	[Rdy, Err, 6x reserved]	Stav quadrocoptery [1B]

Tabulka 5.2: Kódy komunikačního protokolu mezi quadrocopterou a Raspberry PI.

Následující tabulka 5.3 ukazuje, jak by měla komunikace probíhat.

Čas	**QC**	**RPi**	**Vysvětlení**
1	-	0x01	QC čeká, RPi zjišťuje stav natočení QC
2	0x01 0x00 0x00 0x00	0x0B	QC odpovídá na dotaz natočení, RPi zjišťuje aktuální rychlost
3	0x0B 0x00	0x3D	QC odpovídá na dotaz rychlosti, RPi zjišťuje kapacitu baterie
4	0x3D 0x7E	0x40	QC odpovídá na dotaz kapacity baterie, RPi zjišťuje aktuální natočení QC
5	0x40 0x0A	0x7F	QC odpovídá na dotaz aktuálního směru natočení, RPi zjišťuje zda je QC připravena k letu
6	0x7F 0x80	-	QC odpovídá stavem připraveno, RPi pracuje
7	-	0x8B 0x10	QC čeká, RPi dává povel k roztočení motorů

Tabulka 5.3: Ukázka komunikace mezi quadrocopterou [QC] a Raspberry PI [RPi].

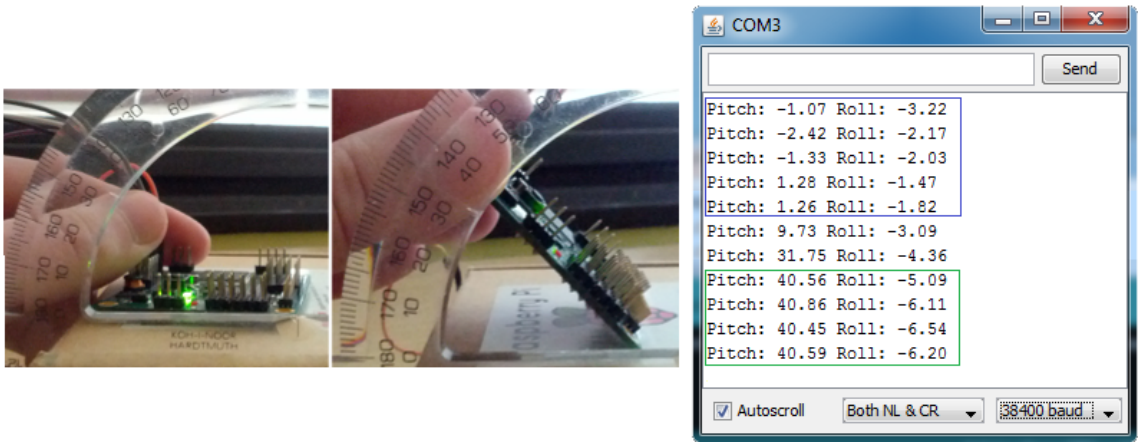
Kapitola 6

Testování

V rámci testování byly otestovány pouze komponenty potřebné k letu. U ostatních periferii byla otestována pouze základní funkčnost.

6.1 Gyroskop a konstrukce

Pro otestování funkce gyroskopu byl sepsán program, který po UART do počítače posílá hodnoty aktuálního náklonu desky. Na obrázku (6.1) je vidět měření probíhá. Na levém obrázku je vidět jak je deska opravdu nakloněna, na pravém obrázku hodnoty vyčtené ze senzoru.

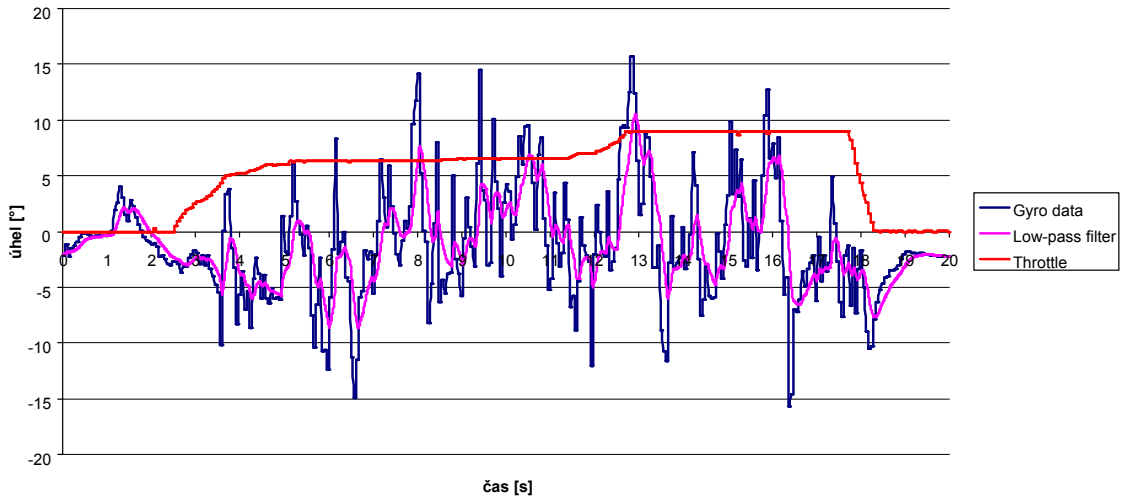


Obrázek 6.1: Ukázka naklonění desky s gyroskopem.

Před zahájení testování PID regulátorů byl proveden test vibrací konstrukce. Dle předpokladů byl jimi senzor gyroskopu ovlivněn. Pro testování byla quardocoptera upevněna v jedné ose a v druhé ose byli spuštěny motory. Konstrukce se byla ručně nakloněna na +45° a následně na -45°. Výsledná data jsou vidět v grafu 6.2.

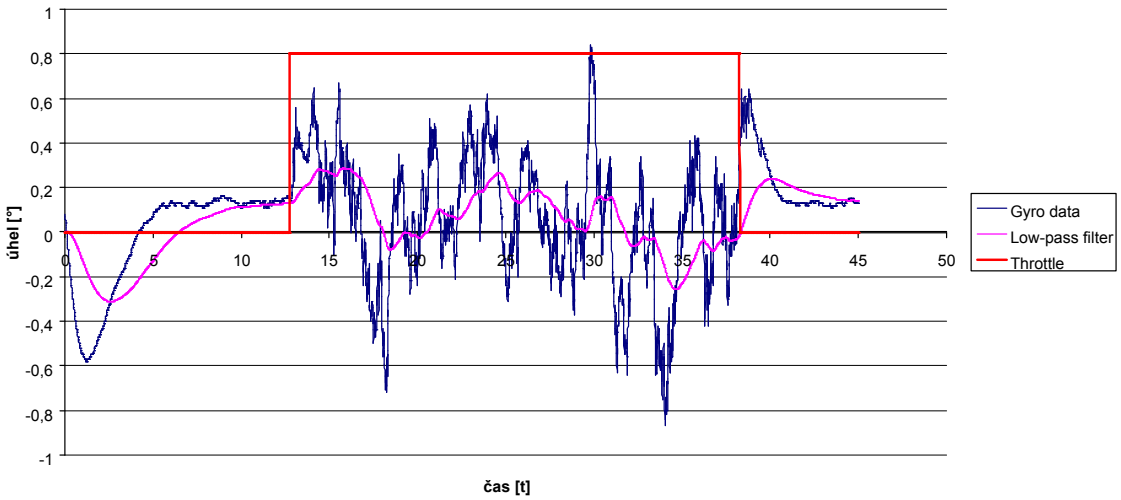
Obrázek 6.2: Graf výstupu gyroskopu zkreslený rušením z vibrací pro pohybující se osu.

Z grafu je patrné, že senzor byl velmi ovlivněn vibracemi z motorů. Pokus o odfiltrování tohoto rušení selhává, jelikož data jsou příliš zarušená. Abychom si vytvořili představu o tom, jak velký vliv na měření vibrace mají, provedeme následující měření s quadrokopterou fixovanou na 0°.



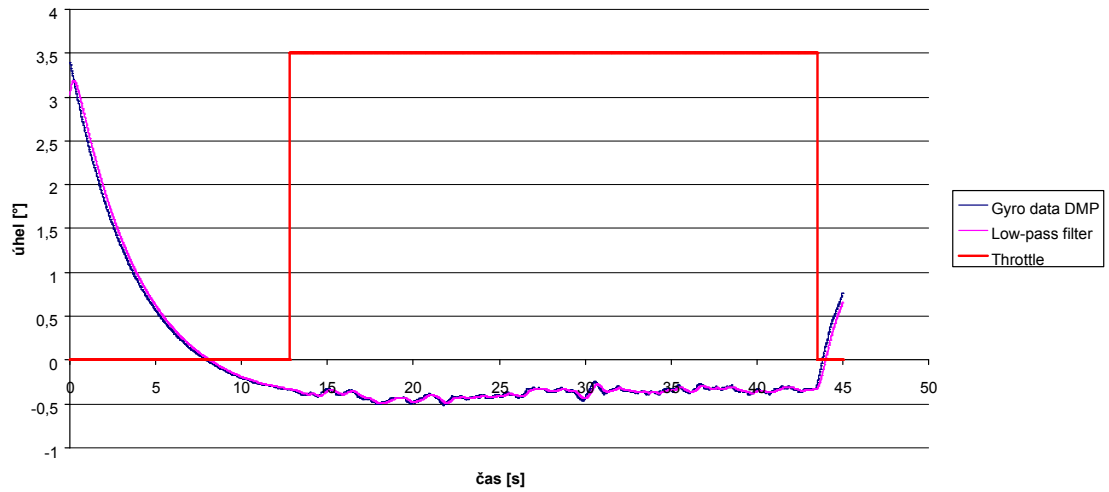
Obrázek 6.3: Graf výstupu gyroskopu zkreslený rušením z vibrací pro pevnou osu.

Pro odstranění vibrací byly nejprve vyváženy vrtule. Odchylka ve vahách listů vrtule by mohla zapříčinit vibrační efekt, který můžeme přirovnat k vibracím u mobilních telefonů. Dalším zdrojem vibrací, kromě vrtulí, může být právě fixace osy k pevnému bodu. Pro zamezení vibrací při použití testovacího uchycení byla quadrocoptera uchycena v gumové podložce, která vibrace pohltila. Na grafu (6.4) je patrné výrazné zlepšení. Po aplikaci lineárního filtru jsou již data použitelná pro stabilizaci quadrocoptery.



Obrázek 6.4: Graf výstupu gyroskopu zkreslený rušením z vibrací pro pevnou osu, lepší uchycení.

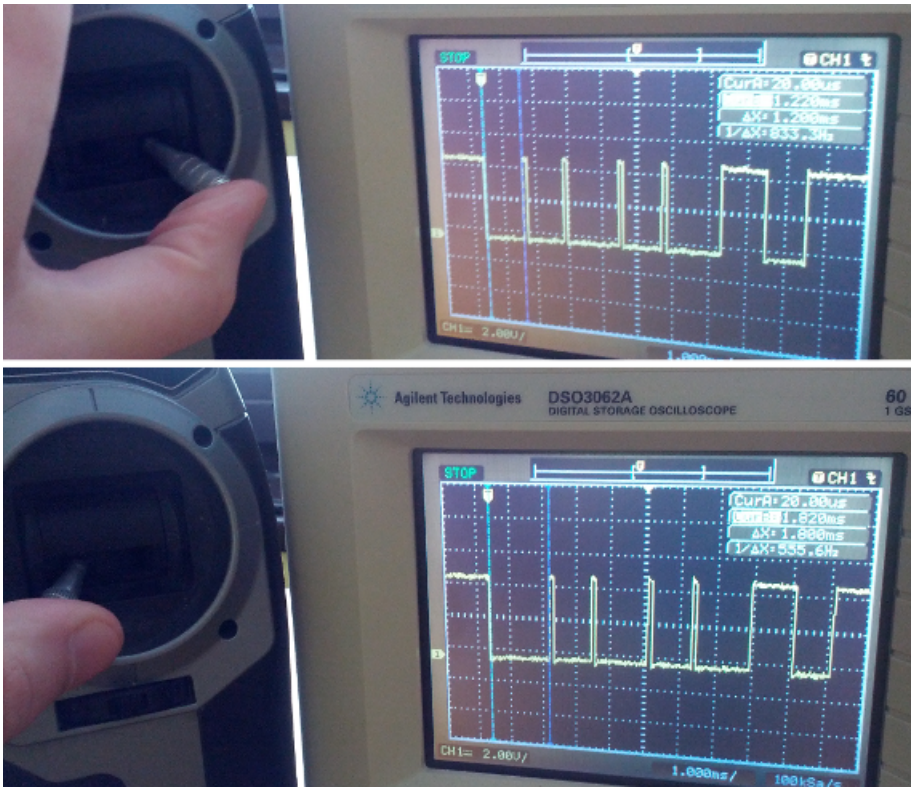
Na následujícím grafu (obr. 6.5) je vidět měření za stejných podmínek, ale byl použit DMP procesor uvnitř senzoru.



Obrázek 6.5: Graf výstupu gyroskopu z DMP procesoru.

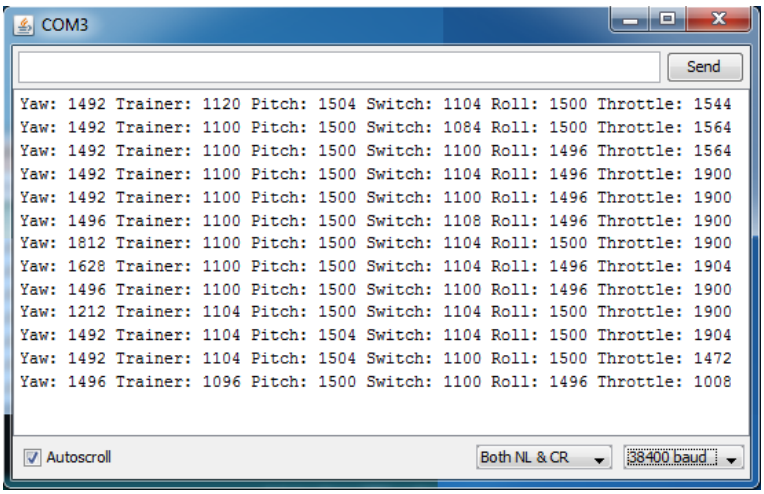
6.2 Rádiový vysílač

Na obrázku 6.6 můžeme vidět jak vypadá signál na výstupu PPM encodéru a jak reaguje na změnu v jednom kanále. Signál je invertovaný, informaci zde proto nese logická 0. Na spodním obrázku můžeme vidět změnu šířky pulzu, kterou vyvolá změna na jednom z kanálů přijímače.



Obrázek 6.6: Měření PPM streamu na PPM encoderu.

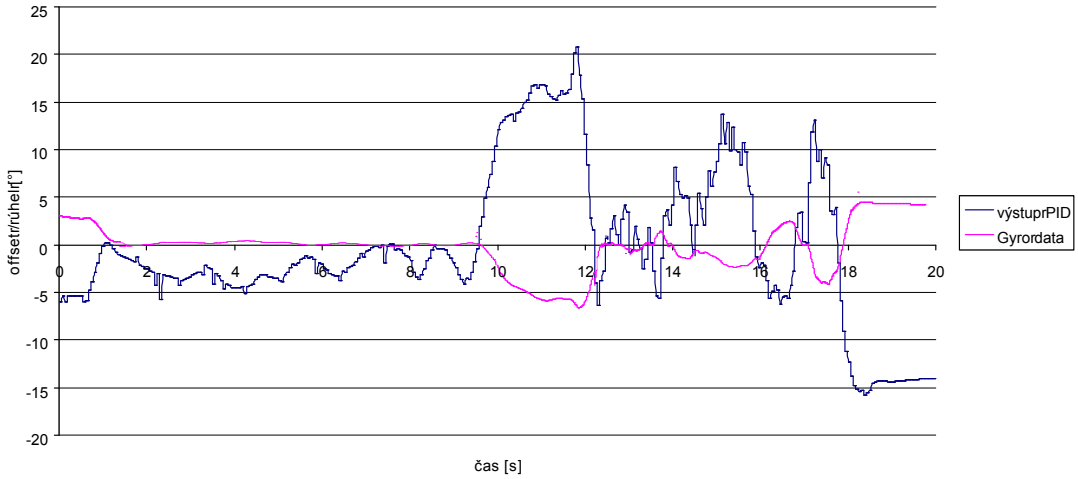
Z radiového přijímače jsou hodnoty kanálů vystavovány postupně s malou časonou prodlevou. Kanály jsou za sabou v pořadí Yaw, Trainer, Pitch, Switch, Roll a Throttle. Na obrázku 6.7 můžeme vidět, že hodnoty nejsou stabilni. Je proto potřeba použít filtr pro odstranení malích zákmitů, ideální je dolní propust.



Obrázek 6.7: Dekódované hodnoty z PPM streamu.

6.3 PID regulátor

Při testování PID regulátoru bylo nejtěžší nalézt správné konstanty K_p, K_i a K_d pro každý regulátor, který je v programu použit. Při testování bylo zjištěno, že i přes vyváženost konstrukce, není možné quadrocopteru stabilizovat. Důvodem byla rozdílná rychlost rotace motorů i přes to, že regulátory otáček měli na vstupu stejná data. Do programu byli proto přidány dva PID regulátory navíc, kteří měli na starost vykompenzovat případné rozdíly. V grafu 6.8 je vidět jak PID kompenzuje rychlost otáčení jednoho z motorů a jak na zm2nu reaguje quadrocoptera. Jak je vidět, kompenzace funguje výborně a quadrocoptera je stabilní, kolem 10 s bylo třikrát po sobě vyvoláno uměle rozvážení quadrokoptery.



Obrázek 6.8: Graf znázorňující korekci rozdílu otáček motorů v jedné ose.

Konstanty pro doladění funkce PID regulátorů jsou vypsány v tabulce 6.1. Tyto konstanty platí pouze pro použitou konstrukci. V případě, že by došlo k jakékoli změně v konstrukci, musely by se konstanty upravit, protože se zároveň změní i vlastnosti systému, který řídíme.

PID	P	I	D
Pitch korekce	0,54	0,2	0,05
Roll korekce	0,56	0,18	0,05
Pitch stabilizace	0,83	1,2	0,2
Roll stabilizace	0,79	1,25	0,17

Tabulka 6.1: Konstanty PID regulátorů.

Kapitola 7

Závěr

Výsledkem práce je funkční quadrocoptera schopná letu. V časovém úseku práce se však nepodařilo uvést do provozu streamovaní videa z Raspberry PI přes wifi síť. Tento úkol byl již ale nad rámec práce a nebyla na něj proto kladena taková důraznost jako na ostatní části.

Jedním z problémů při dokončování práce, byl návrh a výroba plošného spoje řídící desky. Cílem bylo vyrobit desku co nejmenší, aby mohla konkurovat jiným řešením nejen ve funkční oblasti . I přes veliké problémy při návrhu, kdy byl hlavní problém v nalezení správného umístění součástek, se nakonec podařilo vyrobit dvouvrstvý plošný spoj o velikosti 52 cm x 52 cm. Velkou mírou k tomuto cíly přispěl výběr integrovaných obvodů, které v sobě mísili skvělé funkční vlastnosti s malými rozměry. Oživení desky doprovázely potíže s jejím naprogramováním. Nakonec se však desku podařilo oživit.

Bohužel se do dokončení toho textu nepodařilo sestavit druhou verzi konstrukce, kterou měl na starost Vratislav Polívka z Fakulty architektury ČVUT jakožto konzultant v oboru modelářství.

V rámci práce byl sepsán základní program pro stabilizaci letu quadrocoptery. Vytváření softwaru pro řízení zpomalilo zdlouhavé pročítání datasheetu jednotlivých senzorů. Nejvíce problémů však vznikalove spojení se stabilizací. Nalezení konstant pro PID regulátory se ukázalo jako velice náročná činnost, kterou navíc stěžovali nežádoucí vlivy jako vibrace motorů a následné ovlivňování senzorů. Dále rozdíly rychlosti otáčení u jednotlivých motorů, které jsou zapříčiněny nepřítomností zpětné vazby, která by podávala informace o skutečné rychlosti motoru. Tento nedostatek se dá odstranit výměnou motoru za motory s Hallovou sondou. V této práci se tento problém řešil přidáním dalšího PID regulátoru, který prováděl korekce v rychlosti otáčení motorů. Konstany pro stabilní s danou konstrukcí se nakonec podařilo nalést.

Literatura

[1] Záhlava, V. *OrCAD 10.* Grada Publishing,Praha 2004.

[2] Záhlava, V. *Návrh a konstrukce desek plošných spojů.* Vydavatelství ČVUT, Praha 2005.

[3] Vobecký, J. a Záhlava, V. *Elektronika - součástky a obvody, principy a příklady.* Třetí rozšířené vydání Grada Publishing, Praha 2005.

[4] *Latexdocweb - online manuál, 27.12.2011*.
http://cstug.cz.

[5] *RaspberryPI - popis, 27.4.2014*.
http://en.wikipedia.org.

[6] *Ardupilot - popis, 27.4.2014*.
http://ardupilot.com.

[7] *AeroQuad - popis, 27.4.2014*.
https://github.com/AeroQuad/AeroQuad.

[8] *Bluecopter - popis, 27.4.2014*.
https://github.com/baselsw/BlueCopter.

[9] *ArduQuad - popis, 27.4.2014*.
http://diydrones.com.

[10] *ATmega2560 - popis, 27.4.2014*.
http://www.atmel.com.

[11] *Arduino - popis, 27.4.2014*.
http://www.arduino.cc.

[12] *ATmega328p - popis, 27.4.2014*.
http://www.atmel.com.

[13] *MPU-6000 - popis, 27.4.2014*.
http://www.invensense.com.

[14] *MPL3115A2 - popis, 27.4.2014*.
http://www.freescale.com.

[15] *HMC5883L - popis, 27.4.2014*.
http://www.magneticsensors.com.

[16] *GPS - popis, 27.4.2014*.
http://www.trimble.com.

[17] *sonar HC-SR04 - popis, 27.4.2014*.
http://www.elecfreaks.com.

[18] *baterie FOXY G2 - popis, 27.4.2014*.
http://www.pelikandaniel.com.

[19] *vysílač Spectrum DX5e - popis, 27.4.2014*.
http://www.spektrumrc.com.

[20] *přijímač Spectrum AR600 - popis, 27.4.2014*.
http://www.spektrumrc.com.

[21] *Brushed DC electric motor - popis, 27.4.2014*.
http://en.wikipedia.org.

[22] *Brushless DC electric motor - popis, 27.4.2014*.
http://en.wikipedia.org.

[23] *Pulzně šířková modulace - popis, 27.4.2014*.
http://cs.wikipedia.org.

[24] *knihovna MPU-60x0 - popis, 27.4.2014*.
https://github.com/jrowberg/.

[25] *Kalmanův filtr - popis, 27.4.2014*.
http://en.wikipedia.org.

[26] *Popis protokolu NMEA 0183 - popis, 27.4.2014*.
http://aprs.gids.nl/nmea/.

[27] *TME, obchod s elektronickými součástkami, 27.12.2011*.
http://www.tme.cz.

[28] *Pragoboard a.s., výrobce plošných spojů, 27.12.2011*.
http://www.pragoboard.cz.

Kapitola 8

Seznam použitých zkratek

UAV Unmanned Aerial Vehicle

USART Universal Synchronous / Asynchronous Receiver and Transmitter

SPI Serial Peripheral Interface

I^2 Inter-Integrated Circuit

GPS Global Positioning System

MEMS Micro-Electro-Mechanical Systems

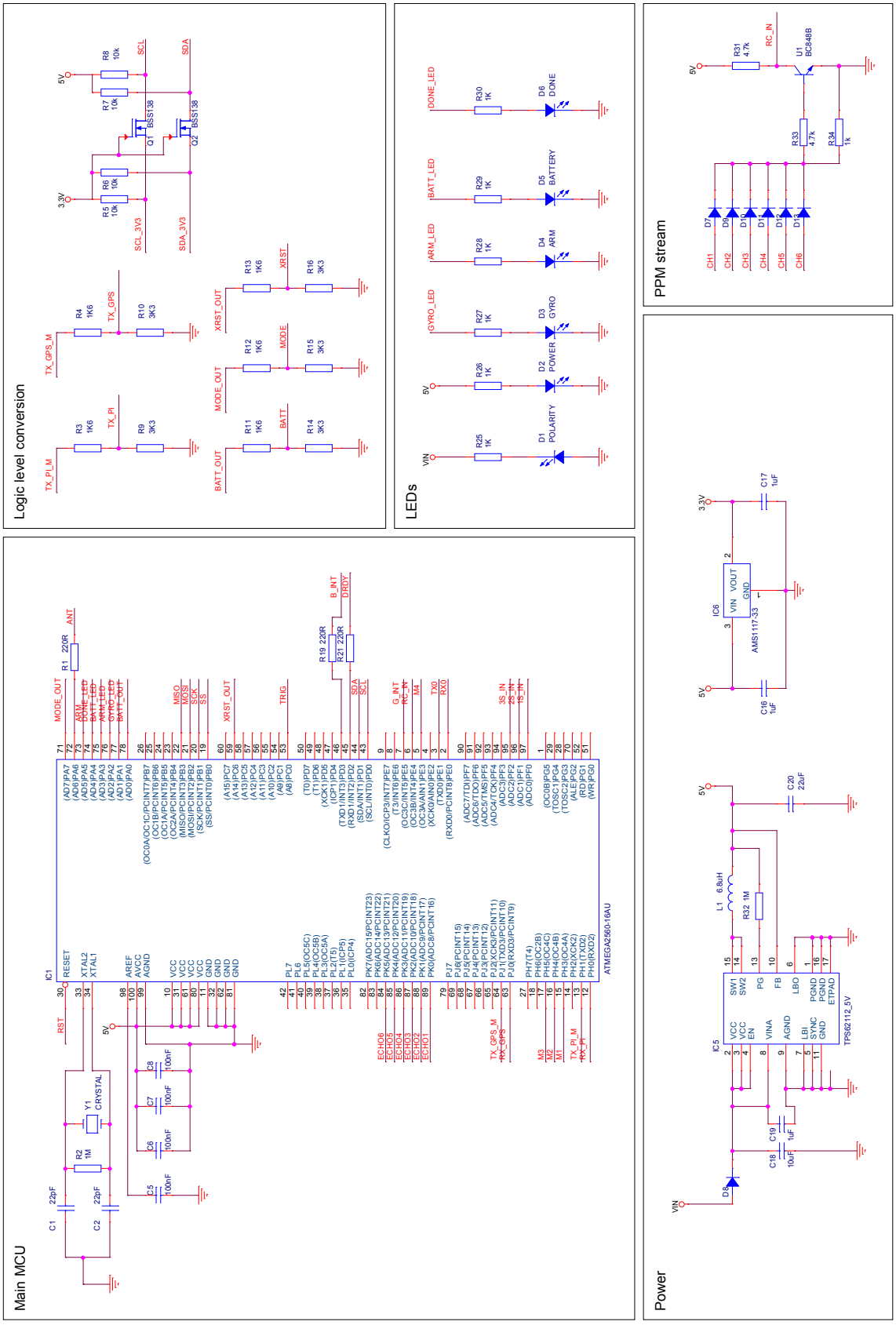
MCU Microcontroller unit

PWM Pulse-width modulation

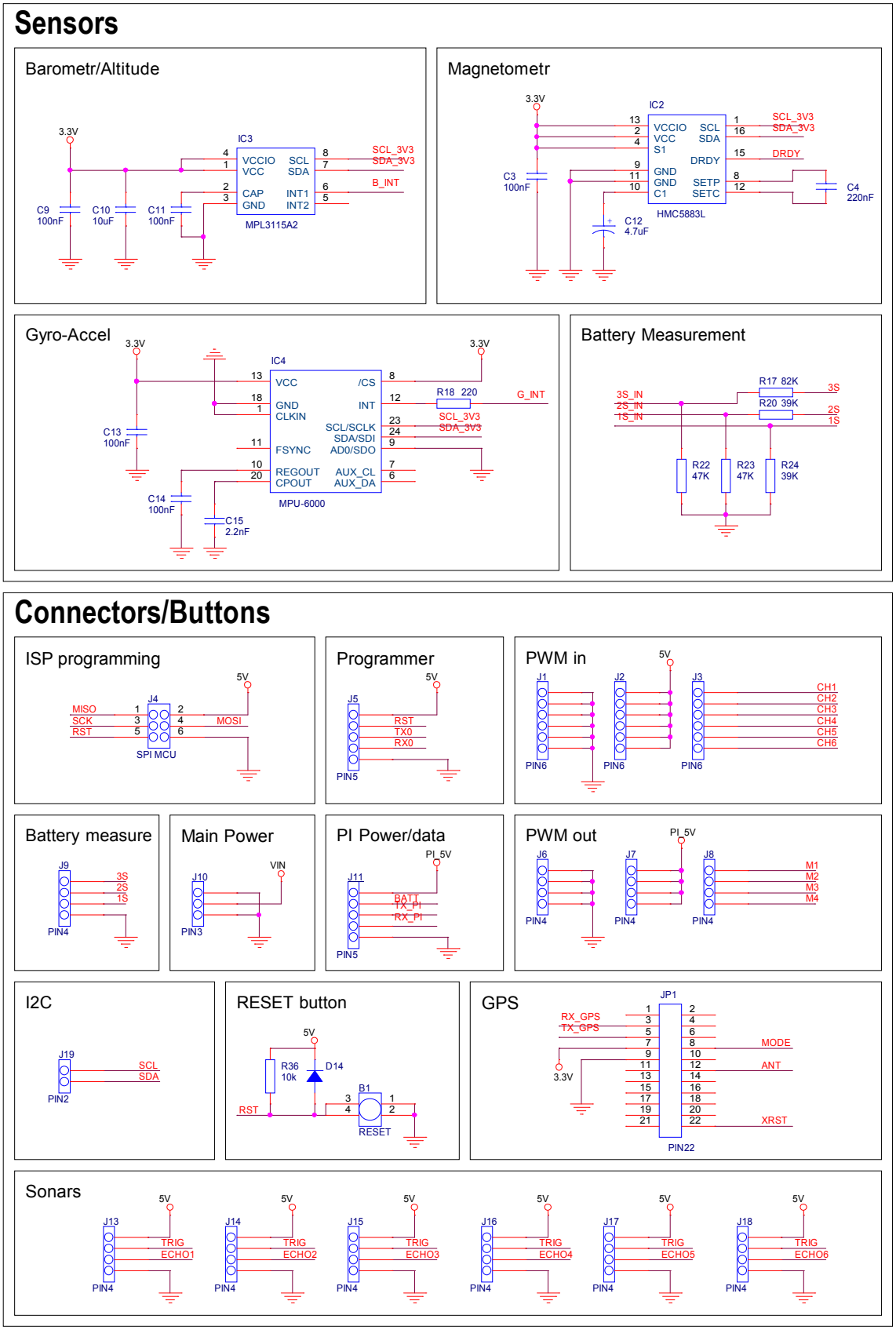
DMP Digital Motion Processor

Příloha A

Schéma desky



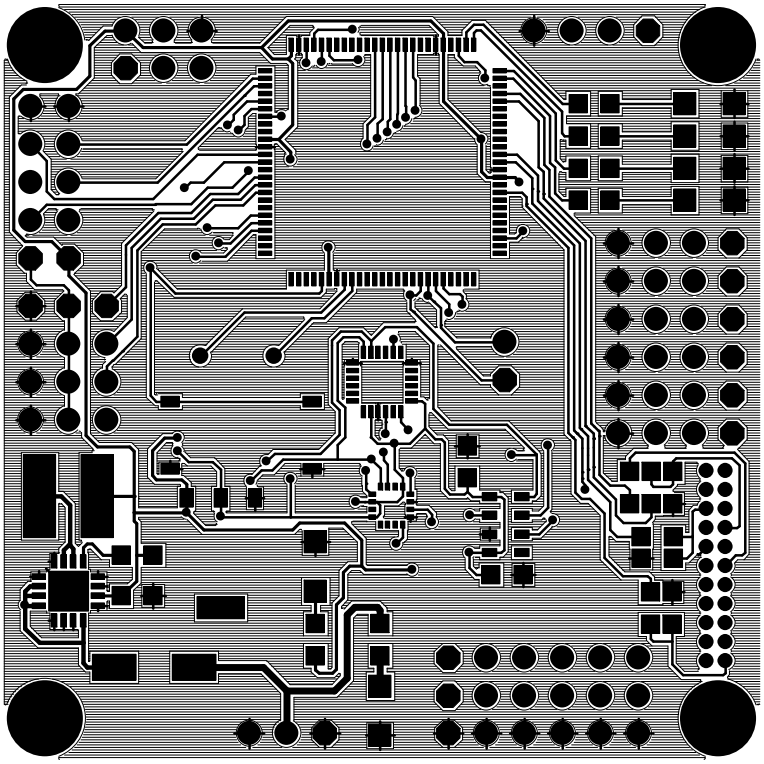
Obrázek A.1: Schéma zapojení MCU, zdroje a převodníků.



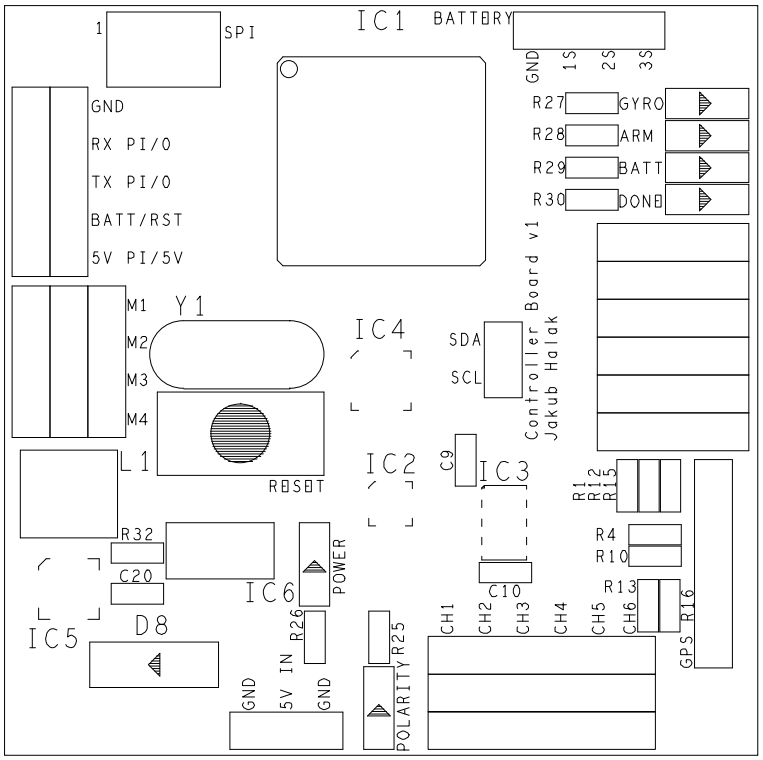
Obrázek A.2: Schéma zapojení senzorů a konektorů.

Příloha B

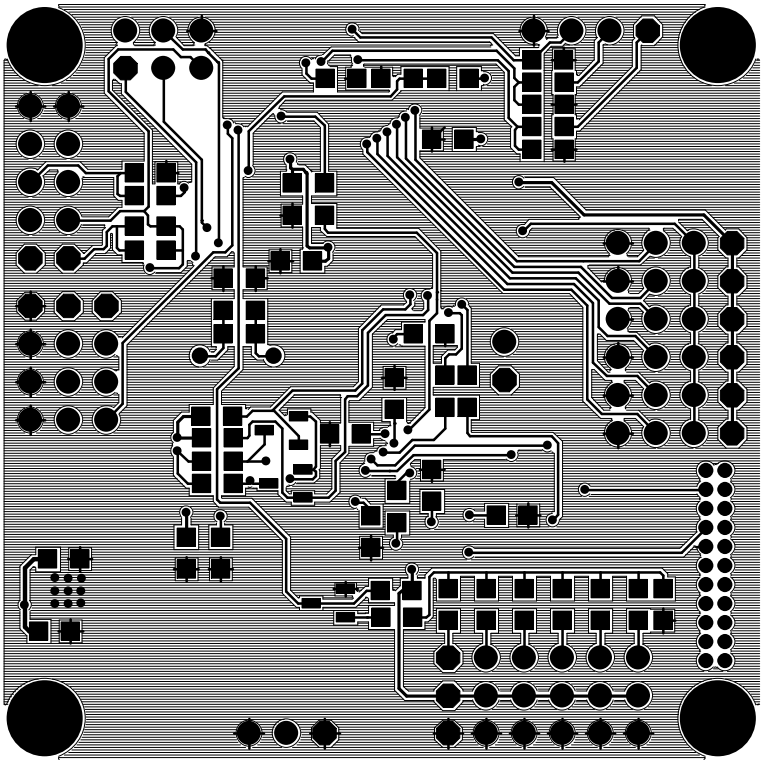
DPS a rozmístění součástek



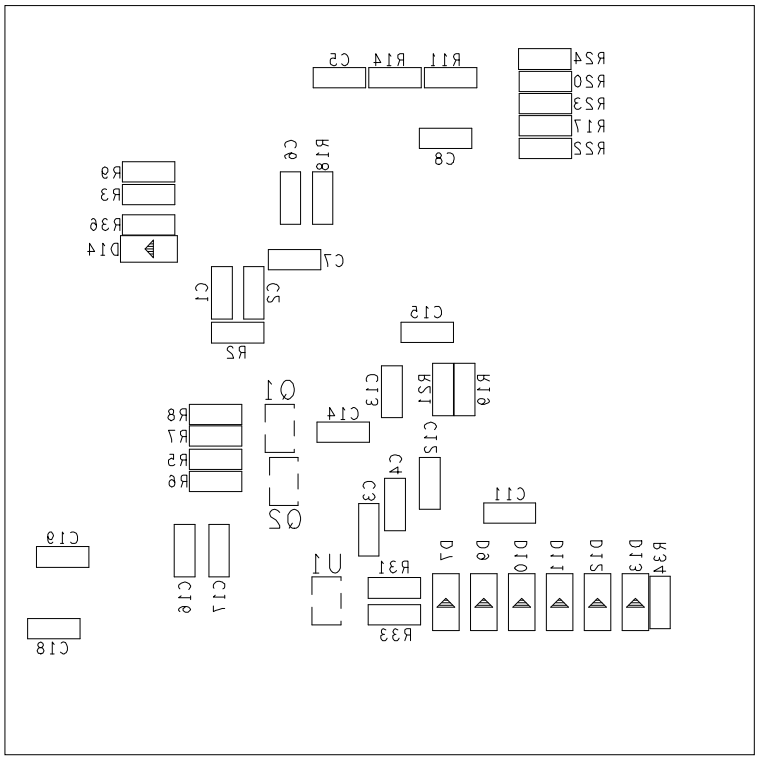
Obrázek B.1: Vrstva TOP



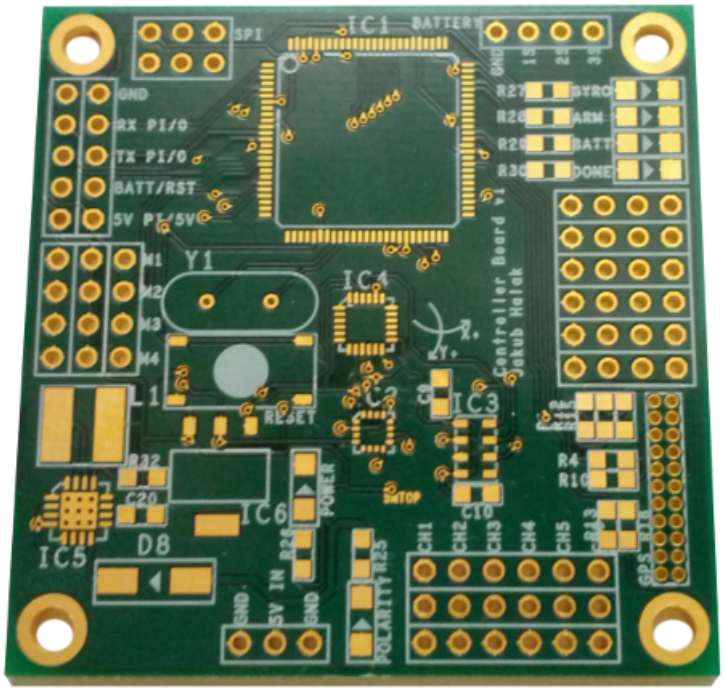
Obrázek B.2: Rozmístění součástek TOP.



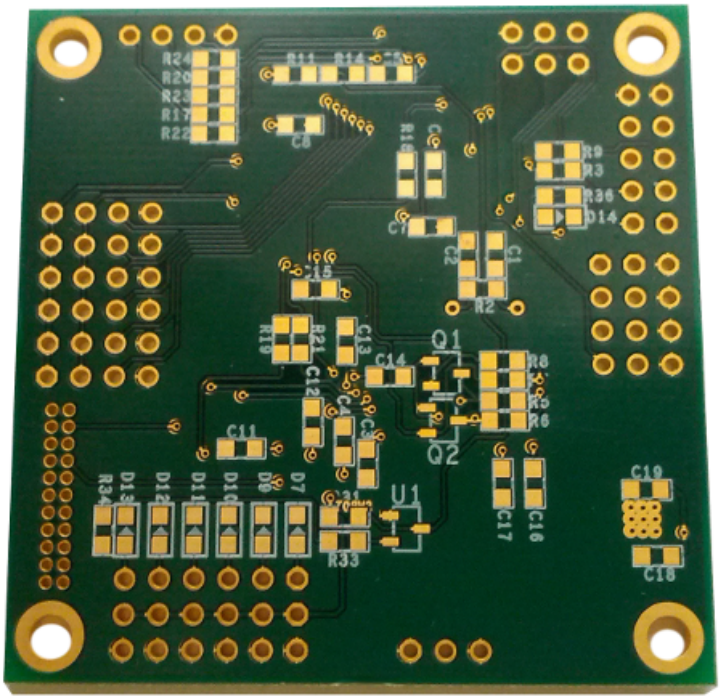
Obrázek B.3: Vrstva BOTTOM



Obrázek B.4: Rozmístění součástek BOTTOM.



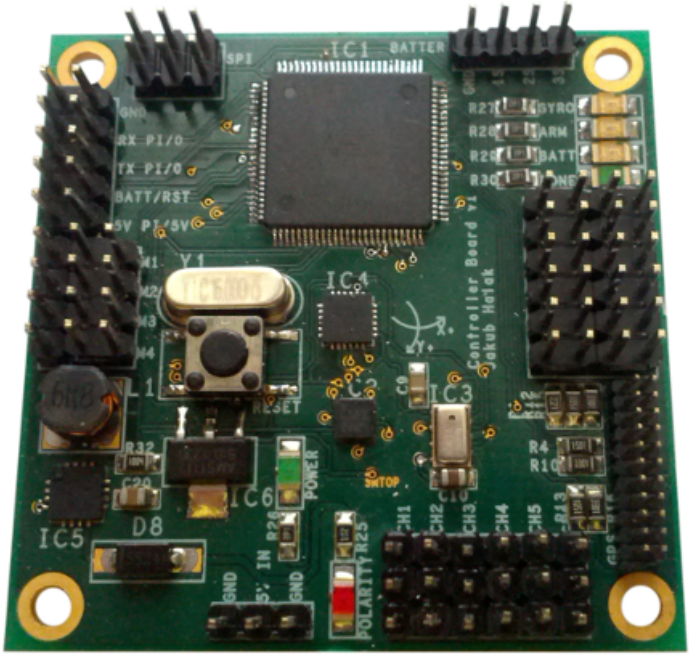
Obrázek B.5: Neosazená deska vrstva TOP



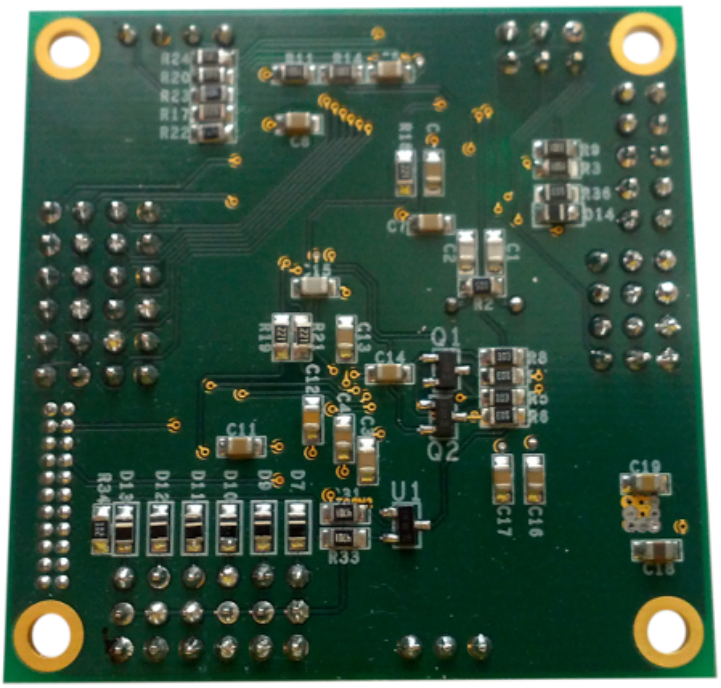
Obrázek B.6: Neosazená deska vrstva BOTTOM.

Příloha C

Osazená deska



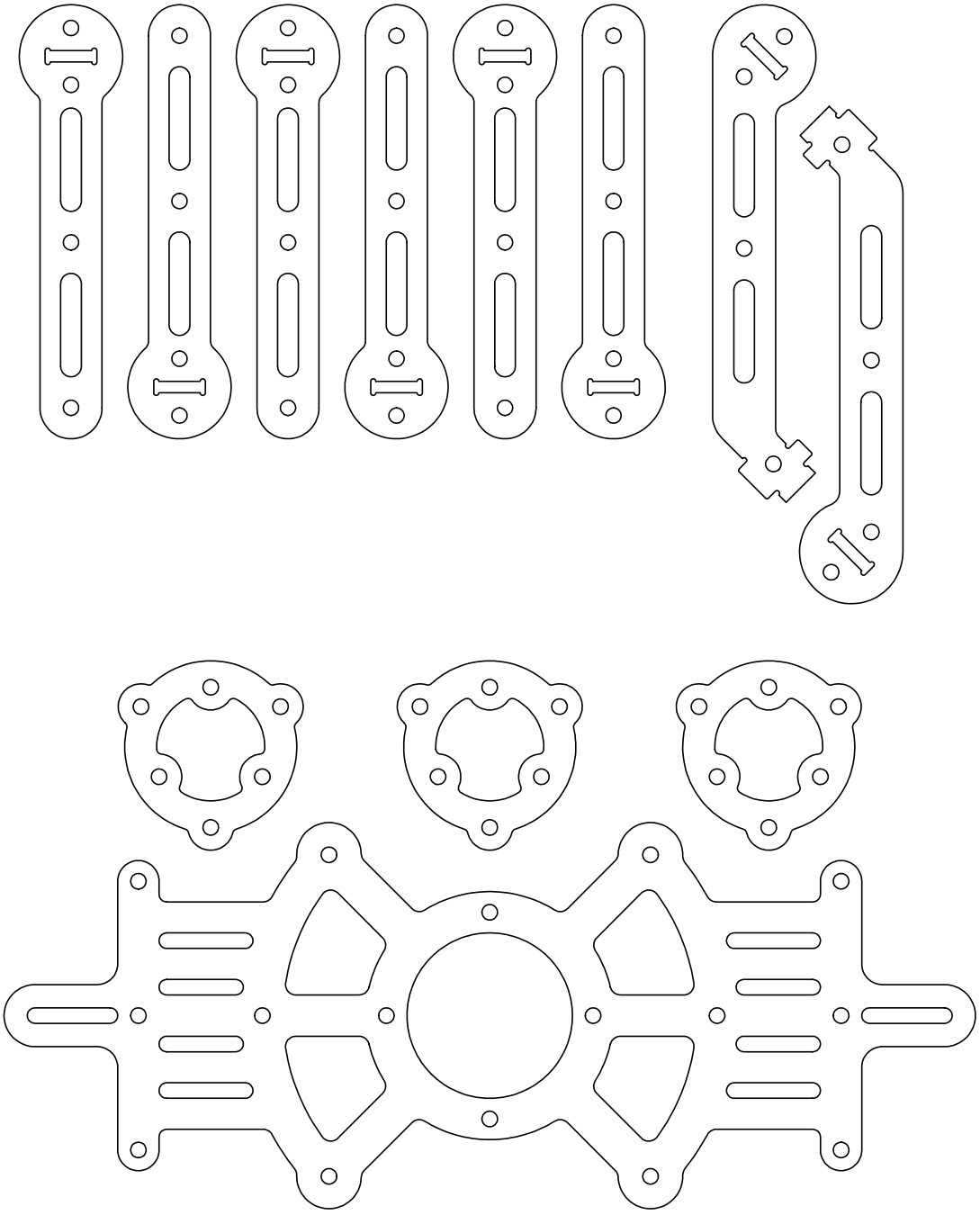
Obrázek C.1: Osazená deska vrstva TOP



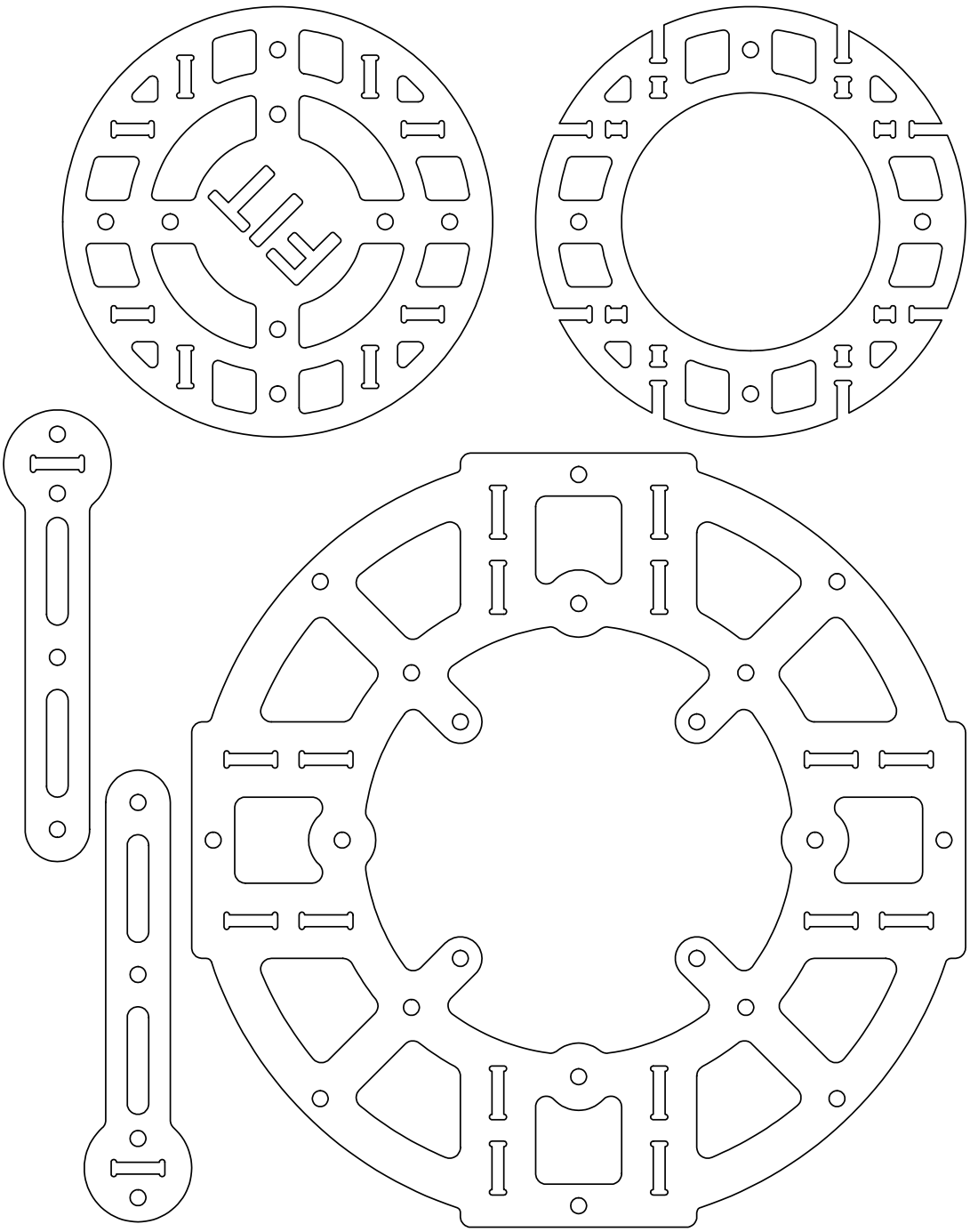
Obrázek C.2: Osazená deska vrstva BOTTOM.

Příloha D

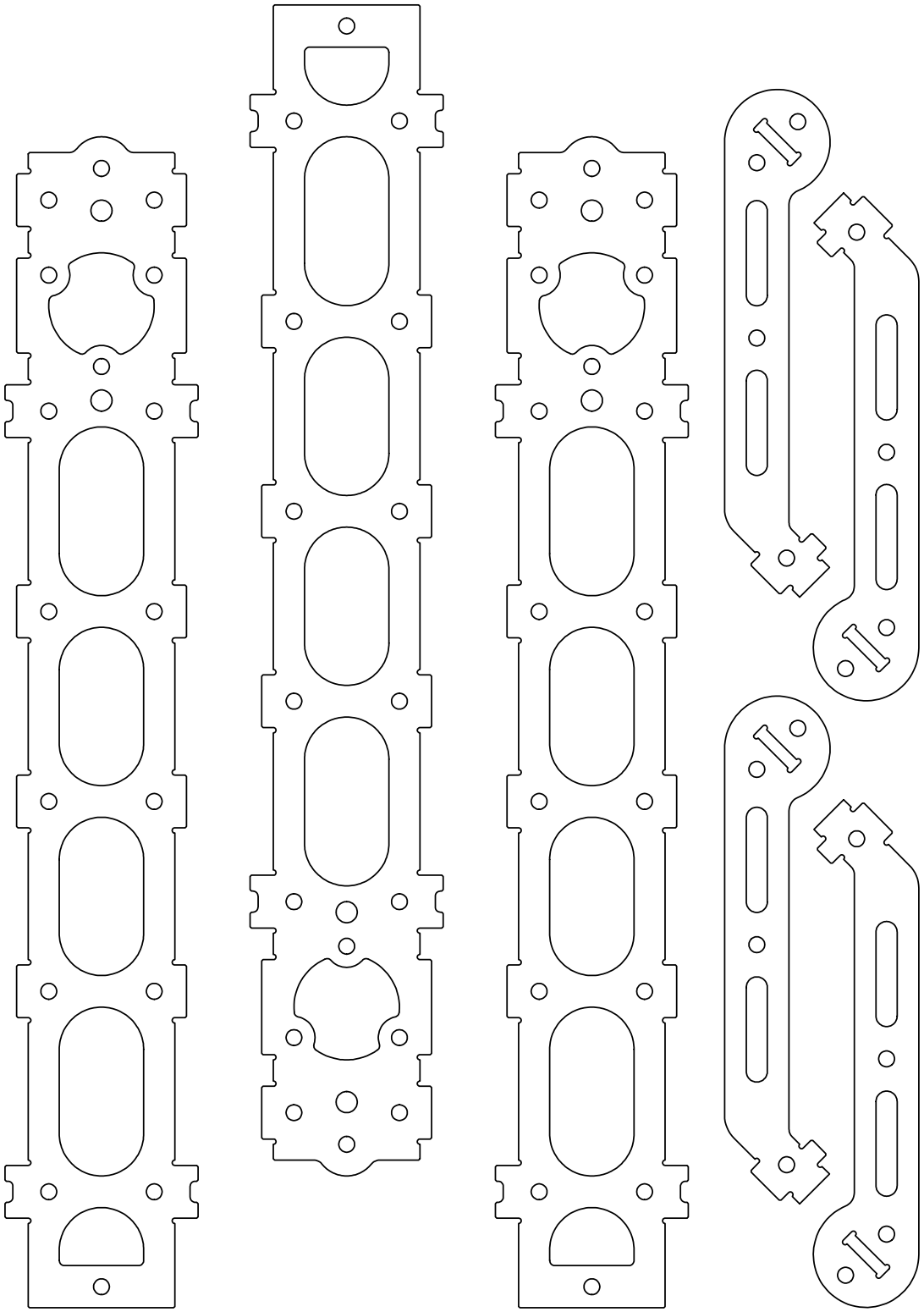
Nakres konstrukce



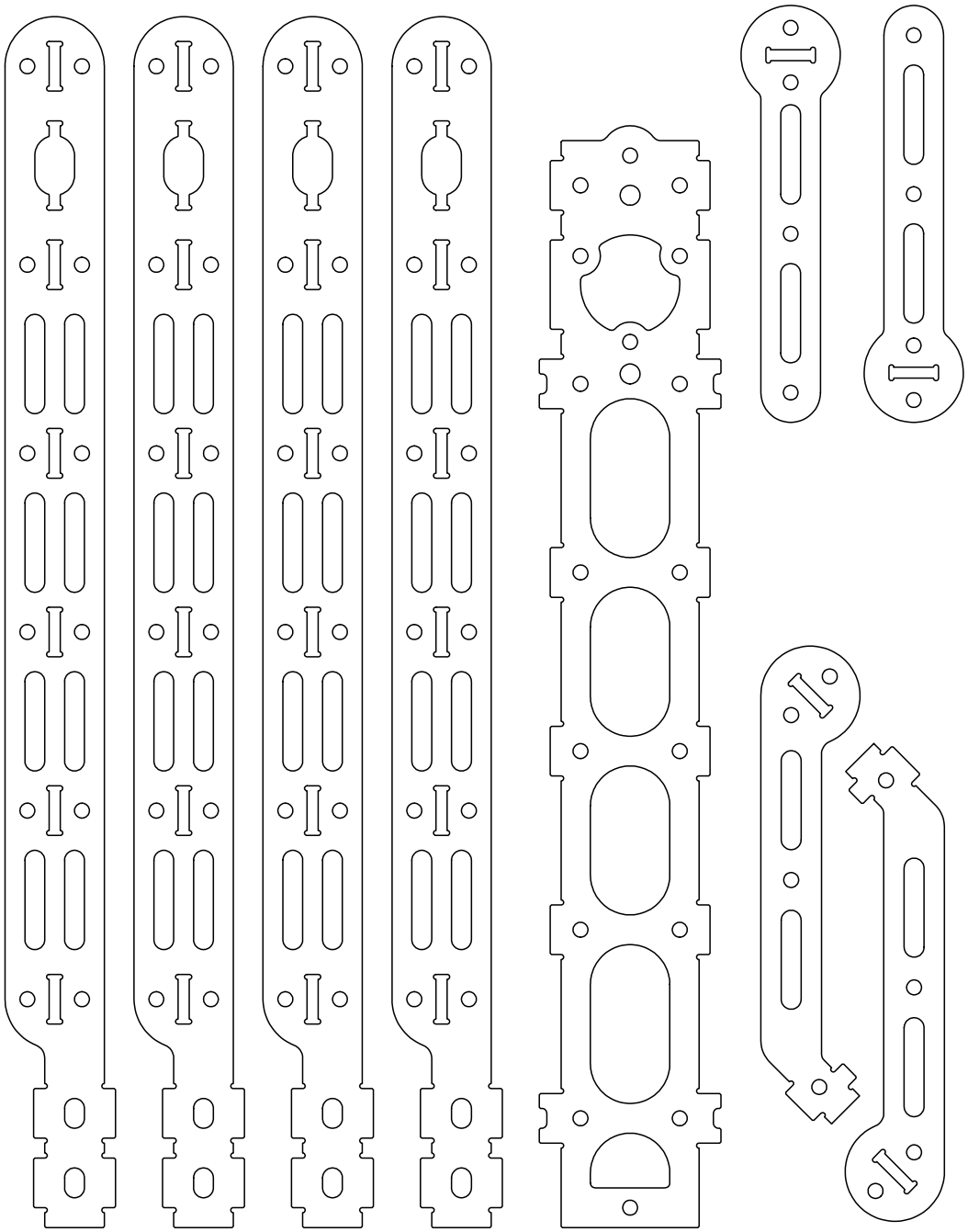
Obrázek D.1: Držák na baterii a vzpěry.



Obrázek D.2: Středový kruh a vzpěry.



Obrázek D.3: Ramena.



Obrázek D.4: Ramena.

Příloha E

Obsah přiloženého CD

Na přiloženém CD jsou přiloženy zdrojové kódy,katalogové listy použitých obvodů a projekty z programu OrCad 16.3.

- Adresář Firmware
  Adresář obsahuje kompletní zdrojové kódy.
- Adresář Datasheet
  V adresáři jsou uloženy katalogové listy použitých obvodů.
- Adresář PCB
  Adresář obsahuje dvě složky (Schema a PCB). Ve složce Schema je kompletní projekt se schématem z programu OrCAD-Capture 16.3. Složka PCB obsahuje kompletní projekt návrhu PCB včetně výrobních podkladů z programu OrCAD-PCBEditor 16.3.