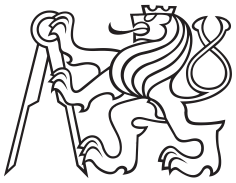
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Systém aktivního řízení pro RC auto

David Vošahlík

Vedoucí: doc. Ing. Martin Hromčík, Ph.D.
Obor: Systémy a řízení
Studijní program: Kybernetika a robotika
Květen 2018

Poděkování

Rád bych poděkoval svému vedoucímu doc. Ing. Martinu Hromčíkovi za jeho cenné rady a odborné konzultace, které mi ve vypracování této práce velmi pomohly. Dále bych také rád zmínil Ing. Martina Helmicha z fakulty strojní, který pomohl s mechanickým řešením RC modelu. Chtěl bych mu poděkovat za jeho konzultace, rady a v neposlední řadě za vlastní mechanické zpracování.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. května 2018

Abstrakt

Tato bakalářská práce pojednává o návrhu a realizaci systému aktivního řízení RC auta.
Cílem této práce je vytvořit testovací platformu pro účel potvrzení principů platných u řídících zákonů aplikovatelných pro automobilový průmysl a jejich následnou validací.
Použitý RC model obahuje dva bezsenzorové BLDC motory a digitální servo pro zatáčení. V práci je popsán jejich návrh a implementace do modelu. Následně je diskutován návrh a výběr senzorických jednotek použitých v modelu.
Dále je pojednáváno o výběru řídící jednotky a implementaci programu do ní. Jako řídící mikrokontrolér byl vybrán ATSAM3X8E v jednotce Arduino Due. Řídící systém ovládá napětí na jednotlivých motorech a natočení serva.
Práce se zabývá implementací komunikace senzorů s řídící jednotkou. Je zde popsán návrh, experimentální ladění, validace a implementace jednoduchých zpětnovazebních a přímovazebních řídících zákonů.

Klíčová slova: torque vectoring, Arduino, senzor, inerciální senzor, active control, RC model, embedded systém

Vedoucí: doc. Ing. Martin Hromčík, Ph.D.

Abstract

This bachelor thesis deals with the design and implementation of an active RC control system.
The aim of this work is to create a test platform for confirming the control laws' principles applicable to the automotive industry and their subsequent validation.
The used RC model contains two BLDC motors and a digital servo used for turning. The thesis describes their design and implementation into the model. The thesis then discusses the design and selection of sensor units used in the model.
It also deals with the selection of the control unit and the subsequent implementation of the program into it. As a microcontroller, the ATSAM3X8E in the Arduino Due unit was selected. The control system controls the voltage on the individual motors and the turning angle of servo.
The thesis deals with implementation of sensor communication with the control unit. It describes the design, experimental tuning, validation and implementation of simple feedback and feedforward control laws.

Keywords: torque vectoring, Arduino, sensor, inertial sensor, active control, RC model, embedded system

Title translation: Active control for an RC scale car

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Tato bakalářská práce se zabývá návrhem a následnou realizací RC modelu za účelem návrhu a validace jednoduchých řídících zákonů. Následným cílem celého tohoto projektu, který převyšuje rámec této práce je návrh a testování principů řídících zákonů aplikovatelných pro automobilový průmysl.
U tohoto projektu jsou ambice na vytvoření zcela nových regulačních systémů, které budou moci být použity v automotive odvětví. Na tomto modelu bude také možno věnovat se (z mnoha důvodů jako např. výukových důvodů) návrhu a implementaci již použitých systémů jako jsou ESP (Electronic Stability Program), který zahrnuje plno systémů jako třeba active steering, ASR (Anti-slip regulation), atd. Celý tento projekt vznikal od úplných základů. Nejprve se navrhlo, nakoupilo a zpracovalo mechanické řešení modelu. Vše toto probíhalo za součinnosti kolegů z fakulty strojní. Po konzultacích s Ing. Helmichem a Ing. Hromčíkem jsme se rozhodli použít silniční model o velikosti 1:10. Dále se navrhly a nakoupily dva modelářské BLDC motory včetně driverů - v modelařině nazývaných regulátorů - a servo. Pro upevnění řídící elektroniky na model se udělal výtisk na 3D tiskárně, který se pomocí suchých zipů připevnil k modelu. Na 3D výtisk se upevnila řídící jednotka Arduino Due společně se senzorickou jednotkou MPU6050, která obsahuje tříosý gyroskop a tříosý akcelerometr.
Dále jsem pro potřeby spojení mezi moduly, napájení elektroniky a zpracování signálu z proudových senzorů vypracoval a nechal vyrobit DPS, kterou jsem následně osadil a oživil.
V jednotce byla rozchozena komunikace se senzory a telemetrií. Dále zde byly implementovány jednoduché zpětnovazební a přímovazební řídící zákony, které stabilizují RC model a zjednodušují jeho ovládání.
Tento systém je s určitými úpravami přenositelný na osobní automobily.

1.1 Stručný obsah jednotlivých kapitol

První kapitola se zabývá úvodem do práce.
Druhá kapitola pojednává o mechanickém zpracování RC modelu. Je zde diskutován výsledek mechanického sestavení modelu, které probíhalo za spolupráce s kolegy z fakulty strojní.
Třetí kapitola pojednává o návrhu a výběru jednotlivých senzorických modulů použitých v modelu. Je zde popsán výběr motorů a telemetrické jednotky. V této kapitole je také popsána spojovací DPS, která se vyrobila v rámci projektu.
Ve čtvrté kapitole je popsán návrh a výběr řídící jednotky. V této kapitole je také popsán implementovaný software v řídící jednotce.
V páté kapitole jsou diskutovány implementované řídící zákony, jejich architektura a validace. Tyto zákony jsou zde následně zhodnoceny.
V závěru je uveden seznam mých osobních přínosů a návrh na další postup v projektu.

1.2 Problematika

Při hledání na internetu jsem našel jeden podobný projekt, zabývající se přestavbou RC modelu s použitím více motorů a jeho řízením. Projekt je na GitHubu viz [Nak14].
Bohužel poslední příspěvky v tomto projektu jsou již 3 roky staré a tedy předpokládám, že tento projekt již neběží. U tohoto projektu jsem se inspiroval mechanickým řešením.
Koncept řízení se v automobilovém průmyslu začíná objevovat ve větší míře v několika posledních desetiletí. V tomto odvětví jsou použity systémy ASR, ABS i active steering. Všechny systémy jsou komfortními systémy starající se o větší stabilitu vozidla na vozovce, případně o zlepšení jízdních vlastností jako je např. zkrácení brzdné dráhy vozidla pomocí systému ABS.
Když se zaměřím na RC komunitu, tak zde se aktivní řízení dá také dohledat. Např. systém Traxass stability management (TSM). Další informace o tomto systému lez nalézt v [Tra18]. Tento systém se velmi podobá systému active control, který jsem použil já ve svém projektu. Systém pomocí zpětné vazby z gyroskopických měření upravuje akční zásahy na natočení předních kol.
Dalším typem systému je tzv. Drift asistent. Jde například o HPI D-BOX drift assistant. Více o tomto systému je v [Rac18a]. Systém pomáhá pilotovi RC modelu se dostat do smyku a v tomto smyku se udržet, aniž by se auto roztočilo kolem vlastní osy. Tento systém funguje na stejném principu jako již popsaný systém TSM. Rozdíl je v nastavení konstant použitých v regulační

smyčce, které mají vliv na celkové chování systému.
Některé mnou implementované řídící zákony vycházejí z těchto systémů (např. zpětná vazba na změnu směru od gyroskopu jako v HPI D-BOX), další řídící zákony jsem však ještě nikde nenašel použity (např. systém torque vectoring).

Kapitola 2

Mechanika modelu

Mechanické řešení modelu vypracoval s mou součinností kolega Ing. Helmich z fakulty strojní.
Z důvodů jednoduchosti a časové nenáročnosti jsem se rozhodl - po konzultaci s Ing. Helmichem - použít již hotový model a ten přepracovat pro naše potřeby. Tyto mechanické úpravy prováděl Ing. Helmich.
Použité mechanické řešení modelu se dá rozdělit na následující seznam, kde jsou jednotlivé komponenty krátce představeny.

2.1 Šasi

Jeden z požadavků na šasi bylo použití silničního vozidla. Po konzultaci s Ing. Helmichem a Ing. Hromčíkem jsem se rozhodl koupit kvůli nenáročnosti přestavby model od společnosti X-Ray, Pan-Car GT 1/10. Na tento model Ing. Helmich vytvořil nové díly a upravil stávající pro použití v tomto projektu. Hlavní částí, která se měnila byla zadní část modelu. Zde se udělala zcela nová zadní část modelu s přímým připojením motorků na poháněná kola. To z toho důvodu, že neexistují RC modely, které by měly více než jeden motor a přímý pohon kol.

2.2 Převody motorů

Kvůli volbě motorů (další podrobnosti v 3.1) byly vybrány převody 8:1. Pinion, který se musel nalepit na motory byl k dostání pouze mosazný s 12 zuby. K tomuto pinionu byl zvolen pastorek s 96 zuby. Jsou to experimentálně zvolené hodnoty, které se v praxi ukázaly jako správně zvolené. Pro další experimentování s modelem se poměr dá změnit výměnou pastorku, která je relativně nenáročná.

2.3 3D výtisk

Pro zabudování elektroniky do RC modelu jsem vytvořil 3D model, jehož projekt je v příloze. Potom jsem model vyexportoval a předal Ing. Helmichovi, který přidal otvory pro uchycení elektroniky a celý 3D model vytiskl na 3D tiskárně. Celý výkres jsem vytvořil v programu Blender 3D, který je hojně používán komunitou 3D tiskařů. V tomto programu jsem vymodeloval i celé okolí modelu, které je též součástí výkresu v příloze.

2.4 Vysílačka a přijímač

Byl vybrán obyčejný vysílač spolu s přijímačem TTX300 od firmy TACTIC pracující na frekvenci 2,4 GHz. Výstupem z tohoto přijímače je PWM signál. Tento PWM signál má frekvenci 50 Hz. Informace je přenášena v pulzu, který má šířku 1 - 2 ms. Obě krajní hodnoty odpovídají krajním hodnotám použitých akčních členů, které jsou přijímačem ovládány. Pro servo odpovídá hodnota 1 ms plnému natočení přední nápravy doleva a 2 ms plnému natočení přední nápravy doprava. Střední hodnota 1,5 ms odpovídá natočení rovně. U motorů odpovídá střední hodnota klidovému režimu, 1 ms odpovídá plnému přiloženému napětí pro pohyb vzad a 2 ms odpovídají plnému přiloženému napětí pro pohyb vpřed.

2.5 Akumulátor

Baterie byla kvůli jmenovitému napětí motorů vybrána dvoučlánková LiPol baterie o kapacitě 4000 mAh. Jeden článek LiPol baterie má hodnotu 3,7V. Jmenovité napětí této baterie je $2 * 3,7V = 7,4V$.

Kapitola 3

Senzory, telemetrie a motory

Pro řízení modelu byly vybrány dva typy senzorů. Jeden inerciální senzor pro měření akcelerací a úhlových rychlostí a dva proudové senzory pro měření proudu v motorech. Dále byly použity dva telemetrické moduly HC-12. Byly implementovány modelářské BLDC motory.
Podrobnosti o těchto komponentech, jejich výběru a implementaci do modelu jsou diskutovány v této kapitole.

3.1 Motory

Výběr motorů se přizpůsobil modelářskému prostředí a to hlavně z důvodu ceny, protože průmyslové motorky dle našich potřeb byly o řád dražší, než motorky používané RC komunitou a běžně dostupné. Naopak o modelářských motorech bylo velmi těžké něco zjistit.
Pro potřebné výpočty byl použit program RC Calc (více informací o tomto programu v [Cal16]). K zjištění potřebných parametrů motorů, jako je výkon a točivý moment potřebný pro jízdu modelu, jsem použil již existující RC model HPI sprint 2 flux viz [Rac18b], který byl dostupný v programu RC Calc a u kterého byly potřebné informace, jako je váha modelu, KV použitého motoru a jmenovité napětí baterky (dvoučlánková LiPol nebo tříčlánková LiPol).
Tento model má převody z motoru na všechna čtyři kola přivedeny v poměru 1:8,3. Bohužel se mi nepodařilo zjistit přesnou váhu tohoto modelu, nicméně obdobné modely mají váhu okolo 1,5 kg. Budu tedy i s rezervou uvažovat 1,4 kg (vypočtené motory vyjdou nakonec u mé konfigurace spíše silnější). Tento

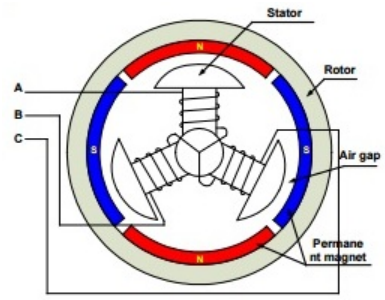
RC model je napájen dvěma články baterie LiPol (3,7V na článek) a tedy dosáhne teoreticky nejvyšších otáček u motoru 43 660 RPM. Použitý motor je HPI Flux Vector 5900 kv. Naneštěstí se mi nepodařilo najít jeho výkon, ale u podobných motorů (BLDC s kv okolo 5900) se výkon pohybuje přibližně mezi 350 - 450W (např. [Hob16] nebo [Hob18a]). Budu tedy předpokládat výkon okolo 350W. Bohužel nemám k dispozici ani momentovou charakteristiku motorů. Musel jsem si tedy přibližně nějakou vypočítat. To znamená, že kroutivý moment (pro 7,4V) který tento motor dává je roven

$$M = P/\omega \tag{3.1}$$

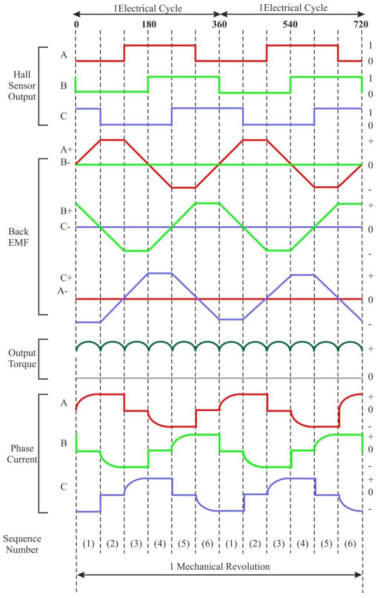
$$\omega = 2\pi f \tag{3.2}$$

$$f = RPM/60 \tag{3.3}$$

Z toho mi vyjde, že $M = \frac{30P}{\pi \cdot rpm}$, kde rpm pro 7,5 volt je 43 660. Výsledné M je $M = 0,077Nm$. Když toto ještě převedu v poměru 1: 8,3 vyjde $M = 0,63Nm$. To je $45 \cdot 10^{-5}\frac{Nm}{g}$. A tato konfigurace by měla dodat dostatečný výkon pro auto o hmotnosti cca 1400g v plné rychlosti.
Z tohoto vycházím při návrhu na konfiguraci 2 motorů pro váhu vyvíjeného RC modelu cca 1500g, protože sice vyjmu diferenciál, pastorky a hřídel, ale přidám jeden motor a jeden regulátor. Jeden motor váží kolem 50g a jeden regulátor kolem 60g. Tedy celkem bude mé přidané příslušenství o 110g těžší a vyjmutý diferenciál a hřídel by mohly (dle mého odhadu) vážit kolem 100 - 150g. Takže se dostanu k celkové váze cca 1500g. Z předchozích úvah vychází, že na tuto váhu by měl být moment cca $M = 45 \cdot 10^{-5} \cdot 1500 = 0,675Nm$. Tento údaj vydělím dvěma a dostanu moment potřebný na jedno kolo, resp. jeden motor. To je $M_p = 0,3375Nm$.
Chtěl-li bych zachovat počet otáček na kolo (5 260rpm), budu potřebovat alespoň motor s výkonem 185W. Protože na našem modelu je použit převod 1:8, musím tyto momenty tímto poměrem zmenšit. Pro poměr 1:8 je potřebný moment $M_{p8} = M_p/8 = 0,0423Nm$. Pro motor popsaný v [Hob18b] mi z předchozích rovnic vychází počet otáček na motoru při výkonu 182W a napětí 7,4V $RPM = \frac{30P}{\pi \cdot M} = 41108$. Pro provoz modelu bude stačit, když motor bude splňovat, že jeho kv je menší než nebo rovno $\frac{41108}{7,4} = 5555$. Čím více se budeme tomuto limitu blížit, tím zvýšíme maximální rychlost modelu. Čím bude kv menší, tím bude mít model větší kroutivý moment. Tento motor má kv 5300 tudíž je dostatečný.
Bohužel tento motor se dal sehnat pouze v Německu, tak jsem se snažil vybrat nějaký podobný v ČR. Nicméně požadavek na to, aby se u daného motoru uvedl i výkon byl u motorů používaných v RC modelech nesplnitelný. Žádný takový jsem nenašel. Zavedl jsem tedy předpoklad, že motory pro modely o velikosti 1:18 s kv 5300 a méně budou použitelné; případně se změní převodový stupeň aby motory fungovaly lépe.
Motory byly s ohledem na výše zmíněné skutečnosti vybrány modelářské BLDC od firmy Castle Creations. Jsou to motory pro RC modely velikosti



Obrázek 3.1: Principiální zapojení BLDC motoru. Převzato z [Con18].



Obrázek 3.2: Průběhy napětí a proudů v BLDC motorech. Převzato z [GRVSGG10].

1:18, tedy o velikost menší než model, ve kterém jsou použity. Tyto motory mají 5300 KV a jsou čtyřpólové. K těmto motorům byly koupeny i modelářské regulátory Sidewinder Micro 2, které regulují přiložené napětí podle vstupní střídy modelářského PWM signálu.
Pro měření otáček je výhodně využito toho, že motory jsou BLDC, jelikož jsou napájeny většinou třífázovým vedením. V případě RC motorů je to pravidlo. Regulátor tohoto motoru vždy přiloží napětí mezi dvě fáze a potom pomocí zpětné indukce napětí měří polohu rotoru a podle této informace rozhodne o tom, na které fáze přiloží napětí v dalším cyklu. Tohoto využívám při měření otáček motoru, jak je popsáno v 3.3.2. Otáčky motoru jsou přímo úměrné frekvenci proudu v jednotlivých fázích motoru (závisí na počtu pólů motoru). Principiální zapojení BLDC motoru je na obrázku 3.1. Průběhy proudů a napětí jsou vidět na obrázku 3.2. Informace pro měření otáček pomocí proudu jsem čerpal z [GRVSGG10].

V několika následujících podkapitolách se věnuji popisu obou typů senzorů použitých v projektu, jejich výběrem, zapojením a komunikací s nimi.

3.2 Inerciální měřící jednotka

Pro použití v tomto projektu jsem volil jednoduchý MEMS (Micro Electro-Mechanical Systems) akcelerometr s kapacitním snímačem zrychlení. Alternativou pro tento typ akcelerometru by byl např. piezoelektrický akcelerometr, který však není běžný, a proto jsem jak z důvodů dostupnosti materiálů, tak z důvodu ceny, volil MEMS akcelerometr.
Gyroskop jsem volil také MEMS, jednak kvůli tomu, že se vyrábějí čipy, které mají v sobě zabudované tříosé akcelerometry společně s tříosými gyroskopy, tak i kvůli tomu, že tento typ gyroskopu je pro tuto aplikaci dostačující. Alternativou pro tento typ gyroskopu by byl např. gyroskop s optickými vlákny (FOG - Fiber optics gyroscope). Oproti gyroskopům s optickými vlákny je MEMS gyroskop levnější a menší, avšak méně přesný a citlivý.

3.2.1 Princip činosti MEMS Akcelerometru

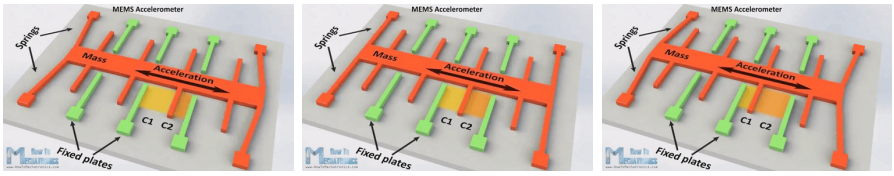
Jak struktura MEMS akcelerometru na čipu vypadá, je principiálně zobrazeno na obrázku 3.3. Oranžová část je uchycena k čipu pomocí pružin na krajích a zelená část je pevně připojena k čipu. Měření je prováděno fyzikálně na principu druhého Newtonowa zákona, který je matematicky vyjádřen jako

$$F = ma,$$

kde F je síla, m je hmotnost a a měřené zrychlení. Senzor má známou hmotnost. Pružiny, na kterých je tato hmotnost uchycena mají definovanou tuhost k. Z této tuhosti se podle vzorce

$$F = kx,$$

kde x je výchylka, vypočítá síla F. Výchylka x se měří pomocí změny kapacity mezi statickými a pohybujícími se elektrodami. Další podrobnosti jsou k nalezení v [SLS18] nebo v [Rip07].



Obrázek 3.3: Principiální činnost MEMS akcelerometru. Převzato z [Gyf17].

3.2.2 Princip činosti MEMS Gyroskopu

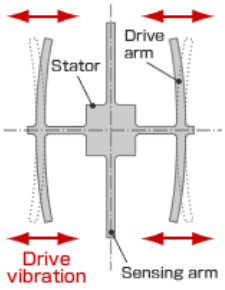
Principiální funkce mechanické struktury vibrujícího MEMS gyroskopu je na obrázku 3.4. Gyroskop pracuje fyzikálně na principu Coriolisovy síly, pro kterou platí vzorec

$$\boldsymbol{F_c} = -2m\boldsymbol{\omega} \times \boldsymbol{v},$$

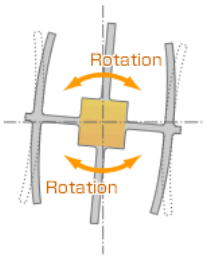
kde $\boldsymbol{F_c}$ je Coriolisova síla, m je hmotnost, $\boldsymbol{\omega}$ je úhlová rychlost a $\boldsymbol{v}$ je posuvná rychlost. Vnější struktury jsou rozkmitávány (viz obr. 3.4a) pomocí elektrostatické síly a tím mají posuvnou rychlost $\boldsymbol{v}$. Když přijde rotace $\boldsymbol{\omega}$ (viz obr. 3.4b) tak na celou strukturu začne působit Coriolisova síla. Protože rozkmitávané struktury se pohybují vždy opačnou rychlostí, působí na obě strany Coriolisova síla stejné velikosti (pohybují se stejně rychle a mají stejnou hmotnost), ale opačné orientace. Toto způsobí, že celý senzor se rozvibruje i ve vertikálním směru (viz obr. 3.4c). Tímto jsou vyvolány vibrace středové struktury, jejichž výchylka je měřena (viz obr. 3.4d). Další podrobnosti jsou k nalezení v [SLS18] nebo v [Rip07].

3.2.3 Výběr IMU jednotky

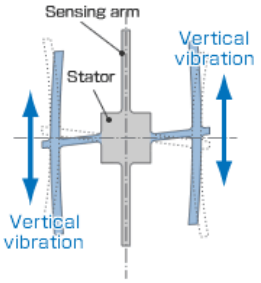
Protože jsem hledal tříosý Akcelerometr (měření všech tří ortogonálních směrů akcelerace) a tříosý gyroskop, rozhodoval jsem se mezi následujícími IMU (inertial measurument unit) jednotkami:



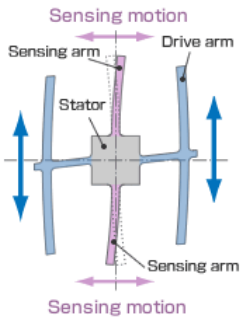
(a) : Klidový stav



(b) : Směr rotace



(c) : Rotace způsobuje vibrování ve vertikálním smměru



(d) : Rotace způsobuje měřenou vibraci

Obrázek 3.4: Principiální činnost MEMS Gyroskopu. Převzato z [Cor18].

- Srovnání některých parametrů tříosých akcelerometrů je v tabulce 3.1.
- Srovnání některých parametrů tříosých gyroskopů je v tabulce 3.2.

	MPU6050	ADXL335	ADXL345
Rozsah [g]	±2, ±4, ±8 a ±16	± 3	±2, ±4, ±8 a ±16
Komunikační rozhraní	I^2C	analogové výstupy	I^2C, SPI
Dynamický rozsah [Hz]	2 - 500	0,5 - 550	0,05 - 1600

Tabulka 3.1: Srovnání akcelerometrů. Informace byly čerpány z [Inv13], [Dev09a] a [Dev09b].

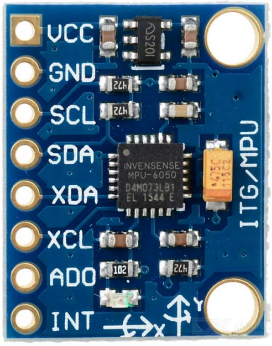
	MPU6050	L3G4200D	ITG3200
Rozsah [10^2 °/s]	±2.5, ±5, ±10 a ±20	±2.5, ±5 a ±20	±20
Komunikační rozhraní	I^2C	I^2C, SPI	I^2C

Tabulka 3.2: Srovnání gyroskopů. Informace byly čerpány z [Inv13], [ST10] a [Inv10]

Z akcelerometrů mi nejlépe vycházel čip ADXL345. Je to z toho důvodu, že má největší dynamický rozsah a SPI rozhraní schopné komunikovat při hodinové frekvenci až 10 MHz. Z gyroskopů mi nejlépe vycházel L3G4200D. Má dobře volitelný rozsah a jeho velkou výhodou je také sběrnice SPI, která umožňuje mnohem rychlejší komunikaci než sběrnice I^2C. Bohužel jsem nenašel žádný již hotový modul, který by integroval oba dva čipy. Nakonec jsem tedy i s přihlédnutím k tomu, že je velké množství materiálů dostupných na internetu, volil čip MPU6050, který v sobě integruje jak tříosý akcelerometr, tak tříosý gyroskop. Je také dostupný v 6 DOF (degree of freedom) modulu běžně používaným Arduino komunitou. Existují na něj už i hotové knihovní rozhraní například v [Row18].

3.2.4 IMU jednotka s čipem MPU6050

Po mechanické zástavbě, která byla provedena ve spolupráci s Ing. Helmichem, jsem připojil jednotku, jež je na obrázku 3.5. Jelikož Arduino má pouze 3,3V tolerantní piny, zvolil jsem napájení jednotky MPU 3,3V. Toto napájení jsem přivedl na pin VCC. Na ping GND jsem zapojil elektrickou zem z jednotky Arduino. Jednotka MPU6050 umožňuje komunikaci a sběr dat z dalších senzorů pomocí druhé sběrnice I^2C, kde tato jednotka působí jako master. Data uložená ze sběrnice je poté možno vyčíst po hlavní sběrnici z k tomu určených registrů. Hlavní I^2C sběrnice je na pinech SDA/SCL a pomocná pro sběr dat je na pinech XDA/XCL. Na piny SCL a SDA jsem tedy zapojil I^2C sběrnici jednotky Arduino. Piny XDA a XCL jsem nechal nezapojené. Dále jednotka MPU6050 umožňuje pomocí pinu AD0 měnit adresu na I^2C sběrnici. Tento pin má připojený v modulu pull down rezistor, a tudíž nepřipojen bude v logické nule. Tento pin jsem nechal nepřipojený. Pin INT umožňuje jednotce MPU6050 dělat interrupty, které jsou programovatelné přes vnitřní registry.



Obrázek 3.5: Fotka připojené jednotky s čipem MPU6050. Převzato z [Ele15].

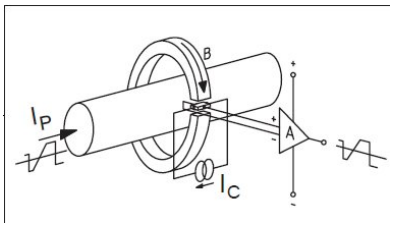
Tento pin jsem nechal nepřipojený. Detaily lze najít v [Inv13].

3.3 Proudový senzor

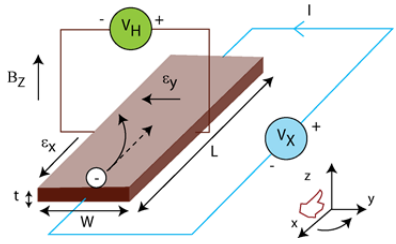
Proudový senzor je využit pro měření proudového odběru motorů. Nabízely se možnosti kontaktního a nebo bezkontaktního řešení. Protože motory jsou BLDC, tak v každé fázi by měl po filtraci filtrem typu dolní propust probíhat periodický proud. To je způsobeno tím, že motorové vinutí je induktivního charakteru a je na něj přiloženo periodické napětí, které je přímo úměrné otáčkám motru. Jeden z hlavních důvodů pro bezkontaktní měření je ten, že bezkontaktním měřením bych nevnášel žádnou další asymetrii do jednotlivých fází. Jako řešení bezkontaktního měření proudu se jevil senzor na principu proudového transformátoru nebo Hallovy sondy. Kontaktní měření je např. měření pomocí proudového bočníku.
Rozhodl jsem se použít bezkontaktní měření, kvůli jeho výhodám.
Měření pomocí proudového transformátoru však vyžaduje převážně vlastní PCB a proto jsem se rozhodl použít měření Hallovou sondou. Pro tento typ měření již existují hotové moduly, které se dají použít a jsou i finančně dostupné. Detaily o těchto senzorech a měření s nimi lze najít v [Haa03].

3.3.1 Hallova sonda

Principální zapojení Hallovy sondy je na obrázku 3.6. Hallova sonda je napájena konstantním zdrojem proudu a pomocí Hallova jevu snímá magnetickou indukci, která je vyvolána protékajícím proudem a pomocí magnetického

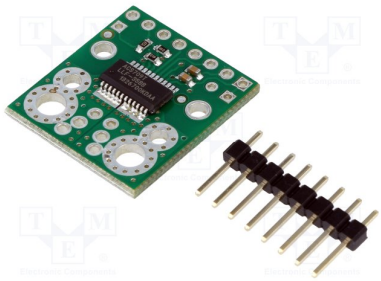


Obrázek 3.6: Principiální funkce Hallov sondy. Převzato z [Por].



Obrázek 3.7: Hallův jev. Převzato z [NPT16].

obvodu přivedena k Hallově sondě. Výstupem této sondy je zesílené Hallovo napětí.
"Hallův jev je vznik elektrického napětí příčně ve vodiči, jímž podélně protéká elektrický proud, který je odchylován magnetickým polem kolmým ke směru proudu." (Převzato z [WIK16]).
Je zobrazen na obr. 3.7.



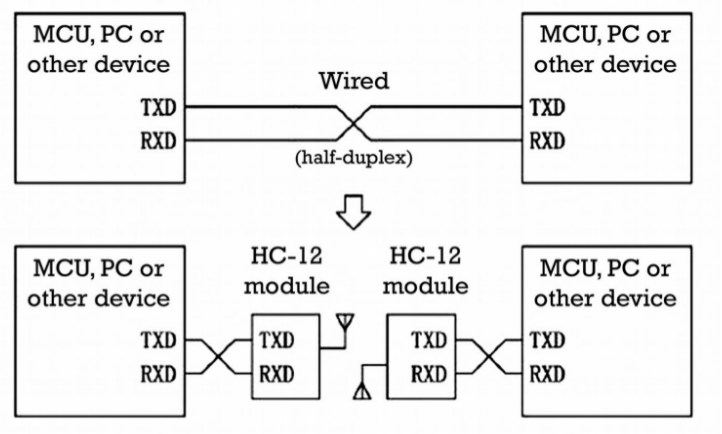
Obrázek 3.8: Modul proudové senzoru POLOLU-2199. Převzato z [Rep18].

3.3.2 Modul proudového senzoru POLOLU-2199

POLOLU-2199 je modul postavený na čipu ACS709, což je proudový senzor s Hallovou sondou. Je zobrazen na obr. 3.8. Tento senzorový modul splňoval všechny požadavky na proudový senzor použitý v modelu, a to sice:

- špičkově měřit alespoň 40 A
- měřit i stejnosměrnou složku
- měřit minimálně do 50 kHz

Modul POLOLU má připojen pin GND na GND pin Arduina. Dále má analogový výstup, což je trochu jeho nevýhoda. Může docházet k elektromagnetickému rušení a tím větší šumovosti měření. Pro měření je využit AD převodník Arduina. Vstup do AD převodníku v Arduinu nesmí překročit 3,3V, proto výstup ze senzoru také nesmí překročit 3,3V. Výstupem je analogové napětí v rozmezí 0 - napájecí napětí (3,3V), které přivedené na pin VCC . Výstupní pin VIOUT je přiveden na spojovací desku (viz 3.5), kde je vyfiltrován a přiveden na Arduino AD převodník. Dále pin VZCR je výstupní analogové napětí, které je rovno nulovému proudu. Tedy když se bude VIOUT rovnat VZCR, neteče senzorem žádný proud. Tento signál je zpracován stejným způsobem jako signál z VIOUT a přiveden na AD převodník jednotky Arduino.
Další piny jsou již nezapojeny. Jsou to piny FAULT a FAULT_EN, které slouží např. k detekci většího proudu než takového, který odpovídá napětí na pinu VOC. Poslední nevyužitý pin je pin FILT, který slouží k filtraci výstupu. Připojením filtračního kondenzátoru se nastavuje šířka pásma výstupu senzoru. Filtrace je v tomto případě vyřešena až u oddělovacího zesilovače na jednotce Interconnect, proto není třeba využívat tento pin.



Obrázek 3.9: Schéma komunikace mezi počítačem a RC modelem při použití modulu HC-12. Převzato z [Lá15].

3.4 Telemetrie

Požadavky na systém telemetrie jsem si nastavil následující:

- rychlost komunikace by byla vhodná kolem 100 kb/s
- bezdrátová komunikace je podmínkou
- dosah bezdrátové komunikace alespoň 1 km (z důvodu provozu modelu ve venkovním prostředí)

Z tohoto jsem vyšel při hledání telemetrického systému a vyšel mi systém popsaný na obrázku 3.9. Tento systém se skládá z dvou bezdrátových komunikačních modulů HC-12. Moduly pracují na nosné frekvenci 433 - 473 MHz podle kanálu. Mají možnost volby jednoho ze 100 kanálů. Podle [Lá15] dokáží komunikovat až na vzdálenost 1,8 km s baud ratem 1200 bps. Moduly pracují pouze polovičním duplexem. Uživatelská zařízení se připojují pomocí sériového komunikačního rozhraní (UART). Pro uživatelská zařízení jsou během provozu neviditelná. Připojení k RC modelu jsem provedl následovně: napájení modulu je připojeno k 3,3V, GND pin na GND, RX pin modulu HC-12 na TX pin modulu Arduino a TX pin modulu HC-12 na RX pin modulu Arduino. Informace byly čerpány z [Lá15]. Zde je také možné najít detailnější informace.
Dále má modul HC-12 možnost přejít do konfiguračního rozhraní a to pomocí pinu SET.

Baud rate ve vzduchu [bps]	Baud rate sériového portu [bps]
5000	1200 2400
15 000	4800 9600
58 000	19 200 38 400
236 000	57 600 115 200

Tabulka 3.3: Přenosové rychlosti modulu HC-12. Převzato z [Lá15]

3.4.1 Konfigurace

Modul HC-12 má následující možnosti konfigurovatelných módů:

- FU1
- FU2
- FU3
- FU4

Základním módem je mód FU3. Je to mód, ve kterém je modul ve full-speed režimu. Podle nastaveného baud rate seriového portu je nastaven baud rate v bezdrátovém přenosu podle tabulky 3.3.

V tomto módu je možno nastavit také vysílací výkon a to až 20 dBm (100 mW). Dále je nastavitelná délka vysílaného slova, parita a počet stop bitů. Tento mód spotřebovává 16 mA. Všechny tyto parametry se nastavují v konfiguračním režimu.
Do tohoto režimu se jednotka dostane tak, že pin SET je nastaven na nulu a jednotka je zresetována. Poté je možno pomocí příkazů definovaných v [Lá15] nastavit tyto parametry.
Mód FU1 je mód s nízkou spotřebou energie, spotřebovává pouze 3,5 mA. Kvůli nízké spotřebě je zde omezení v dosahu. FU1 pracuje pouze do vzdálenosti 200 m.
Mód FU2 má extrémně nízkou spotřebou - spotřebovává pouze 80 μA. Tento mód má také jako mód FU1 dosah do 200m. Oproti módu FU1 se liší tím, že může pracovat pouze do baud rate 4800 bps. Také mezi posílanými pakety musí být alespoň 1s klidový stav.
Mód FU4 je mód pro komunikaci na maximální vzdálenost. FU4 zvládá

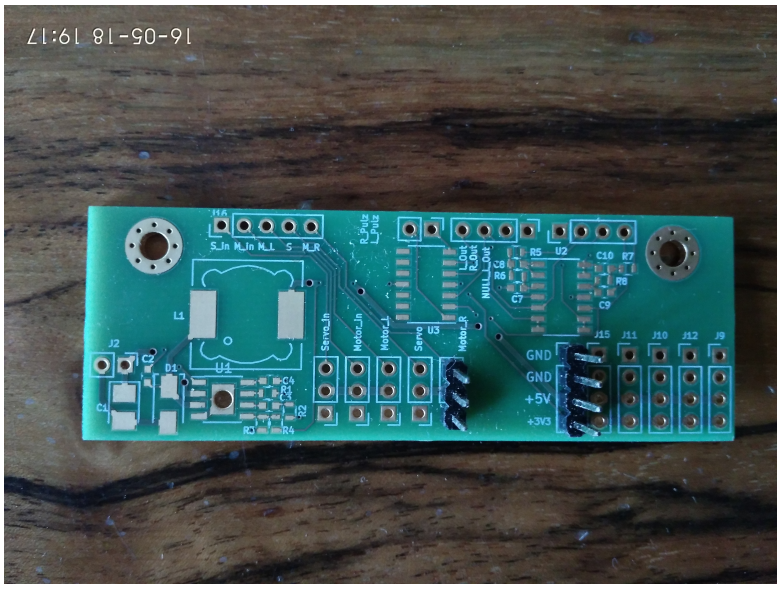
komunikaci až na 1,8 km (podle [Lá15]) při baud rate 1200 bps. Mezi pakety musí být alespoň 2s klidový stav.

3.4.2 Použité nastavení

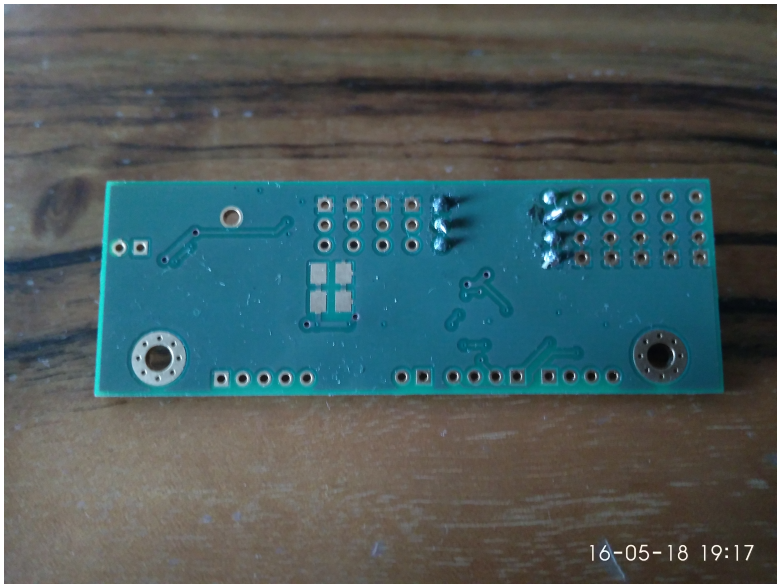
Pro použití v tomto projektu jsem zvolil základní mód FU3. Kvůli zabezpečení bezproblémového přenosu jsem nastavil kontrolu sudou parity. Dále jsem nastavil baud rate na 115,2 kbps. Počet přenášených bitů jsem zvolil 8 s jedním stop bitem.

3.5 Spojovací deska

Spojovací deska je PCB vyrobená v rámci projektu u firmy PragoBoard. Všechna data a schémata jsem vytvořil pomocí programu KiCAD. Je to open source program určený právě pro vývoj PCB. Motivací pro tuto výrobu byl fakt, že všechny moduly potřebovaly být spolu spojeny a napájeny. Schéma PCB je v příloze C. Celý projekt je přiložen k práci.
Deska je složena z konektorů, které jsou připojeny z přijímače. Další konektory jsou zapojeny do akčních členů - motorů a serva. Ty jsou napájeny z obou regulátorů patřících k motorům. Z těchto regulátorů vystupuje napětí přibližně 5,7V. Proto jsem dále na desku přidal buck regulator (step down), který reguluje vstupní baterkové napětí na 5V, ty jsou následně přivedeny na Arduino. Těchto 5V se poté pomocí LDO regulátoru na jednotce Arduino sníží na 3,3V, které jsou zpět přivedeny na jeden z několika napájecích pinů, jimiž je spojovací deska vybavena. Je zde i blok s operačními zesilovači, který dělá filtraci a zároveň díky vlastnostem operačního zesilovače má téměř nulový výstupní odpor a tedy se může již napřímo zapojit do AD převodníku Arduina (viz 4.2.8). Použitý filtr se jmenuje Sallen-Key filtr. Zlomová frekvence ω_0 obvodu se vypočítá podle vzorce $\omega_0 = \frac{1}{RC}$, kde $R = R_6 = R_8 = 3,3k\Omega$ a $C = C_7 = C_8 = 10n$. Zlomová frekvence tohoto filtru typu dolní propust je tedy přibližně 5 kHz, protože $f_0 = \frac{\omega_0}{2\pi} = 4825Hz$. Tuto frekvenci jsem experimentálně stanovil na základě průběhů z měření proudu.
Fotka DPS z přední strany je na obrázku 3.10 a ze spodní strany na obrázku 3.11.



Obrázek 3.10: Svrchní strana spojovací desky.



Obrázek 3.11: Spodní strana spojovací desky.

Kapitola 4

Řídící jednotka

Pro aktivní řízení RC soupravy je použita μC (mikrokontrolérová) jednotka Arduino Due. Zde jsou naimplementovány jak komunikace, nastavení senzorů a telemetrie, tak i řídící zákony.

4.1 Výběr jednotky

Jednotku jsem vybíral z již dostupných řešení. V budoucnu navrhuji celé řešení včetně připojení senzorů a telemetrie zhotovit na jednu DPS desku kvůli kompaktnosti, elektromagnetickému rušení a z dalších důvodů. Zpřehlední se tím celé zapojení a ubude mnoho kabelů. Výběr jednotky jsem udělal z následujících možností:

- **NUCLEO-L476RG** - Vývojový kit od společnosti ST. Je postaven na procesoru STM32L476RGT. Tento procesor je dostatečně výpočetně výkonný pro aplikaci v RC soupravě. Má frekvenci jádra až 80 MHz, 1 MB FLASH a 128 kB SRAM. Je z řady Low power procesorů od firmy ST. Má dostatečný počet periférií potřebných v projektu jako jsou např. AD převodník, I^2C controller, DMA... Jeho předností je i nízká cena. Na výběr jsou i jiné Nucleo Kity, které ale mají buď moc velké rozměry nebo jsou méně výpočetně zdatné.
- **Arduino Due** - Open hardware platforma původem pocházející ze severoitalských univerzit. Je podobná vývojovému kitu Nucleo z předchozího

bodu. Platforma je postavena na procesoru ATSAM3X8E od společnosti Atmel. Procesor má takt jádra až 84 MHz, 512 kB FLASH a 100 kB SRAM. Dále má plno periférií včetně těch, které jsou pro tento projekt potřebné (AD převodník, I^2C komunikační rozhraní,...). Velmi velkou výhodou tohoto řešení je velká dostupnost materiálů a příkladů použití Arduino komunitou na různých fórech.

- **Teensy 3.2** Alternativou k jednotkám Nucleo a Arduino Due je jednotka Teensy, která je rozměrově o mnoho menší (35x17mm vůči 82x70mm u Nuclea). Jednotka je postavena na procesoru MK20DX256VLH7 od firmy NXP. Procesor má frekvenci jádra 72 MHz, 256 kB FLASH a 64 kB SRAM. Má také ostatní potřebné periférie jako μC z předchozích bodů. Výhodou modulu Teensy je jeho velikost i cena. Velkou nevýhodou je však fakt, že se prodává pouze v USA.
- **PIXHAWK flight controller** - V užším výběru byla i RC komunitou používaná letová řídící jednotka PIXHAWK. Nicméně tato jednotka svojí cenou přesahující 200 USD a stejně tak i nedostatečným počtem vstupů a výstupů nevyhovuje použití v tomto projektu.

Zvažoval jsem i několik dalších jednotek, ale ty po bližším přezkoumání nebyly výpočetně dostatečné. Nakonec jsem se rozhodl pro Arduino Due a to ze dvou důvodů. Jednak to byla jedna z výkonnějších jednotek, dále měla nejvíce periférií a druhý - hlavní - důvod byl ten, že k této jednotce existuje velké množství materiálů a příkladů na různých fórech používaných komunitou Arduino.

4.2 Arduino Due

Arduino Due je Open hardware platforma. Všechny její podklady jsou na jejich stránce. Arduino Due je postaveno na čipu ATSAM3X8E. Tento čip má jádro ARM Cortex-M3. Jeho blokové schéma je na obrázku 4.1. V následujících sekcích popíši vybrané periférie využité v projektu. Všechny následující informace o periféríích pocházejí z [Cor15].

4.2.1 NVIC

NVIC - Nested vector interrupt controller, je periférie obsluhující přerušení jak z IO pinů tak z vnitřních periférií. NVIC má tabulku registrovaných zdrojů

přerušení a k nim odpovídající ISR (Interrupt Service Routine). Každému přerušení je přiřazena priorita. V případě chodu běžného programu a příchodu přerušení je pozastaven běh programu a je spuštěna odpovídající ISR. V případě obsluhy ISR a příchodu přerušení s vyšší prioritou je ISR s nižší prioritou pozastavena a je spuštěna ISR s vyšší prioritou. Když ISR s vyšší prioritou doběhne, vrátí se procesor k vykonávání ISR s nižší prioritou a následně k běžnému chodu programu. V ATSAM3X8E je použita inverzní priorita, tedy čím nižší je číslo nastavené jako priorita, tím má dané přerušení prioritu větší.
V tomto projektu bylo NVIC použito na mnoha místech jako jsou např. komunikace přes I^2C, vyčítání dat z ADC, opakované přerušení od TC a jiné.

4.2.2 PDC - Periferal DMA Controller

Každá periférie v ATSAM3X8E má vlastní DMA kanál (V případě Full Duplex periférií jsou to kanály dva). DMA se zabývá přenosem dat bez využívání procesorových instrukcí, tedy výrazně šetří procesorový čas.
PDC je periférie, jak už z názvu vyplývá, starající se o přenos dat mezi periférií jako např. I^2C a RAM pamětí na čipu. Vše se děje pomocí vnitřní sběrnice AHB jak je vidět na blokovém schématu 4.1.
Tuto periférii stačí nastavit a ona sama podle nastavení na konci přenosu vyvolá přerušení nebo nastaví příslušný bit v registrech patřících k této periférii. Přenos může být jak half duplex tak i full duplex
Periférie je vybavena spojením s NVIC (Nested vector interrupt controller) periférií pro obsluhu přerušení jako je např. konec přenosu, konec vysílání, chyba přenosu aj. Pro použití periférie je třeba nastavit příslušné přenosové registry (RHR, THR, TCR, RCR,...) a zápisem do Control registru spustit přenos/příjem.
V tomto projektu je PDC použito při provozu telemetrie a k odečítání dat ze senzorů. Používají ho tedy periférie ADC, TWI a USART.

4.2.3 PIO

PIO - Parallel input output controller - je periférie ovládající výstupní piny. V případě procesoru ATSAM3X8E je většina pinů multiplexována. Některé piny mají totiž přístup k periférii A i B. K tomu ještě mohou být klasickými IO piny (vstupně - výstupními).
V tomto projektu je několik vstupních pinů, které jsou většinou nastaveny tak, aby při změně logické úrovně generovaly přerušení. Popsaný mechanismus je použit např. pro měření PWM, kterou generuje přijímač.

4.2.4 TWI

TWI - Two wire interface - je periférie implementující také některé I^2C standardy. Omezím se pouze na jejich popis, protože jsou použity v tomto projektu. I^2C je synchronní sběrnice typu master-slave. Obsahuje dva signály a to sice SCK a SDA. SCK je hodinový signál, který má ve standardním módu frekvenci 100 kHz. Celý standard je hardwarově standartizován výstupně jako open-drain. Tedy výstupy mohou sběrnici pouze nastavit do logické nuly. Na návrat do logické jedničky slouží pull-up rezistory, které jsou již implementovány na jednotce Arduino Due. Další detaily lze najít v standardu [Sem14].
K podporovaným I^2C standardům patří také Fast Mode Speed, běžící na hodinové frekvenci 400 kHz. Tohoto módu jsem využil při komunikaci Arduina s čipem MPU6050. Všechna čtení z této periférie fungují pomocí PDC popisovaného v 4.2.2 v součinnosti s přerušeními popisovanými v části 4.2.1.

4.2.5 USART

USART - Universal synchronous asynchronous receiver transmitter - je klasické sériové rozhraní. V ATSAM3X8E jsou tyto periférie celkem tři. Tato periférie jde nastavit pro mnoho standartizovaných komunikačních protokolů včetně asynchronního sériového, který je použit v tomto projektu.
Tato periférie je nastavena v součinnosti s PDC rozhraním pro vysílání měřených dat a příjem příkazů do jednotky. Je zde také využito procesorových přerušení. Tato periférie je nastavena na rychlost 115,2 kbps. Je použito 8 datových bitů, sudá parita a jeden stop bit. Vše se nastavuje v registrech této periférie. Více o této periférii lze najít v [Cor15].

4.2.6 TC

TC - Timer counter - je periférie pro měření času nebo jako počítadlo. Jednotka Arduino obsahuje 3 HW čítače. Jedná se o 32 bitové čítače, pro které je možno nastavit různý zdroj signálu k čítání.
V tomto projektu je využita tato periférie pro měření času. Jelikož měřím maximálně 20 ms (při měření přijímaného signálu z příjmače viz 2.4), tak mi stačí zvolit čítaný signál s maximální rychlostí (což je MasterClock/2 = 84/2 MHz = 42 MHz), abych dosáhl co největší přesnosti změřeného času. Tedy pro představu 20 ms představuje $20\,000 \cdot 42 = 840\,000$ připočtených jednotek

k hodnotě čítače. To je asi $0,2\%$ z maximální hodnoty čítače (maximální hodonta je 2^{32}).

4.2.7 PWM

PWM - Pulse width modulation, je signál často používaný k řízení výkonu DC výkonových komponent jako je např. motor nebo pole LED diod. ATSAM3X8E má 8 makrobuněk schopných produkovat nebo měřit samostatně PWM signál. V RC modelech je to signál používaný k přenosu informací viz 2.4.
V tomto projektu je PWM signál použit pro řízení obou použitých motorů a serva. Pro generoování PWM signálu stačí nastavit hodinový signál, od kterého daný PWM signál je odvozený, a poté je potřeba ještě nastavit vnitřní registry pro porovnání s PWM čítačem, které zaručují přechod z log. 1 do log. 0 v průběhu jedné periody.

4.2.8 ADC

Poslední použitou periférií je ADC (Analog digital converter). Slouží k převodu analogových do digitálních hodnot. μC ATSAM3X8E má 1 AD převodník, který je multiplexován do 16 AD kanálů. Tento AD převodník je schopný převádět s rychlostí 1 MSa/s. Převádí s rozlišením 10 nebo 12 bitů. Protože tento AD převodník nemá oddělovací zesilovač a podle [Cor15] může odebírat v diferenčním módu až 9 mA, bylo potřeba před připojením proudových senzorů k ADC vložit oddělovací buffer, který však zároveň slouží jako low-pass filtr viz kapitola 3.5.
Podle teorie platí, že čím větší je OSR (oversampling ratio - koeficient převzorkování), tím menší je efektivní hodnota šumu ve vzorkovaném pásmu (detaily v [Ved04]). Z toho mi vyplývá, že čím více bude vzorků za sekundu, tím se dostanu na hodnoty s menším vlivem šumu. Oproti tomu jsem experimentálně zjistil, že když je vzorků moc, jejich následné zpracování trvá déle než jednu periodu hlavní řídící smyčky.
AD převodník se v tomto projektu používá pro určení proudu protékajícího motory.

4.3 Software

V této části práce se věnuji základnímu popisu implementovaného software v jednotce Arduino. Celý software je psaný v jazyce C++ ve vývojovém prostředí poskytnutém a vyvíjeném přímo firmou Atmel. Použité prostředí je Atmel Studio ve verzi 7. Celý software vzešel z knihoven Arduino a postupně, jak už implementace těchto knihoven nestačila (z důvodů vyšších časových nároků nebo použití periférie PDC) jsem předělával jednotlivé funkce prací přímo s registry popisovanými v [Cor15]. Do Atmel Studia jsem přešel velice brzy, protože Arduino IDE (vývojové prostředí vyvíjené a podporované komunitou Arduino) je pro větší projekty hlavně kvůli organizaci kódu nevhodné. Kvůli migraci kódu do Atmel Studia jsem využil freeware trial doplňku k Atmel Studiu - Visual micro, který umožňuje používat veškeré Arduino knihovny v rámci Atmel Studia. Všechny knihovny a jejich dokumentaci lze najít na [Ard18].
Celý projekt jsem také uploadoval na GitHub. Repozitář je v [Vos18]. Software je rozdělen na následující části a podčásti

- Konfigurace - v konfigurační části jsou nastaveny všechny periférie, které jsou popsány výše. Je zde implementována i inicializace dalších funkčních bloků kódu potřebných k běhu hlavní smyčky.

- Hlavní smyčka - smyčka, která se periodicky opakuje po počáteční konfiguraci systému. Dá se rozdělit na dvě části:

  - Časově kritický program - zde je část kódu, která je časově kritická a je potřeba ji opakovaně pouštět po přesně určených časových úsecích. Tato část kódu má také přednost před časově nekritickou komunikací. Toho je docíleno pomocí periférie TC (popsáno v 4.2.6). Jako časově kritická komunikace je implementována např. řídící smyčka.

  - Časově nekritický program - zde je implementována ta část kódu, u které není důležité, kdy se provede. Jsou to např. pomocné debugovací výpisy nebo telemetrická data. Tato část kódu je prováděna v hlavní programové smyčce, mimo všechna přerušení.

- Obsluha přerušení - do této skupiny patří periodické zpracování událostí od různých periférií. Je zde např. periodické vyčítání registrů z MPU6050, které je realizováno pomocí PDC periférie.

4.3.1 Konfigurace

Úvodní funkcí, která se zavolá po startu μC je funkce setup(). V této funkci se následně zavolají další funkce, které nastavují různé periférie. Jsou to funkce

- **void set_up_adc()** - funkce na inicializaci AD převodníku. Nejprve se zde pomocí ASF (Atmel Software Framework) spustí v PMC (power management controller) napájení AD převodníku. Poté se periférie restartuje a v CMR (Controll mode register) nastaví celý mód AD převodníku. AD převodník je spouštěn pouze pomocí SW příkazu a ne po HW události. Dále se zde nastaví 12 bitové rozlišení. Hodiny AD jsou nastaveny podle rovnice
$$ADCClock = MCK/((PRESCAL + 1) * 2),$$
kde MCK má frekvenci 84 MHz a PRESCAL je nastaven na 37, a tedy hodiny AD převodníku jsou 1,11 MHz. S touto hodnotou probíhá konverze s frekvencí $f_c = 25kHz$.
Byly povoleny kanály 1 - 4. To je kvůli snímaní čtyř signálů pocházejících z proudových senzorů (VIOUT_1, VZCR_1, VIOUT_2, VZCR_2). Pak se zde nastavuje NVIC. Pro vyčítání dat jsem nastavil PDC periférie a přerušení. Jediné přerušení, které jsem povolil bylo ENDRX - tedy přerušení vyvolané PDC dokončením přenosu z periférie do SRAM. Velikost přenášených dat jsem zde zvolil 1000. Je to hodnota odpovídající časovému úseku $t_s = \frac{1000}{25000} = 40ms$. Toto je dostatečně dlouhá doba na zjištění frekvence změřených period. S touto periodou můžu teoreticky měřit frekvence až $f = 1/t_s = 25Hz$. Kdybych ještě chtěl stanovit OSR, stanovil bych ho následovně: Vím, že signál je filtrován na $f_p = 5kHz$. Tedy
$$OSR = \frac{f_c}{f_p} = 5.$$
Toto je experimentálně určená hodnota. Když nastavím větší hodnoty, je jejich následné zpracování v řídící smyčce moc dlouhé a smyčka nestačí doběhnout do začátku dalšího intervalu.
Tato nastavení byla udělána kvůli digitálnímu měření frekvence, která se ve výsledku ukázala jako velice nepřesná a tato měření byla nahrazena těmi popisovanými v sekci 4.3.4. Po těchto nastaveních se spustí AD převodník v režimu freerun. Freerun je režim, který spustí další konverzi ihned, jakmile ta první skončí.

- **void initialize_TWI()** - funkce na inicializaci TWI rozhraní. TWI rozhraní je v Master módu. Je využit Fast mode ze standardu I^2C s hodinovou frekvencí 400 kHz. Je zde také použito vyčítání pomocí PDC periférie. Na konci přečtení registru je vyvoláno přerušení typu ENDRX. Při inicializaci této periférie je zároveň spuštěno napájení v jednotce

MPU6050. Zde je nastavený použitý rozsah gyroskopů a akcelerometrů. Pro oba dva jsem vybral největší šířku pásma, tedy pro gyroskop 256 Hz a pro akcelerometr 250 Hz. Maximální rozsah pro gyroskop jsem volil nejmenší možný, tedy $\pm 250°/s$. Tento maximální rozsah jsem si experimentálně potvrdil - během testovací jízdy nebyl dosažen (při stabilních jízdách). Dále jsem nastavil maximální rozsah akcelerometru na $\pm 2g$. To jsem také experimentálně ověřoval a nebyl dosaženy ani 3/4 rozsahu akcelerometru.

- **void set_up_telemetry()** - tato funkce se zabývá základním nastavením periférie USART3, která je napojená na jednotku HC-12 a komunikuje tedy na dálku (viz sekce 3.4). V této funkci je zapnut USART3 v asynchronním módu. Je použito nastavení s 8 bity, jedním stop bitem a sudou paritou.

- Dalšími funkcemi je zapnuta a inicializována periférie TC. Dále jsou zde také spuštěny a nastaveny přerušení na GPIO pinech, kde jsou zapojeny jak výstupy od vysílače, tak výstupy ze spojovací desky, které slouží k měření otáček.

4.3.2 Komunikace

V různých částech SW je použita komunikace s vnějšími přístroji. Hlavní komunikační jednotkou je HC-12, nicméně pro debugovací účely je použita knihovní funkcionalita komunity Arduino. Tato třída užitá pro debugovací zprávy se jmenuje SerialUSB. Debugovací komunikace probíhá přes USB rozhraní a je vesměs použita jednosměrně z μC do počítače. Pro standardní komunikaci včetně např. telemetrických údajů jsou aplikovány USART3 a dvě jednotky HC-12.
Pro standardní komunikaci bylo implementováno několik ze základních syscall funkcí, a to sice funkce *_write* a *_read*. Obě funkce jsou použity standardní knihovní funkcí *printf*, která je poté s výhodami (jako je např. formátování) užívána.
Pro komunikaci jsem zavedl jednoduchý textový protokol, kde jednotka Arduino čeká na příkaz ze strany počítače a na ten zareaguje - komunikace je typu master-slave. Tento protokol má následující strukturu:

- Příkaz - jeden byte na rozlišení příkazů

- Data - data náležící k příkazu

- Konec - končící znak zprávy - v hex 0x78 (znak x).

Pomocí těchto příkazů jsou v zásadě implementovatelné jakékoliv příkazy. V jednotce jsou implementovány celkem tři druhy příkazů

- Změny koeficientů v řídící smyčce - jsou zde nastavována různá zesílení řídící smyčky. Těchto příkazů bylo využito při experimentálním ladění různých řídících zákonů.
- Změny v architektuře řídící smyčky - pomocí těchto příkazů se zapínají nebo vypínají různé zpětné či dopředné vazby, atd.
- Telemetrie - pomocí těchto příkazů se zapíná vysílání telemetrických zpráv. Telemetrická data jsou vysílána jako text v ASCII kódu a jsou oddělena řádky. V řádku jsou jednotlivé sloupce odděleny pomocí čárek. První je vždy vysílán čas a poté různá měřená data jako např. rychlost otáčení kol, proudy v motorech, zrychlení, apod.

4.3.3 Hlavní smyčka

Tato část kódu je rozdělena na dvě části - časově kritický a časově nekritický program. Časově kritický program je popsán v kapitole 5.
Dále je zde časově nekritická část kódu. Je to procesorová hlavní smyčka. Tato smyčka má nejnižší prioritu a všechna přerušení jsou proto obsloužena dříve. V této části kódu jsou periodicky nastavovány procesy pro vyčítání senzorů. Je zde volána funkce $IMU\_read\_next()$, která má za úkol nastavit nové čtení dat z registrů MPU6050 vždy, když předchozí čtení je již ukončeno. Také je v tu kontrolován příjem znaků z řídícího počítače (přes periférii USART3). Tyto přijaté znaky jsou poté zpracovány a vyhodnoceny jak je popsáno v 4.3.2. Poslední část kódu se zabývá posíláním telemetrických dat v případě, že je odesílání nastaveno. Toto se děje pouze jednou za stanovený čas, aby nebyl zahlcen komunikační kanál. Experimentálně jsem periodu posílání telemetrických dat nastavil na 60 ms.
Když použiji menší periodu, tak tím, že jednotka HC-12 umí pouze Half-Duplex komunikaci, se moje zpráva z počítače nedostane do Arduina a jednotka se tím stává neovladatelnou. Tato perioda zasílání dat závisí na velikosti telemetrických dat (čím více je posíláno dat, tím menší může být opakovací perioda). V současné podobě je zasíláno 6 údajů. Jsou to:

- Otáčky levého kola
- Otáčky pravého kola
- inerciální rychlost - integrovaná data z akcelerometru v ose y (osa pohybu auta dopředu a dozadu).

- Akcelerace v ose Y
- Proud v levém motoru
- Proud v pravém motoru

4.3.4 Zpracování měřených signálů z proudového senzoru

Signál z proudového senzoru je nejdříve vyfiltrován a oddělen pomocí analogového filtru na spojovací desce (viz 3.5). Tento signál je přiveden na AD převodník jednotky Arduino.
Výstupní signál a signál nulového proudu jsou přivedeny na operační zesilovač, který pracuje jako komparátor (viz C). Výstup tohoto komparátoru je přiveden na digitální piny jednotky Arduino, které jsou nastaveny na přerušení a tím se měří frekvence průběhu proudu v motoru.
Tento údaj obsahuje šum a proto je zpracováván jednoduchým algoritmem popsaným dále. V případě, že přerušení přijde dříve než je nastavena minimální perioda signálu z motoru, je vyhodnoceno jako šum a není započítáno. Tato kontrola probíhá v obou půl-periodách signálu. Výstupní signál je i přes tyto kontroly nadále šumový a je proto filtrován jednoduchým digitálním filtrem typu dolní propust druhého řádu. Konečná data jsou i přesto dost zatížená šumem, nicméně se už dají použít.
Data z AD převodníku jsou odečtena - nulový signál je odečten od výstupního signálu proudového senzoru. Následně jsou průměrována za nastavenou periodu z nastavených vzorků (koeficienty programu). Tyto data jsou pak ještě filtrována také digitálním filtrem typu dolní propust druhého řádu.
Na grafu 4.2 je změřená akcelerace RC modelu. Z 4.2b je vidět, že průběh proudu přibližně odpovídá očekávaným hodnotám. Naopak na grafu 4.2a je vidět, že měření otáček obsahuje výraznou šumovou složku.

4.3.5 Struktury použité při implementaci zákonů řízení

Pro jednodušší implementaci jednotlivých zákonů řízení jsem si připravil infrastrukturu, která mi zajišťuje dostatečnou modularitu skládání jednotlivých bloků. Vytvořil jsem strukturu signal_block, která obsahuje dva členy. Těmi jsou **void**∗ pointer_on_data a pointer na funkci **double** (∗serial_signal_output)(**double**, signal
Toto je generický vzor pro všechny použité bloky. V současné době jsou implementovány dva bloky. Těmi jsou blok Gain a z_function.
Blok Gain, jak už název napovídá, je zesilovací blok. Blok z_function je implementace diskrétní přenosové funkce.
Dále je implenetována následující struktura:

```
typedef struct {
                signal_block** blocks;
                int size_used;
                int size;
        } serial_signal_path_t;
```

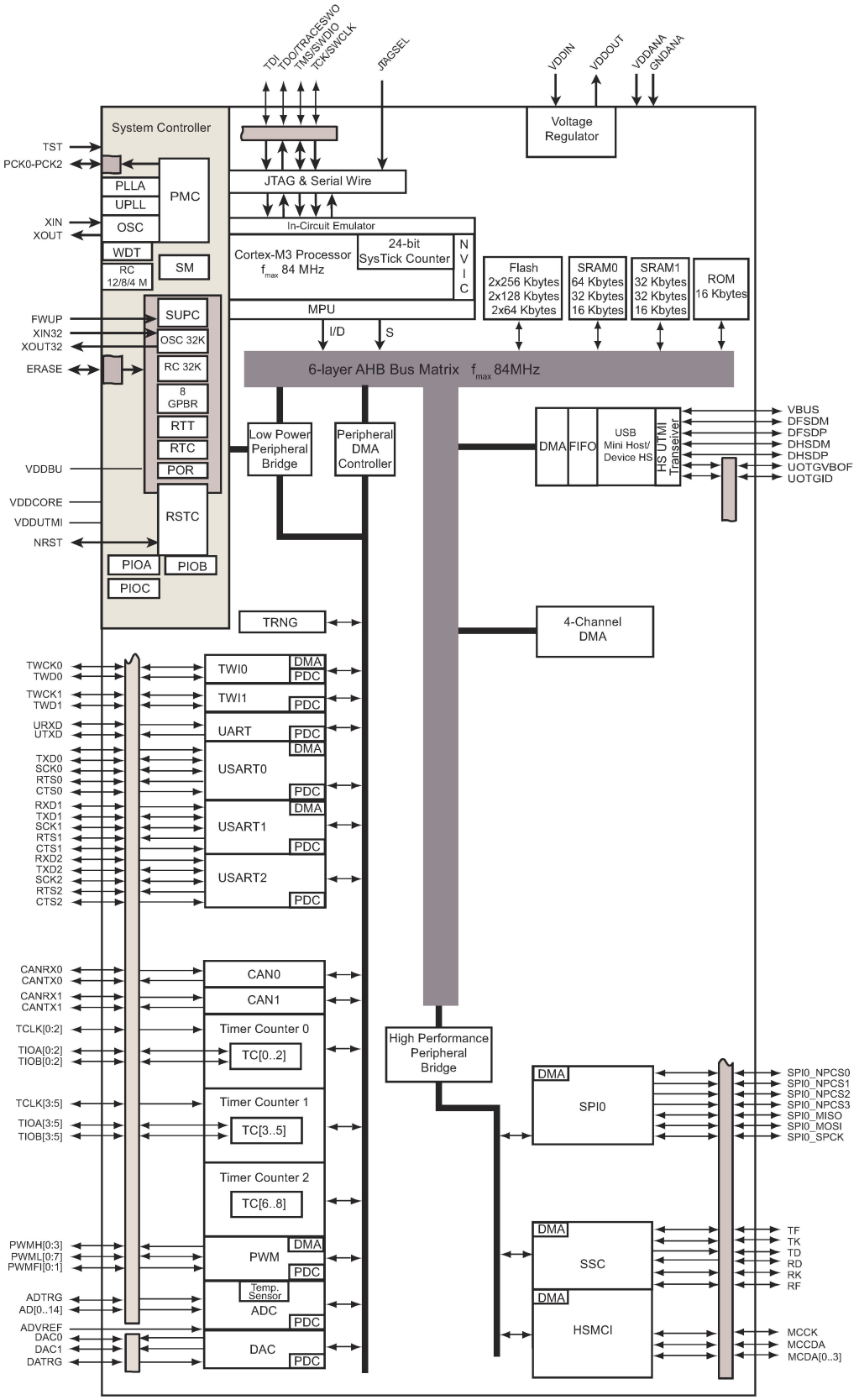
Tato struktura představuje sériové řazení jednotlivých signálových bloků za sebe. Celý proces používání této infrastruktury se skládá ze dvou částí, které ukáži na vzorových blocích kódu. Nejprve je potřeba dotyčné sériové řazení bloků nakonfigurovat. To je ukázáno např. na jednoduché přenosové funkci znázorňující filtr typu DP:

```
#define GYRO_FEEDBACK_GAIN 0.013
serial_signal_path_t gyro_fb;
char register_low_pass()
{
        z_function* discrete = (z_function*) malloc(
                sizeof(z_function));
        if (discrete == NULL)
                return 0;
        discrete->denominator_size = 3;
        discrete->nominator_size = 3;
        discrete->denominator = (double *)calloc(
                sizeof(double), discrete->denominator_size);
        discrete->nominator = (double *)calloc(
                sizeof(double), discrete->nominator_size);
        if (discrete->denominator == NULL || discrete->nominator == NULL)
                return 0;
        double* den = discrete->denominator;
        double* nom = discrete->nominator;
        den[2] = 0.006981;
        den[1] = - 0.1671;
        den[0] = 1;
        nom[0] = 0.21;
        nom[1] = 0.4199;
        nom[2] = 0.21;
        signal_block* low_pass = (signal_block*)malloc(
        sizeof(signal_block));
        if (low_pass == NULL)
                return 0;
        Transfer_function(discrete, low_pass);
        register_member(low_pass, &gyro_fb);
        return 1;
}
```

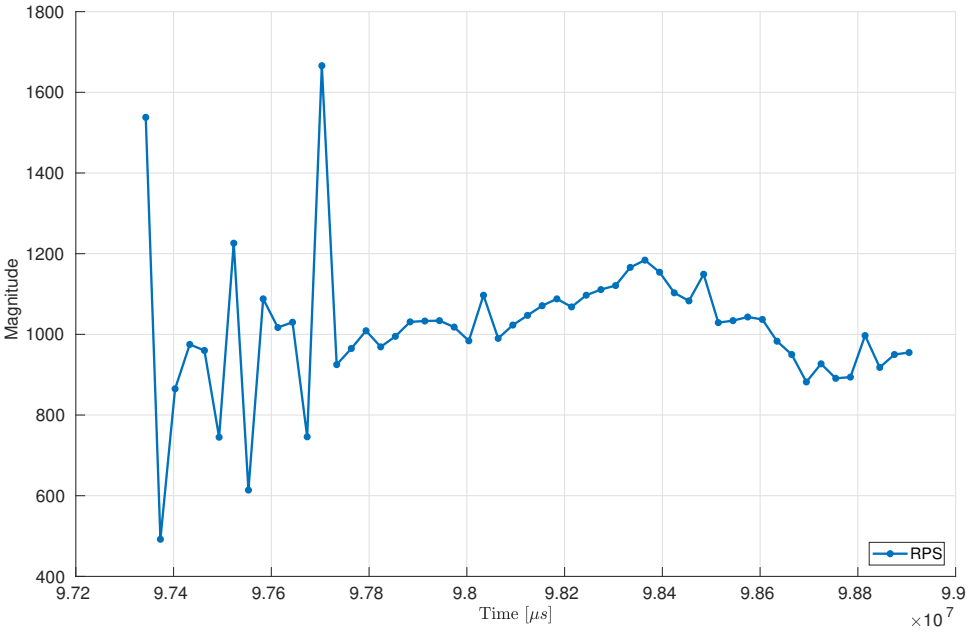
Kde je nejprve nastavení bloku low_pass a poté je zavolána metoda register_member, která přijímá parametry typu signal_block* a serial_signal_path_t*. Tato metoda přidá daný blok předaný v prvním argumentu do řetězce sériového řazení bloků předaného v druhém argumentu. Následné použití vypadá v kódu takto:

```
out_steer = inputs.steering - get_output(gyro_z, &gyro_fb);
```

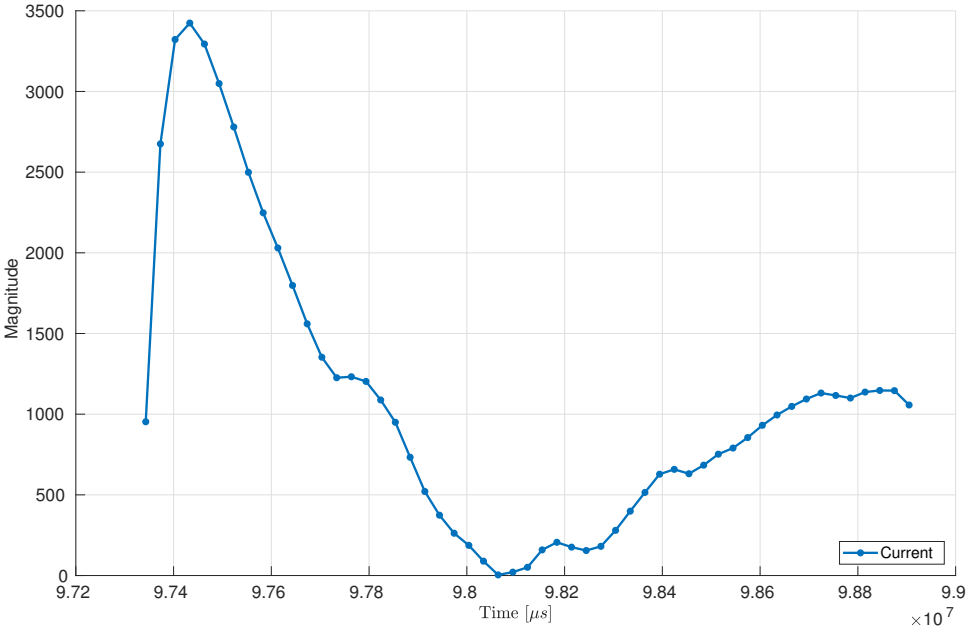
Z této ukázky je stěžejní použití metody get_output, která prezentuje použití celé infrastruktury. Tato metoda přijímá jako první argument vstup do sériového řazení signálových bloků a jako druhý argument přijímá odkaz na strukturu uchovávající celé řazení bloků. Návratová hodnota této funkce je výstup z celého sériového řazení bloků.
Tato infrastruktura je velmi flexibilní a jednoduchá na použití. Velmi jednoduše se dají naimplementovat další potřebné bloky.



Obrázek 4.1: Blokové schéma čipu ATSAM3X8E. Převzato z [Cor15]



(a) : Vypočtené otáčky ze změřených dat



(b) : Změřený proud

Obrázek 4.2: Výstupy měření na proudových senzorech

Kapitola 5

Zákony řízení: návrh, experimentální ladění a validace

Celý systém je z povahy μC diskrétní. Pomocí periférie TC je nastavena perioda na 3846 μs. Tato perioda odpovídá frekvenci 260 Hz, což je více než maximální opakovací frekvence jednotky MPU6050, jež je senzorem s nejdelší periodou dostupnosti nových dat. Pro návrh zákonů řízení jsem se inspiroval systémy, které jsou použity buď v RC modelářském světě nebo u normálních osobních vozidel.
Všechny experimentální ladění a následná testování, která jsou níže popsána, jsem prováděl v tělocvičně ČVUT Fakulty strojní v budově A na Karlově náměstí. Jsou zde kluzké parkety, na kterých je velmi těžké ovládat RC model bez zapnutých řídících systémů.
Systém má dva vstupy - ovládání serva a motorů. Vše se ovládá pomocí PWM signálu popsaným v 2.4. Výstupy z přijímače jsou tedy časové údaje hovořící o délce PWM pulsů. Výsledným výstupem z přijímače po zpracování jeho signálu je šířka PWM pulzu, která může nabývat hodnot 1000 - 2000 μs. Pro linearizaci vstupů jsem zvolil úvodní kalibrační sekvenci, která sbírá definovaný počet vzorků a poté z nich určí střední hodnotu, která je vždy odečítána od obdržených časových údajů. Dále se také zaznamenává maximální a minimální přijatá hodnota. Tento údaj slouží ke škálování přijatého signálu. Výpočet probíhá podle kódu

```
int vstup = input_time - stred_vstupniho_signalu;
int result;
if (vstup > 0)
{
        float scale = (float)MAX_VYCHYLKA_AKCNIHO_ZASAHU)/
                (float)(max_vstupni_sig - stred_vstupniho_signalu);
        result = vstup*scale;
```

```
}
else if (vstup < 0)
{
        float scale = ((float)MAX_VYCHYLKA_AKCNIHO_ZASAHU)/
                (float)(stred_vstupniho_signalu - min_vstupni_sig);
        result = vstup*scale;
}
else
        result = 0;
return result;
```

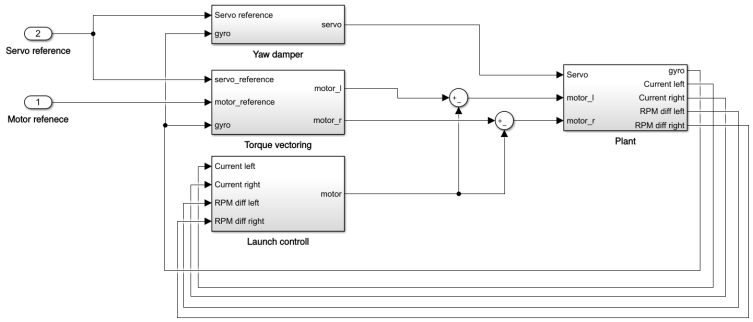
Kde $MAX\_VYCHYLKA\_AKCNIHO\_ZASAHU$ je definována jako makro s hodnotou 500. Dále *stred_vstupniho_signalu* je průměr výstupu z přijímače za trimovací dobu (měřeno v μs). *input_time* je doba pulzu měřená v μs. Tím je dosaženo toho, že vstupy jsou lineární a mají maximální hodnotu $\pm$500. Také se tímto přístupem řeší různé maximální hodnoty a drobné odchylky u různých typů přijímačů a vysílačů.
Se stejnou myšlenkou jsou linearizovány také výstupy na akční členy - servo a motory. Je zde však nelinearita v limitech signálů. Může být nastaveno maximálně $\pm$500. O převod tohoto lineárního signálu do PWM ovládacího signálu se stará soubor *OutputMapper.cpp*.

Do RC modelu jsem v návaznosti na model uvedený v [Kie05] implementoval následující systémy:

- **Torque vectoring** - řídící zákony, které podporují a stabilizují RC model v zatáčení pomocí akčních zásahů na motorech.
- **Active steering** - je technologie převzatá z leteckého průmyslu. Tento zákon řízení je popsán v [Nel98].
- **Launch control** - je protiprokluzový systém podobný tomu, který lze nalézt v osobních automobilech. V automotive se tento systém nazývá ASR.

Pro testování těchto systémů byly navrženy testovací jízdy. Na předem definovanou dobu D se jako reference do systému přivedou definované vstupy. Navrhnul jsem následující testovací scénáře:

- **Akcelerace** - testování zrychlení RC modelu. Natočení přední nápravy se nechá nulové a přivede se předem definovaný ovládací vstup na motory.



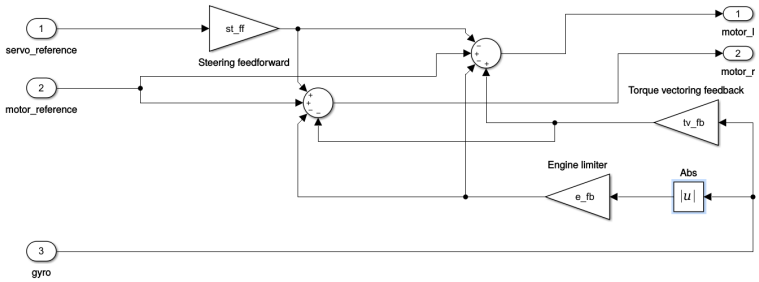
Obrázek 5.1: Schéma celé regulační smyčky

- **Zatáčení** - testovací scénář pro průjezd zatáčkou. Nastaví se ovládací vstup na motory a během celého testu je maximální natočení přední nápravy.
- **Průjezd zatáčkami** - testovací scénář, který testuje průjezd několika po sobě jdoucími zatáčkami. Jízda je rozdělena do pěti úseků, z nichž každý trvá 20% testovací doby D. Na motory je nastaven definovaný ovládací vstup a natočení přední nápravy se mění vždy po každém úseku na opačný úhel natočení. Na referenci pro servo se tedy přivádí periodický signál s amplitudou 300.

Schéma celého systému je na obrázku 5.1. Regulační systém má dva vstupy, které jsou napojeny na přijímač. Jsou to ovládání motoru a serva. Systém má 5 výstupů, další veličiny, které jsou zajímavé pro měření, se dají přidat. Výstupy jsou měření gyroskopu, proudů protékajících oběma motory a otáčky obou motorů.

5.1 Torque vectoring

Tento systém je jak pro automotive tak pro RC modelářství relativně nový pojem. Pro použití v RC modelech není možné takový systém použít, protože modely jsou vždy poháněny pouze jedním motorem. V klasických osobních automobilech nastává stejný problém použitelnosti tohoto systému. Zajímavý začne být tento koncept až pro elektromobily, které mohou mít také přímo ovladatelná kola jednotlivými motory (v elektromobilu může být více motorů). Architektura tohoto systému je na schématu 5.2. Vstupy do tohoto systému jsou reference pro servo, reference pro motor a výstup z gyroskopu kolem osy z RC modelu. Výstupy z tohoto systému jsou akčními zásahy na motory (levý a pravý). Je zde implementovaná statická zpětnovazební smyčka



Obrázek 5.2: Schéma systému Torque vectoring

s koeficientem engine feedback limiter. Tato smyčka ubírá akční zásah na motor v závislosti na hodnotě z gyroskopu. Další smyčkou je statická zpětnovazební smyčka nazvaná torque vectoring feedback. Zde je implementována podobná funkcionalita jako v engine feedback limiter s tím rozdílem, že tu není absolutní hodnota z gyroskopu. Tedy motory zabírají proti směru otáčení. Poslední implementovaná regulační smyčka je statická přímovazební. Vhodnou konstantou je zesílena reference pro servo. Tento řídící zákon usnadňuje zatáčení.

5.2 Launch control

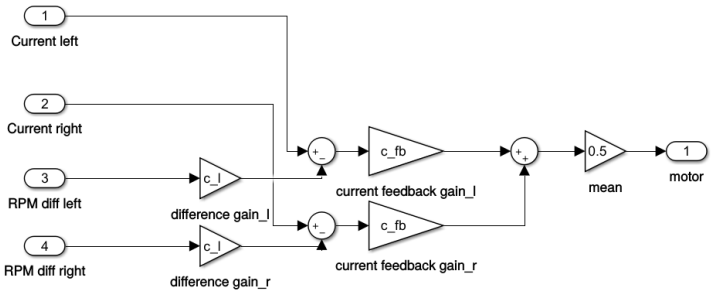
Launch control je systém, který je již dlouhou dobu používán v osobních automobilech pod názvem ASR (Anti-Slip regulation). Detaily jsou dostupné v [Kie05]. Tento systém zabraňuje prokluzu kol, způsobeným přílišným výkonem motorů. Ve chvíli, kdy začne kolo prokluzovat, tento systém pomocí zpětné vazby tlumí přiložený výkon.
Implementace v RC modelu není pomocí klasických vstupů a výstupů, kterými jsou otáčky na nepoháněné a poháněné nápravě. Kvůli chybějícímu měření otáček předních kol jsem přistoupil k experimentálnímu nastavení architektury regulátoru podle schématu 5.3.
Jak je na tomto schématu vidět, tak vstupními veličinami je rozdíl změřených otáček a proud v motoru. To je dvakrát pro obě kola. Kvůli vysokému šumu měření otáček byl ještě přidán filtr typu dolní propust s diskrétní přenosovou funkcí

$$H(z) = \frac{0,001224z^2 + 0,002447z + 0,001224}{z^2 - 1,86z + 0,865}$$

Tato funkce byla vypočítána pomocí MATLABu a funkce c2d z přenosové funkce

$$H(s) = \frac{\omega_0^2}{(s + \omega_0)^2},$$

kde $\omega_0 = 2\pi f$ a f je zlomová frekvence jednoduchého filtru DP druhého řádu. Zlomovou frekvenci jsem u tohoto filtru volil experimentálně na 5 Hz.



Obrázek 5.3: Schéma systému Launch control

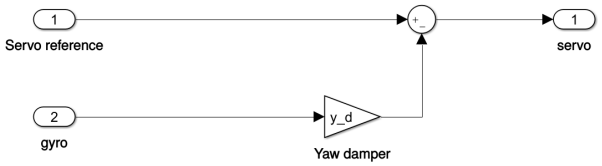
Tento filtr vnáší zpoždění do měření, avšak byl nutný kvůli vysoké šumovosti měření.

5.3 Active steering

Active steering je systém používaný již dlouhou dobu v leteckých regulačních systémech. V tomto odvětví se systém nazývá yaw damper. O tomto použití je možné se dočíst v [Nel98]. Nicméně v nedávné době se začal používat také v RC modelech. Na trh byl uveden systém TSM - Traxass stability management, na kterém, ačkoliv není open source, je vidět, že výsledky jimi prezentované na videích, které jsou k nalezení v [Tra18], odpovídají systémům yaw damper v leteckém průmyslu.
Dle těchto videí to vypadá, že je použit jak pouhý tlumič rychlé dynamiky, tak i systém na regulaci úhlu natočení - v leteckém průmyslu nazývaném Heading hold and select. Heading hold and select jsem v RC modelu neimplementoval. Dalším podobným projektem je také již zmiňovaný HPI D-BOX, který se chová jako active steering, mnou implementovaný v RC modelu.
Schéma celého systému je na obrázku 5.4. Vstupy do tohoto systému jsou reference pro servo a výstup gyroskopu. Výstup je pouze jeden a to akční zásah na servo. Výstup gyroskopu přenásoben vhodnou konstantou působí pomocí akčního zásahu proti otáčivému pohybu. Experimentálně jsem zjistil, že při velkém roztočení motorů se celé auto začne chvět a třást. Na tento pohyb tento systém také reaguje a vnáší šum do řízení. Toto jsem vyřešil zařazením filtru typu DP, který jsem naimplementoval podobně jako v 5.2. Spojitá přenosová funkce byla experimentálně stanovena na

$$H(s) = \frac{98596}{(s+314)^2}$$

.



Obrázek 5.4: Schéma systému active steering

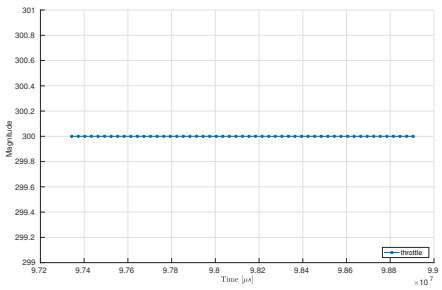
5.4 Validace sytémů

Pro validaci jsem navrhnul testovací scénáře popsané výše. Testoval jsem vždy 3 varianty:

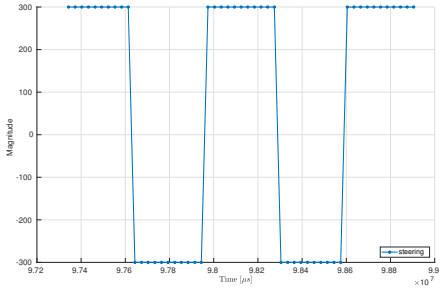
- **Vše zapnuté** - všechny mnou navržené regulační systémy jsou v provozu
- **Bez launch control** - jsou zapnuté pouze systémy active steering a torque vectoring.
- **Bez zásahu** - všechny regulační systémy jsou vypnuté a vstupy jsou bez zásahu převedeny do akčních členů.

Pro scénář akcelerace jsem dosahoval podobných výsledků pro všechny varianty - dokud byla jízda pro všechny varianty stabilní. Pro scénář zatáčení jsem dosahoval výrazně lepších výsledků pro varianty *Vše zapnuté* a *Bez launch control* oproti variantě Bez zásahu. Avšak nejzajímavější testovací scénář byl Průjezd zatáčkami. Pro tento scénář jsem změřil některé výstupy, které jsou dále prezentovány.
Bohužel, s přihlédnutím k povaze ovládání motorů pomocí regulátorů (ovládají přiložené napětí na motory), výsledky silně závisely na stavu nabití akumulátoru. Změřil jsem v krátkém sledu za sebou, kde tedy předpokládám stejné podmínky (především stejná úroveň nabití akumulátoru), všechny tři varianty zapnutí regulačních systémů. Vstupy, pro které jsem navržené systémy testoval, jsou na grafu 5.5. Pro porovnání výsledků zde uvedu výstupy z gyroskopu pro všechny varianty. Porovnání je vidět na grafu 5.6. Pro variantu *Vše zapnuté* je to průběh nazvaný all systems active, pro variantu *Bez launch control* je to průběh nazvaný without launch control a pro variantu *Bez zásahu* je to průběh open loop.
Jak je vidět z grafů, tak se všemi aktivními systémy se RC model neroztočil kolem vlastní osy (nebyl nestabilní) a dokázal projet scénářem. Již pro variantu *Bez launch control* se RC model občas roztočil kolem vlastní osy (launch control nemá tak velký vliv na stabilitu viz 5.5). Záviselo dost na

lokálních podmínkách - tedy jaká byla zrovna nerovnost povrchu, přilnavost povrchu atd. Na graf jsem vybral jeden ze změřených průběhů, kdy už celý tento systém byl nestabilní.
Oproti předchozím variantám varianta *Bez zásahu* dopadla pro tyto vstupy podle očekávání špatně - všechna testovací měření dopadla se stejným výsledkem - a to sice tak, že systém je nestabilní a RC model se roztočil kolem vlastní osy. Pro porovnání jsem testoval při jakých hodnotách vstupů varianta *Bez zásahu* začne být stabilní. Obě předchozí varianty si stabilitu zachovaly. Došel jsem na poloviční vstup pro Motor reference, kdy se varianta *Bez zásahu* začala chovat stabilně. Vstupy pro tento test jsou na grafu 5.7. Výstupy změřené na gyroskopu jsou grafu 5.8. Je zde vidět, že systém se chová již stabilně.



(a) : Průběh vstupu Motor reference změřený pomocí řídící jednotky

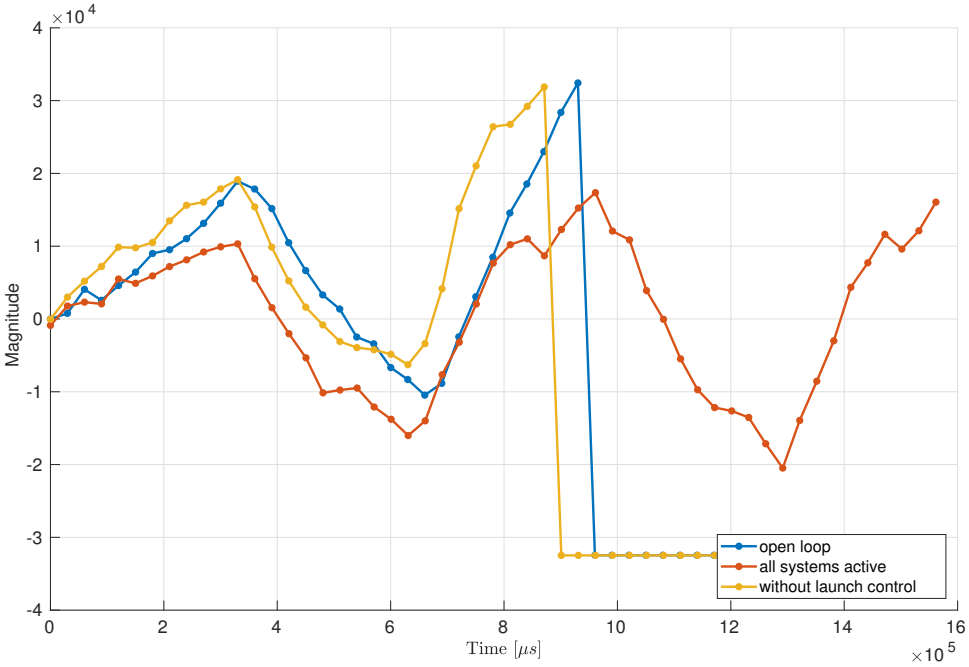


(b) : Průběh vstupu Servo reference změřený pomocí řídící jednotky

Obrázek 5.5: Průběhy vstupů pro testovací scénář "Průjezd zatáčkami"- Rychlejší varianta

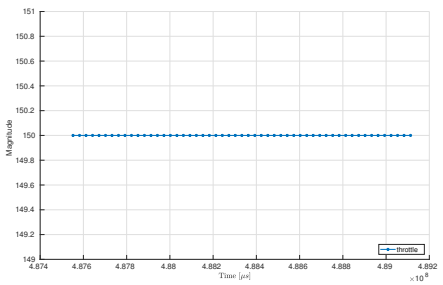
5.5 Shrnutí dosažených výsledků

Všechny řídící zákony byly naimplementovány s pomocí (ve formě konzultací) vedoucího práce. Active steering i torque vectoring dosahují kýžených výsledků, přičemž při manuálním ovládání RC modelu je velký rozdíl při jejich aktivním zapojení oproti variantě *Bez zásahu*.
Systém launch control nedosahuje očekávaných kvalit a jen drobně zlepšuje jízdní vlastnosti RC modelu. Tento systém jsem se snažil nejprve naimplementovat pomocí dat z akcelerometru a otáček poháněných kol. Bohužel po prvních testech akcelerometru se ukázalo, že pomocí integrace nelze určit rychlost RC modelu. Integrační chyba u použitého akcelerometru je tak velká (je i s časem proměnlivá), že nelze tento akcelerometr použít k určení inerciální rychlosti. Další výraznou slabinou tohoto systému je měření otáček kol. Toto měření je velice nepřesné a zatížené šumem. Jen zhruba odpovídá pozorované realitě.
Pro další postup v tomto projektu navrhuji vytvořit systém pro měření rych-

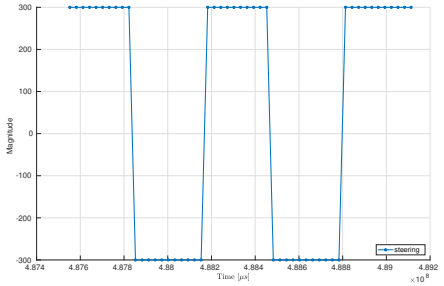


Obrázek 5.6: Změřené hodnoty z gyroskopu, porovnání výsledků pro varianty *Vše zapnuté*, *Bez launch control* a *Bez aktivního řízení*. Pro vstupy z grafu 5.5.

losti kol, a to jak u poháněné nápravy, tak i u nepoháněné nápravy, a tím pádem se více přiblížit architektuře systému ASR používaným v osobních automobilech (viz [Kie05]). Dalším možným vylepšením je vytvoření systému pro měření/odhad inerciální rychlosti, která by byla pro následný návrh dalších řídících zákonů zajímavá.

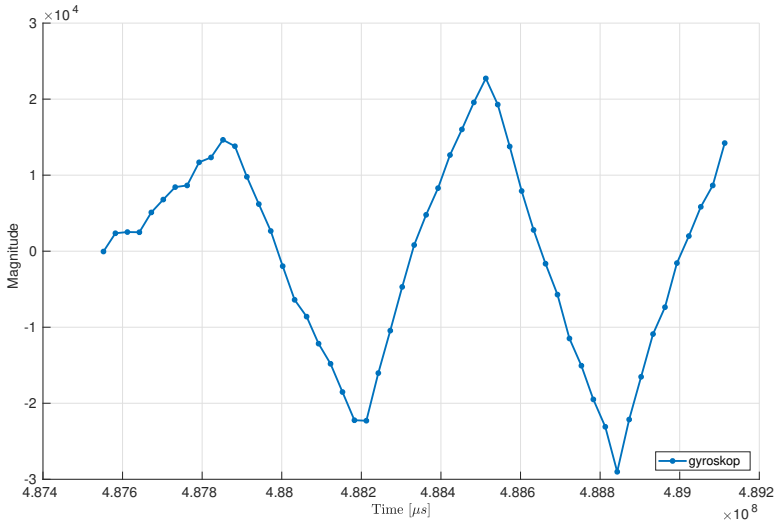


(a) : Průběh vstupu Motor reference změřený pomocí řídící jednotky



(b) : Průběh vstupu Servo reference změřený pomocí řídící jednotky

Obrázek 5.7: Průběhy vstupů pro testovací scénář *Průjezd zatáčkami* - Pro tyto vstupy jsou stabilní již všechny varianty testů.



Obrázek 5.8: Změřené hodnoty z gyroskopu, pro variantu *Bez aktivního řízení* pro vstupy z grafu 5.7.

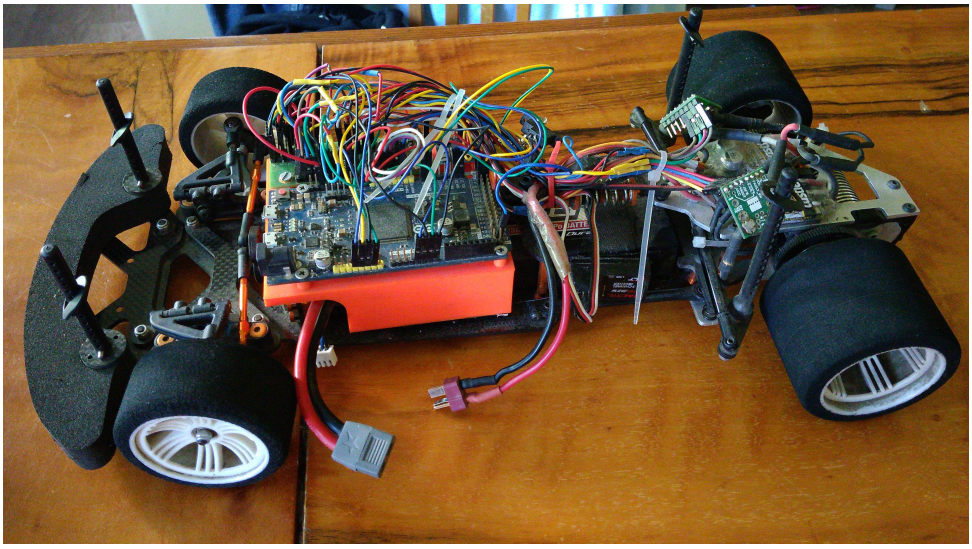
Kapitola 6

Závěr

Tato práce se zabývala vytvořením testovací platformy pro účel potvrzení principů platných u řídících zákonů aplikovatelných pro automobilový průmysl a jejich následnou validací.
Mým osobním přínosem do tohoto projektu bylo:

- Návrh základního mechanického RC modelu - vybral jsem po konzultacích s panem Ing. Helmichem model, který by se hodil pro testování řídících zákonů platných pro silniční vozidlo. Rozhodl jsem se pro model Xray Pan car GT 1/10. Další práce ohledně mechanické úpravy prováděl pan Helmich bez mého přispění.
- Návrh modelu pro 3D tisk - vytvořil jsem model pro 3D tisk, který slouží k připevnění elektroniky. Tento model jsem předal Ing. Helmichovi, který ho po drobných úpravách (přidání montážních otvorů) vytiskl.
- Návrh a implementace elektroniky - navrhl jsem pro nákup všechny součástky, které se netýkaly mechanického zpracování. Jsou to:
  - Motory a jejich regulátory, servo, akumulátor, vysílač a přijímač.
  - Senzory - proudový senzor, gyroskop a akcelerometr
  - Řídící jednotka Arduino Due
  - Telemetrická jednotka HC-12

  Tyto moduly jsem následně spojil a vytvořil kód pro jejich správnou funkci.
- Návrh spojovací desky - pro potřebu spojení všech použitých jednotek, jejich napájení a jednoduchého analogového předzpracování signálu z

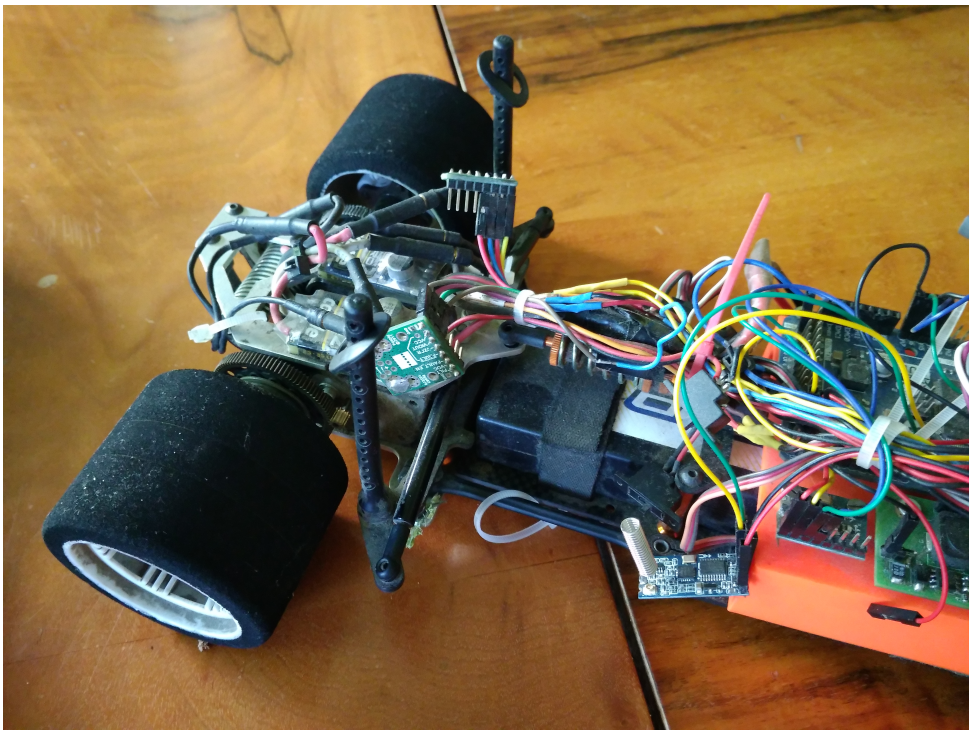


Obrázek 6.1: Finální vzhled RC modelu.

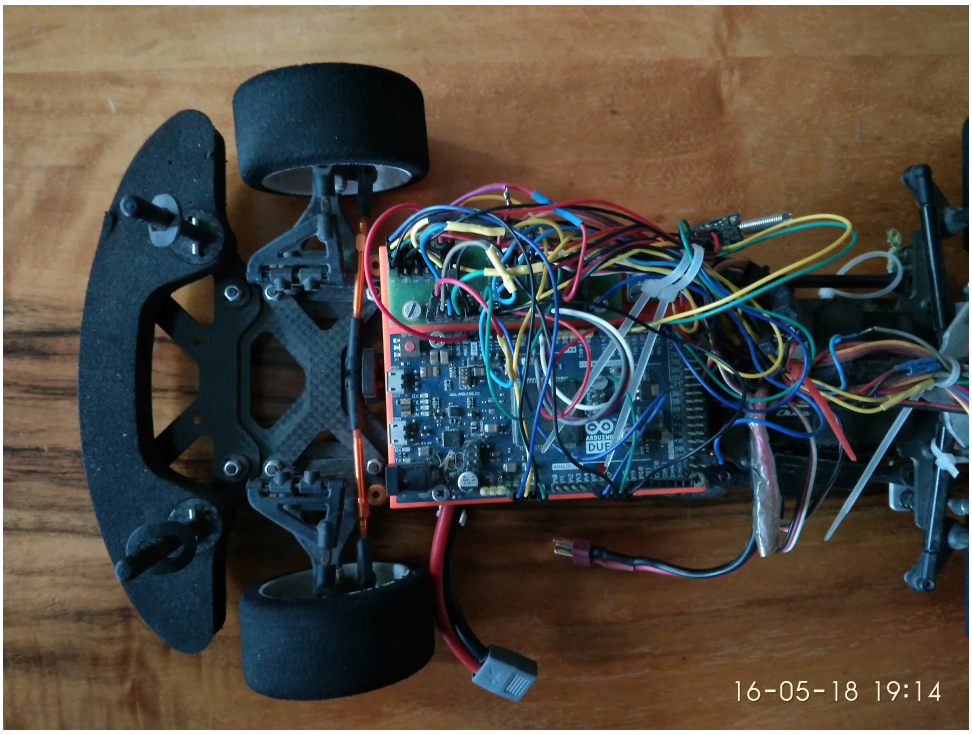
proudových senzorů jsem navrhl, nechal vyrobit a následně osadil a zapojil spojovací desku.

- Návrh a validace řídících zákonů - navrhl jsem a zvalidoval některé jednoduché řídící zákony.

Jak bylo napsáno v 5.5 systémy torque vectoring a active steering mají dobrý výsledek. Systém launch control na tom není až tak dobře. V sekci 5.5 je navrhnut další postup pro vylepšení tohoto systému.
Směrem, kterým by se další práce na tomto modelu měla ubírat, je zlepšení elektromagnetického rušení. Kvůli dlouhé společné cestě od proudových senzorů k spojovací desce vzniká jak velká kapacitní, tak i velká indukční vazba mezi vodiči přenášejícími analogový signál. Toto je z části kompenzováno proudovým zatížením výstupů senzorů. Avšak i přes tento fakt jsou data ponořena ve velkém šumu. Další možností je použití digitálních senzorů.
Elektromagnetické rušení je však problém i pro jiné systémy jako např. pro příjem signálu a nebo pro akční členy. V případě vedení vodičů moc blízko u sebe se akční členy začnou chovat nevyzpytatelně. Toto je možné řešit např. vytvořením jedné kompletní PCB, kde by byla přítomna všechna použitá elektronika (senzory, řídící jednotka,...).
Další věcí, kterou by se nadále projekt měl zabývat, je připojení k MATLABu/Simulinku, aby se dal provádět model based design.
Celý RC model včetně detailů je na obrázcích 6.1, 6.3 a 6.2.



Obrázek 6.2: Nainstalované motory a k nim připojené regulátory společně s proudovými senzory.



Obrázek 6.3: Detail zapojené elektroniky v RC modelu

Příloha A

Literatura

[Ard18] Arduino, *Arduino homepage*, https://www.arduino.cc/, 2018.

[Ato13] Blogger Atom, *Arduino due pinout*, http://embdedsystems.blogspot.cz/2013/11/arduino-due-pinout.html, nov 2013.

[Cal16] CalcRC, *Calc rc*, http://rc.305.cz/view.php?cisloclanku=2014010004, 2016.

[Con18] Electrical Concepts, *Brushless dc (bldc) motor*, https://etrical.blogspot.cz/2016/05/brushless-dc-bldc-motor.html, jan 2018.

[Cor15] Atmel Corporation, *Atmel sam3x / sam3a series*, http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf, mar 2015.

[Cor18] Seiko Epson Corp., *Gyro sensors*, https://www5.epsondevice.com/en/information/technical_info/gyro/, 2018.

[Dev09a] Analog Devices, *Adxl335 small, low power, 3-axis ±3 g accelerometer*, https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf, 2009.

[Dev09b] ———, *Adxl345 3-axis, ±2 g, ±4 g, ±8 g, ±16 g digital accelerometer*, http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf, 2009.

[Ele15] Makerlab Electronics, *Triple axis accelerometer and gyro breakout – mpu6050*, https://www.makerlab-electronics.com/product/triple-axis-accelerometer-gyro-breakout-mpu6050/, 2015.

[GRVSGG10] José Carlos Gamazo-Real, Ernesto Vázquez-Sánchez, and Jaime Gómez-Gil, *Position and speed control of brushless dc motors using sensorless techniques and application trends*, Sensors **10** (2010), no. 7, 6901–6947.

[Gyf17] Gyfcat, *How mems accelerometer gyroscope magnetometer work*, https://gfycat.com/gifs/detail/GrouchyThickAmericanbobtail, may 2017.

[Haa03] Vladimír Haasz, *Elektrická měření: přístroje a metody*, Vydavatelství ČVUT, Praha, 2003.

[Hob16] HobbyKing, *Hobbyking x-car 5.5 turn sensored brushless motor (6069kv)*, https://hobbyking.com/en_us/hobbyking-x-car-5-5-turn-sensored-brushless-motor-6069kv.html?___store=en_us, 2016.

[Hob18a] ———, *Turnigy trackstar 6.5t sensored brushless motor 5485kv (roar approved)*, https://hobbyking.com/en_us/turnigy-trackstar-6-5t-sensored-brushless-motor-5485kv-roar-approve
html, 2018.

[Hob18b] ———, *Turnigy xk2430-5300kv brushless inrunner*, https://hobbyking.com/en_us/turnigy-xk2430-5300kv-brushless-inrunner.html, 2018.

[Inv10] InvenSense, *Itg-3200 product specification revision 1.4*, https://www.sparkfun.com/datasheets/Sensors/Gyro/PS-ITG-3200-00-01.4.pdf, 2010.

[Inv13] ———, *Mpu - 6000 and mpu - 6050 product specification revision 3.4*, https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf, 2013.

[Kie05] U Kiencke, *Automotive control systems : for engine, driveline, and vehicle*, Springer, Berlin New York, 2005.

[Lá15] Tereza Lásková, *Hc-12 wireless serial port communication module*, https://drive.google.com/file/d/0B4B30jzMyzG8c1FkYjZNcndRdHM/view, 2015.

[Nak14] Rajesh Nakarja, *Project aelith digital controller*, https://github.com/hammerfet/AelithController, 2014.

[Nel98] Robert Nelson, *Flight stability and automatic control*, WCB/McGraw Hill, Boston, Mass, 1998.

[NPT16] NPTEL, *Hall effect*, http://nptel.ac.in/courses/115103030/26, 2016.

[Por] PowerGuru – Power Electronics Information Portal, *Introduction to closed loop hall effect current transducers*, http://www.powerguru.org/closed-loop-hall-effect-current-transducers/.

[Rac18a] HPI Racing, *Hpi d-box 2 adjustable stability controll system*, http://www.hpiracing.com/en/part/105409, 2018.

[Rac18b] ———, *Rtr sprint 2 flux w*, http://www.hpiracing.com/en/kit/106165, 2018.

[Rep18] TME Czech Republic, *Pololu-2199*, https://www.tme.eu/cz/de433MhzHC-12SI4463tails/pololu-2199/moduly-cidel/pololu/, 2018.

[Rip07] Pavel Ripka, *Modern sensors handbook*, ISTE USA, Newport Beach, CA, 2007.

[Row18] Jeff Rowberg, *I2c device library*, https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050, 2018.

[Sem14] NXP Semiconductors, *Um10204, i^2 c-bus specification and user manual*, https://www.nxp.com/docs/en/user-guide/UM10204.pdf, 2014.

[SLS18] Michael Stanley, Jongmin Lee, and Andreas Spanias, *Sensor analysis for the internet of things*, Morgan & Claypool, 2018.

[ST10] ST, *L3g4200d, mems motion sensor: ultra-stable three-axis digital output gyroscope*, https://www.parallax.com/sites/default/files/downloads/27911-L3G4200D-Gyroscope-Manufacturer-Datasheet.pdf, 2010.

[Tra18] Traxxas, *Tsm - traxxas stability management*, https://traxxas.com/products/parts/traxxas-stability-management, 2018.

[Ved04] Josef Vedral, *Elektronické obvody pro měřící techniku*, České vysoké učení technické, Praha, 2004.

[Vos18] David Vosahlik, *Rc car project github repository*, https://github.com/dvosahlik/RC_Car, 2018.

[WIK16] WIKISKRIPTA, *Hallův jev*, https://www.wikiskripta.eu/w/Hall%C5%AFv_jev, 2016.

Příloha B

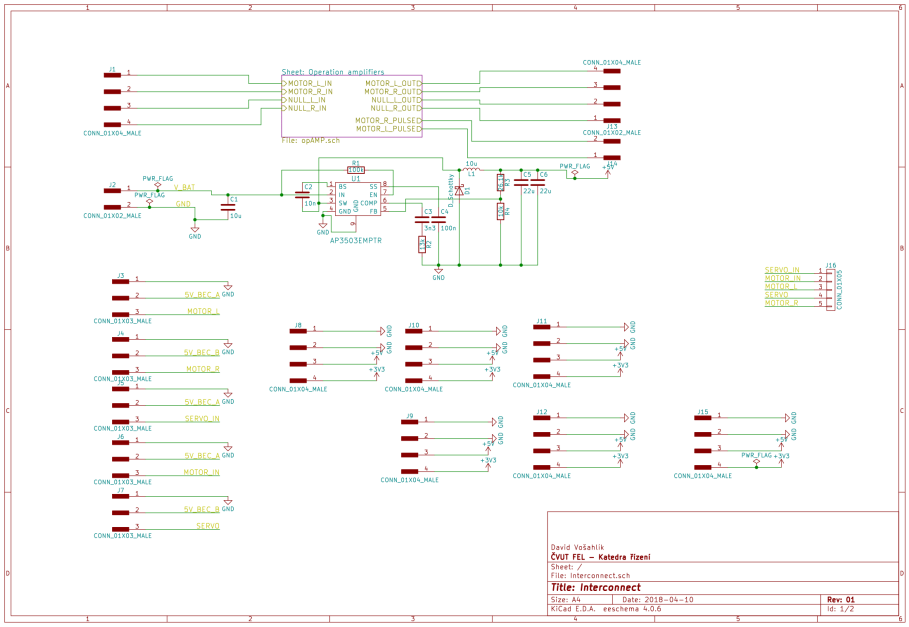
Seznam příloh

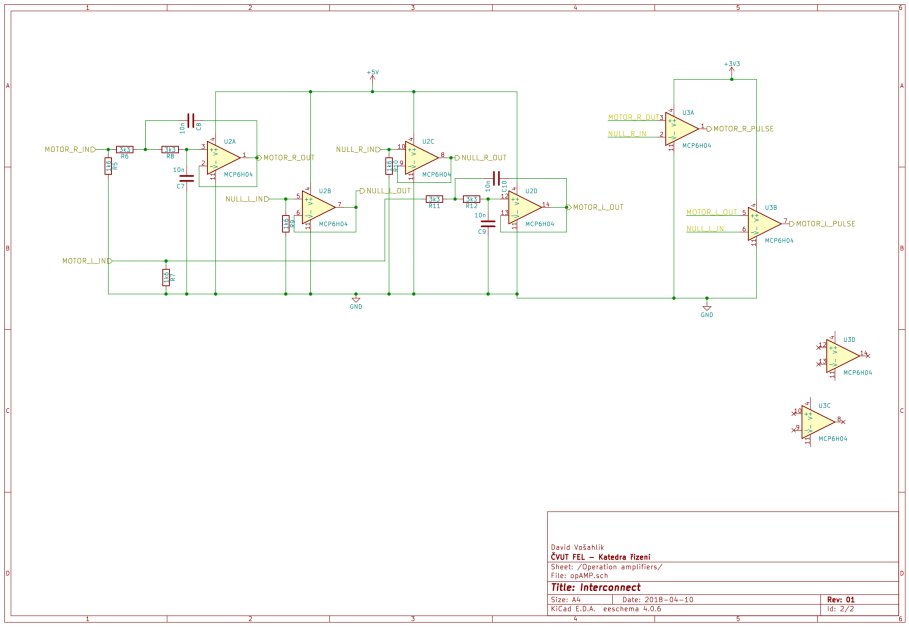
Zde jsou popsány přílohy odevzdávané spolu s prací na DVD nosiči. V kořenovém adresáři DVD jsou následující soubory:

- Xray.blend - jedná se o projekt v programu Blender 3D. V tomto projektu je vymodelováno okolí 3D výtisku použitého pro mechanické sestavení. Tento 3D výtisk je zde také vymodelován.
- rc_car_source.zip - je komprimovaná složka, ve které je umístěn projekt pro Atmel Studio 7. V tomto projektu je zdrojový kód pro řídící jednotku Arduino Due
- rc_car_PCB.zip - je komprimovaná složka s projektem na spojovací DPS. Projekt je vytvořen v programu KiCAD.
- Systém aktivního řízení pro RC auto.pdf - je tato bakalářská práce v elektronické podobě.

Příloha C

Schéma spojovací desky



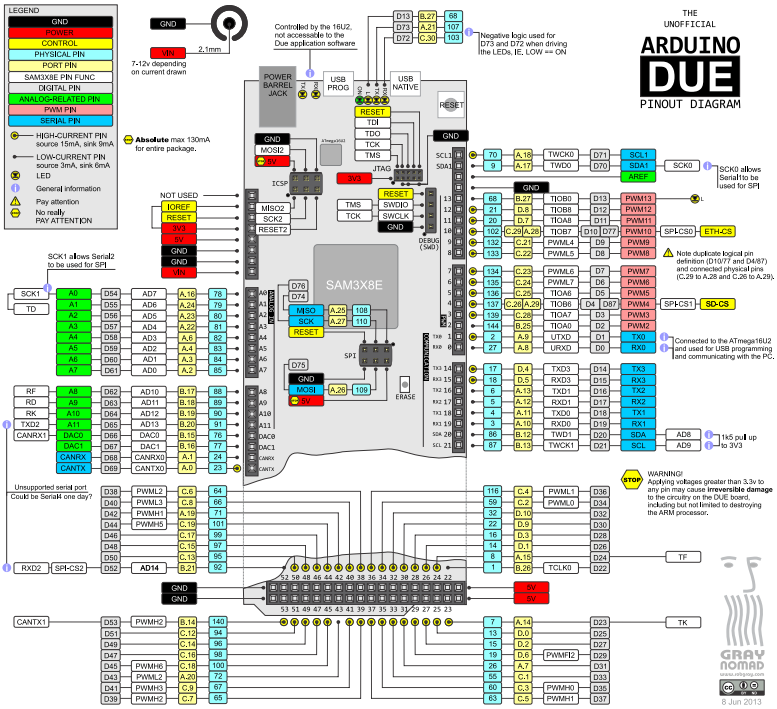


Obrázek C.1: Schéma spojovací desky vytvořené v programu KiCAD.

Příloha D

Zapojení jednotky Arduino Due

Na obrázku D.1 je schématicky znázorněna jednotka Arduino Due. Připojení jednotlivých pinů je popsáno v tabulce D.1.



Obrázek D.1: Schématické zobrazení pinů jednotky Arduino Due (pinout). Převzato z [Ato13]

Název pinu	Význam pinu	**připojené místo**
AREF	Analogová reference	+3,3V
PWM 8		Ovládání serva
PWM 7		Ovládání pravého motoru
PWM 6		Ovládání levého motoru
TX3	TX pin USART 3	RX pin HC-12
RX3	RX pin USART 3	TX pin HC-12
SDA	I^2C pin	SDA pin MPU6050
SCL	I^2C pin	SCL pin MPU6050
D 22	digitální pin	výstup přijímače - motor
D 26	digitální pin	výstup měření otáček - pravý
D 27	digitální pin	výstup měření otáček - levý
D 53	digitální pin	výstup přijímače - servo
A 7	analogový pin	nulový výstup POLOLU - levý
A 6	analogový pin	výstup POLOLU - levý
A 5	analogový pin	nulový výstup POLOLU - pravý
A 4	analogový pin	výstup POLOLU - pravý
GND	napájení	GND Interconnect
5V	napájení 5V pro Arduino	výstup napájení z Interconnect
3V3	výstup 3V3 z Arduina	napájení 3V3 pro Interconnect

Tabulka D.1: Zapojení jednotlivých pinů na řídící jednotce Arduino Due.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vošahlík** Jméno: **David** Osobní číslo: **439578**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Systémy a řízení**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém aktivního řízení pro RC auto

Název bakalářské práce anglicky:

Active control for an RC scale car

Pokyny pro vypracování:

1. Navrhněte a zrealizujte palubní řídicí jednotku a proveďte zástavbu do RC modelu.
2. Zrealizujte vhodnou senzorickou jednotku (inerciální měření, případně měření proudových odběrů a otáček motorů).
3. Zprovozněte komunikaci mezi řídicí jednotkou, senzory a akčními členy.
4. Navrhněte systém telemetrie.
5. Implementujte a naparametrizujte vybrané jednoduché řídicí zákony.
6. Experimentálně zvalidujte vybraná řešení a zhodnoťte výsledky.

Seznam doporučené literatury:

[1] Kiencke, Nielsen, Automotive Control Systems, Springer 2005.
[2] Nelson, Flight stability and automatic control, McGraw-Hill Education 1997.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Martin Hromčík, Ph.D., katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

doc. Ing. Martin Hromčík, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta