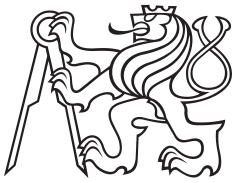
Bakalářská práce



České
vysoké
učení technické
v Praze

FEL

Fakulta elektrotechnická
Katedra řídící techniky

Návrh a realizace řídícího systému pro autonomní autodráhové autíčko určené pro experimentování s inteligentními konvoji

Marek Bečka

Vedoucí: doc. Ing. Zdeněk Hurák, Ph.D.
Obor: Kybernetika a robotika
Studijní program: Systémy a řízení
Únor 2018

Poděkování

Mé díky patří vedoucímu práce doc. Ing. Zdeňku Hurákovi, Ph.D. za konzultace i připomínky a celé katedře řídící techniky za poskytnuté zázemí. Dále děkuji rodině a přátelům za podporu.

Prohlášení

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

V Praze, 25. května 2018

Marek Bečka

Abstrakt

Tato bakalářská práce má za cíl vytvořit platformu pro experimentování s algoritmy pro distribuované řízení konvojů automobilů. Řešení využívá běžnou autodráhu s na míru navrhnutými autonomními autíčky. Tato autíčka jsou postavena na zakoupeném podvozku a karoserii vyrobené pomocí 3D tisku. Platforma navazuje na předchozí implementaci stejného problému. Oproti této verzi je navrhnuto nové hardwarové řešení, které využívá hotových vývojových kitů pro mechatronické systémy. Hlavní výpočetní jednotkou je BeagleBone Blue, který je doplněn o běžně dostupné modulární senzory pro měření rychlosti autíčka a vzdálenosti k následujícímu vozu. Software je obdobně jako v komerční praxi automaticky generován z grafických modelů v prostředí Simulink. Oproti předchozí verzi se tak zlepšila reprodukovatelnost hardwaru i komfort softwaru pro experimenty. Práce popisuje postup návrhu hardwarového řešení, softwarové řešení a experimenty ověřující funkčnost systému.

Klíčová slova: Autonomní konvoj, kolona, autodráha, BeagleBone Blue

Vedoucí: doc. Ing. Zdeněk Hurák, Ph.D.
Katedra řídící techniky,
Karlovo námestí 13/E,
Praha 2

Abstract

The goal of this bachelor´s thesis is to create a platform for experimenting with algorithms for distributed control of car platoon. The solution utilizes ordinary slot car track and self-made tailored autonomous cars. These cars are built on purchased chassis with 3D printed car body. The platform is a follow-up on previous implementation of the same issue. But in comparison, this version uses new hardware solution, which utilizes development kits for mechatronics systems. The main computational unit is BeagleBone Blue, completed with commonly available modular sensors for measuring the cars speed and distance to another vehicle. Software is generated from graphic modules in Simulink, as usual in commercial sphere. This thesis describes the process of development of hardware and software solution and experiments verifying the right functionality of the system.

Keywords: Autonomous convoy, Platoon, Slotcar Track, BeagleBone Blue

Title translation: Design and realization of a control system for an autonomous slot car intended for experimenting with vehicular platoons

Obsah

1 Úvod **1**

1.1 Předchozí implementace 2

2 Hardware **5**

2.1 Podvozek 5

2.2 Karoserie a držáky BB 5

2.3 Elektronika 8

2.3.1 BeagleBone Blue 8

2.3.2 Napájení 9

2.3.3 Externí senzory 10

3 Software **13**

3.1 Ovladače 14

3.1.1 Senzor přiblížení 14

3.1.2 Senzor IMU (Inertial Measurement Unit) 14

3.2 Simulink 15

3.2.1 Senzor přiblížení 15

3.2.2 Senzor odometrie 16

3.2.3 Komunikace 16

3.2.4 Model autíčka 16

3.2.5 Regulace rychlosti 17

3.2.6 Regulace vzdálenosti autu vpředu 17

4 Zprovoznění **19**

4.1 Update desky 19

4.2 Nastavení Simulinku 21

5 Experimenty **23**

5.0.1 Udržování konstantní rychlosti - tempomat 23

5.0.2 Akcelerometr 23

5.0.3 Senzor vzdálenosti 24

6 Závěr **29**

A Literatura **31**

B Zadání práce **33**

Obrázky

1.1 Původní verze 2

1.2 Degradace barvy starého ozubeného kola 3

2.1 Upravený podvozek Slot.it HSR2 6

2.2 Podvozek osazený držáky, kondenzátory a senzorem odometrie 6

2.3 Navrhnuté díly vytištěné 3D tiskárnou 7

2.4 Konvoj dvou kamionů 8

2.5 BeagleBone Blue 9

2.6 Adafruit VCNL4010 10

2.7 Magnetic Encoder Kit Pololu . . . 11

5.1 Průběh rychlosti bez regulace . . 24

5.2 Průběh rychlosti s regulací 25

5.3 Akční zásah – regulace rychlosti 25

5.4 Akcelerometr (nefiltrovaný) 26

5.5 Akcelerometr (filtrovaný) 26

5.6 Gyroskop 27

5.7 Senzor vzdálenosti na rovince . . 27

5.8 Senzor vzdálenosti v zatáčce . . . 28

Tabulky

Kapitola 1

Úvod

Žijeme v době, kdy považujeme různé asistenty řízení za zcela běžnou a přirozenou součást našich životů. Obyčejný tempomat udržující konstantní rychlost je součástí výbavy mnoha aut již desetiletí, jeho vylepšená adaptivní verze snímající vzdálenost k vozu před námi se stává standardem v současnosti. Mnoho firem dnes vyvíjí autonomní vozidla, která dále zlepší komfort cestování automobily. Dalším krokem jsou skupiny kooperujících autonomních vozů, jež dále zlepší bezpečnost, sníží znečištění a zvýší kapacitu komunikací. Tyto vozy mohou jezdit výhledově i bez lidské obsluhy, a tak má tato technologie potenciál také ušetřit pracovní sílu.

Tyto skupiny mohou tvořit konvoj automobilů, který se zformuje na začátku jízdy a pokračuje až do cílové destinace. Jiný přístup by mohl být řízení kolony vozů, kde se jednotlivé automobily připojují a odpojují dle svých cílových destinací. Ucelený konvoj by mohl být žádoucí např. pro nákladní automobily jedné spediční firmy, které mají společný cíl stovky nebo tisíce kilometrů daleko.

Cílem této práce je navrhnout a realizovat hardware i software pro testování řídících algoritmů právě takových konvojů. Platforma je postavena na základu autodráhy. To má oproti skutečné silnici výhodu absence jednoho stupně volnosti, a tak se můžeme soustředit skutečně jen na regulaci rychlosti a kooperaci v podélném směru jízdy. Odpadá tak problém udržení auta v jízdním pruhu.

Pro stavbu využiji jednodeskové počítače BeagleBone Blue a existující

modulární řešení ostatních senzorů, které jsou přímo určeny pro podobné robotické aplikace. Řídící software bude generován z programu Matlab Simulink. Oproti předchozím verzím tato řešení zásadně zlepší vlastnosti systému, jako jsou reprodukovatelnost, rychlost softwaru, uživatelská přívětivost pro experimenty a dokumentace pro další případná rozšíření.

1.1 Předchozí implementace

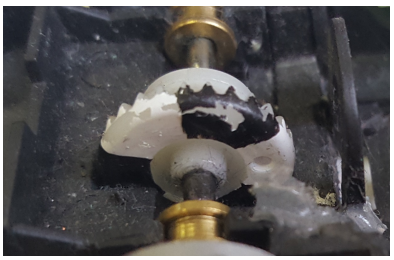
Obrázek 1.1: Původní verze

Stejným problémem jako já se zabývali již dva předešlí studenti ve svých diplomových pracích, konkrétně Dan Martinec [Mar12] a Martin Lád [Lá17][MLH17]. Ve své práci vyjdu právě ze stávající verze Martina Láda (1.1) a pokusím se ji zlepšit.

Předchozí implementace konvoje využívá z části na míru (studentem) navržený hardware. Tato řešení jsou plně funkční a pro samotné experimenty vyhovující. Problémem je, pokud bychom se snažili platformu modifikovat a nebo i pouze reprodukovat. Předchozí verze totiž využívá navrhnuté procesorové desky osazené mikrokontrolérem *STM32F401*, která je použita pro řízení motoru, akcelerometr, gyroskop, odometrii a napájení autíčka. Deska komunikuje s hlavním počítačem (*Raspberry Pi*). Celá tato infrastruktura je složitá a pro pochopení vyžaduje značný čas. Dalším problémem je stavba nového autíčka, kdy je třeba vyrobit PCB desku, tu osadit součástkami a oživit. Původní autíčko navíc výrobce Carrera přestal vyrábět, takže by nově vyrobené vozidlo bylo odlišné od původních.

Senzor vzdálenosti k následujícímu autíčku v optimálním případě funguje,

ovšem má hluché místo pokud bylo autíčko vpředu příliš blízko (<1,5 cm). Pro řídící algoritmus tak bylo složité vyhodnotit, jestli je skutečná vzdálenost 0 nebo naopak auto vpředu daleko mimo rozsah senzoru. Podobně pro odometrii byl použit optický čítač pulzů z nápravy, který měl nižší rozlišení a vyrobený kotouček s bíločernými pruhy časem degradoval (1.2).



Obrázek 1.2: Degradace barvy starého ozubeného kola

Předchozí verze platformy využívá upravené běžné autodráhové autíčko. Kvůli tomu byl problém autíčko vůbec osadit potřebnou výbavou. Do standardní karoserie se musely vejít kromě motoru navíc spodní deska, Raspberry a superkapacitor. To si vyžádalo některé úpravy, které snížily atraktivitu pro laického diváka (absence čelního skla, viditelné PCB Raspberry, senzory vzdálenosti místo světel).

Software předchozí verze je celý napsaný v jazyce Java. To sice částečně usnadňovalo vývoj GUI, ovšem za cenu rychlosti a výkonnosti řídící části softwaru. Ve výsledku vznikl velký monolitický program, který obsahoval jak GUI tak řídící algoritmy. Interakce se Simulinkem byla možná, ale pouze prostřednictvím právě zmíněného hlavního programu v Javě. Běžným standardem v dnešním automotive průmyslu přitom je navrhnout řízení v Simulinku a kód generovat automaticky právě z něj.

V mé práci se pokusím tyto neduhy odstranit. Desku použiji jednu standardní, místo úpravy běžného autíčka postavím vlastní karoserii na koupeném podvozku. Potřebné senzory budou modulární a snadno připojitelné běžnými rozhraními k BeagleBone. Poslední cíl je řídící algoritmy celé generovat ze Simulinku.

Kapitola 2

Hardware

V této části popíši jednotlivé součásti hardwarového řešení autíčka, a to jak elektrické části, tak ostatních dílů.

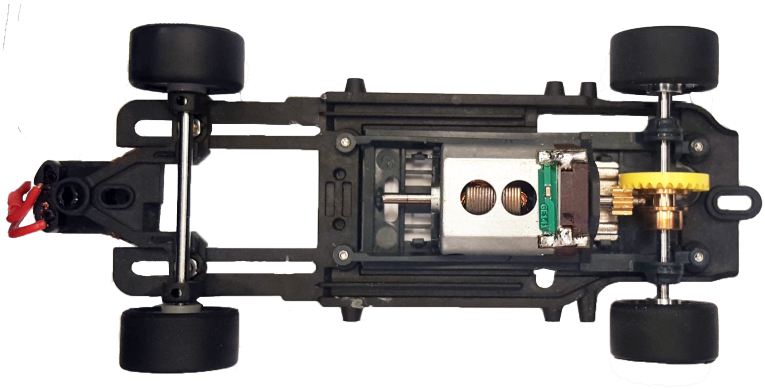
2.1 Podvozek

Pro projekt jsem zvolil plně konfigurovatelné, univerzální šasi Slot.it HRS2[1]. Tento podvozek využívá inline konfiguraci motoru s devítizubým pastorkem a 28 zuby na větším kole na nápravě. Převod je tak 3,1:1. Rozvor je volitelný, vzhledem k proporcím kamionu jsem ho nastavil na 84 mm a podle tohoto rozměru jsem navrhl karoserii. Dále bylo nutné podvozek mírně upravit kvůli montáži držáků desky Beaglebone. Tato úprava zahrnuje vyvrtanou díru pro zadní držák a mírné zvětšení drážek pro šrouby, které drží karoserii2.1.

2.2 Karoserie a držáky BB

Součástí práce je také návrh karoserie a již zmíněných držáků pro desku. Tyto díly jsem vymodeloval v programu Autodesk Fusion 360. Stl modely jsem

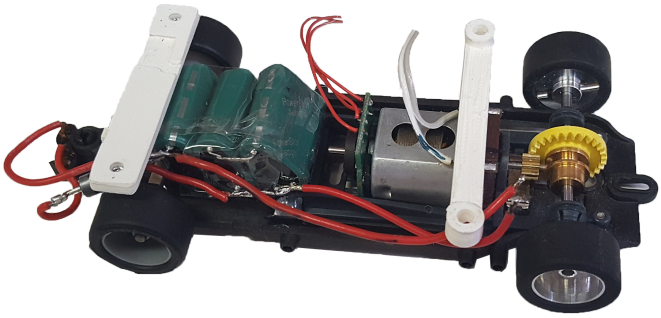
[1] `https://www.pendleslotracing.co.uk/slot-it-hrs2-complete-chassis-inline-sich109c.html`



Obrázek 2.1: Upravený podvozek Slot.it HSR2

následně zpracoval Slicer programem Ultimaker Cura. Vzniklý formát .gcode formát jsem následně přenesl do 3D tiskárny Ultimaker 2+ a vytiskl. Použitý filament je pro naše potřeby populární a plně postačující plast PLA. Výhoda tohoto materiálu je snadná tisknutelnost, dostatečná pevnost a také pružnost pro případ srážek.

Držáky. BeagleBone je třeba pevně uchytit k podvozku, k čemuž jsem navrhl přední a zadní držák. Tyto díly jsou připevněny pomocí šroubu M2,5 (přední) nebo vrutů do plastu M3 (zadní) k podvozku. Tvar předního držáku je přizpůsoben poloze nápravy a součástkám na BB. Zadní držák obepíná motor. Díly jsem navrhoval ve třech iteracích, aby dokonale seděly.



Obrázek 2.2: Podvozek osazený držáky, kondenzátory a senzorem odometrie

Karoserie. Na koupený podvozek bylo třeba navrhnout karoserii, která uchání komponenty uvnitř během nehod. Vzhledem k popularitě předchozí verze pro veřejnost na různých akcích typu Den otevřených dveří, Gaudeamus apod. je další neméně důležitou funkcí vzhled. Rozhodl jsem se navrhnout karoserii kamionu ve stylu Tesla Semi. Návrh probíhal ve dvou etapách. Protože se první cesta pomocí přesných výkresů nakonec ukázala jako slepá, tak jsem ji i přes dlouhý vývoj nakonec zahodil a začal znovu. Lepší přístup k návrhu nakonec byl přes sochařské (sculpting) prostředí. V programu Fusion 360 navrhl podle skici tvar, který jsem postupně vylepšoval.

Karoserie je rozdělená do pěti samostatně tisknutelných částí – přední nárazník s boky, čelní sklo, střechu, zadní vodorovnou část nad nápravou a zadní nárazník2.3. Uchycení karoserie je v případě hlavní části vpředu řešeno matkami M2,5. Boky drží pomocí tyček, které pasují do děr v podvozku. Tyto tyčky je třeba po vytištění nahřát a mírně (∼2 mm) ohnout dolů. Zkoušel jsem je tisknout i již ohnuté, ale v tom případě se zbytek zbytek karoserie musí tisknout s podpěrami a to zbytečně snížilo kvalitu tisku. V předu je naznačena maska se světlomety, ve kterých je 3 mm díra jako příprava pro LED diodu.

Čelní sklo drží samo díky navrhnutým zarážkám. Střecha je přilepena ke karoserii lepidlem. Zadní část nad nápravou je uchycena pomocí šroubů do plastu, které vedou ze zadního držáku skrz BB a úplně vzadu také jedním šroubem M2,5. Zadní nárazník je ke zbytku připevněný pomocí zámečků, takže se dá snadno vycvaknout. To je vhodné kvůli přístupu k micro USB konektoru, který je občas nutný pro vývoj a ladění (např. když BeagleBone se nepřipojí na wifi).

Obrázek 2.3: Navrhnuté díly vytištěné 3D tiskárnou

Karoserie je z bílého PVA, čelní sklo je vytištěno z černého materiálu. Tloušťka stěn je převážně 1,5 mm. V předním nárazníku je díra pro senzor přiblížení o rozměru 8 × 8 mm. Podobně nad zadní nápravou uprostřed je v karoserii 5 mm kulatá díra, která umožňuje v budoucnu případně připojit i návěs. To by jednak ovlivnilo dynamiku a pokusy s řízením by tak byly ještě zajímavější, jednak zvýšilo atraktivitu pro diváky.

Obrázek 2.4: Konvoj dvou kamionů

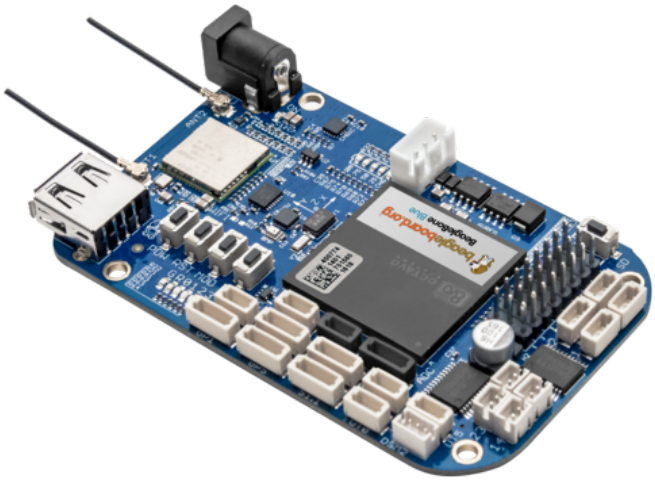
2.3 Elektronika

2.3.1 BeagleBone Blue

Autíčko je postaveno na kompaktním (86 × 55 mm) jednodeskovém počítači BeagleBone Blue (2.5). Tato deska je all-in-one otevřená platforma se SoC Octavo OSD3358, na němž běží operační systém Linux. Výhodou a důvodem volby BeagleBone Blue jsou rozměry PCB, výbava vhodná pro robotické aplikace a naopak absence zbytečných konektorů (HDMI...). Nechybí podpora Matlabu, ani obsáhlá dokumentace jak na internetu tak v podobě tištěných publikací[Gri17]. V předchozí verzi použité Raspberry Pi Computing Module je zamýšleno spíše jako levný počítač pro výuku programování, takže má slabší I/O rozhraní a chybí mu potřebná výbava (řízení motorů, akcelerometry apod.)

Z desky jsem odpájel rozměrné konektory napájení a USB-A. USB typu A není potřeba, jelikož drátovou konektivitu zajišťuje micro USB. Místo konektoru napájení jsem napájel dráty přímo na PCB. Z integrované výbavy zmíním pouze část, kterou jsem využil v projektu.

Toshiba TB6612FNG je čip s H-můstkem pro ovládání kartáčových DC motorů s permanentními magnety. Na PCB jsou tyto čipy osazeny dva a každý může ovládat až dva motory. Tento čip má všechny potřebné funkce jako otáčení po/proti směru hodinových ručiček, brzdění nebo funkci stop.



Obrázek 2.5: BeagleBone Blue

Napájecí napětí $V_{max} = 15$ V, v případě BB je čip napájen 12 V. Maximální výstupní proudy jsou $I_{peak} = 3{,}2$ A, $I_{avg} = 1{,}2$ A. Konektory pro připojení motoru k driveru jsou typu *JST-ZH (2 pin)*.

Kvadraturní dekodér. BB má čtyři dekodéry, které jsou součástí SoC. Na PCB jsou vyvedené do konektorů *JST-SH (4 pin)*. Používám je pro dekódování signálů ze senzoru rychlosti otáčení.

9-osá IMU (Inertial Measurement Unit). Čip MPU9250 je připojený pomocí interní sběrnice I^2C číslo 2. Čip integruje akcelerometr, gyroskop, magnetometr a teploměr.

Z ostatních rozhraní využiji **micro USB**, **wifi** a externí **I^2C** sběrnici vyvedenou konektorem *JST-SH (4 pin)*. V operačním systému Debian je tato sběrnice označená číslem 1.

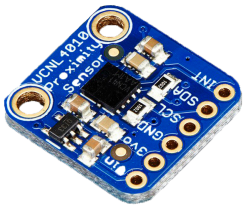
2.3.2 Napájení

Superkapacitor. BeagleBone je napájen ze sběračem proudu z kolejnic autodráhy. Původně jsem toto napájení zálohoval elektrolytickým kondenzátorem 10000 µF / 16 V. Toto řešení stačilo podle výpočtů na pokrytí malých výpadků ($< 0{,}2$ s), ovšem nakonec se ukázalo jako nedostatečné (externí senzory se občas zasekly, i když autíčko pokračovalo dál). Nyní napájení zálohuji sériovou kombinací tří superkapacitorů 1,5F / 5,4 V. BB tak bezpečně vydrží několik sekund ($\sim$10 s) bez napájení. Kvůli zamezení přepólování kondenzátorů je

předřazena klasická dioda 2 A / 1000 V. Superkapacitory jsou umístěny mezi přední nápravou a motorem podvozku.

2.3.3 Externí senzory

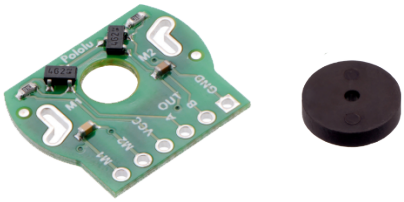
Senzor přiblížení. Pro regulaci rychlosti autíčka je třeba znát vzdálenost k autíčku vpředu. K tomuto účelu jsem zvolil **Adafruit VCNL4010**[2]. Rozměry PCB senzoru jsou 18 × 16,5 mm, což usnadňuje integraci do karoserie. Senzor funguje na principu měření množství odraženého infračerveného světla, jehož pulz je vyzářen na začátku každého měření. Výstupem senzoru je proximity. Toto číslo je po odečtení offsetu silně nelineárně závislé na vzdálenosti (∼12 pro 15 cm, ∼15600 pro 1 cm). Z datasheetu[3] i experimentálně jsem zjistil, že senzor dobře měří vzdálenosti ploch pod úhlem. Pro naši aplikaci, kdy autíčka mohou jet do zatáček je tato vlastnost nezbytná. Naopak nevýhodou senzoru je, že má nezměnitelnou I^{2}C adresu 0x13. Kvůli tomu jsem nakonec použil jen jeden vpředu na každé autíčko. V budoucnu by tento problém šlo vyřešit I^{2}C switchem, např. TCA9546A. PCB senzoru je přilepeno ke karoserii epoxidovým lepidlem.



Obrázek 2.6: Adafruit VCNL4010

Senzor rychlosti. Rychlost je měřena pomocí magnetického enkodéru Pololu[4][5]. Tento senzor využívá kombinace magnetického disku a dvou Hallových sond. Výstupem jsou pulzy, které dekódují již zmíněné dekodéry na BB. Rozlišení senzoru je 20 pulzů na otáčku motoru, což je ekvivalentní 62 pulzům na otáčku kola. Senzor je instalovaný na prodloužené hřídeli motoru. Sice tak neměříme přímo otáčky na nápravě, ovšem převody jsou velmi kvalitní a vůle je naprosto zanedbatelná. PCB senzoru je zacvaknuto do připravené pozice na podvozku2.2.

[2] produktová stránka: `https://www.adafruit.com/product/466`
[3] datasheet: `https://cdn-shop.adafruit.com/datasheets/vcnl4000.pdf`
[4] produktová stránka: `https://www.pololu.com/product/3499`
[5] datasheet: `https://www.pololu.com/file/0J815/TLE4946-2K.pdf`



Obrázek 2.7: Magnetic Encoder Kit Pololu

Kapitola 3

Software

Minulá verze používala software vyvinutý v Javě. To mělo výhodu např. v (obecně) uživatelské přívětivosti kódu, díky Java Virtual Machine přenositelnosti kódu mezi platformami nebo nativní podpoře grafiky pro GUI. Obecně Java ovšem pro podobné aplikace příliš vhodná není, a to kvůli své těžkopádnosti a relativní pomalosti. Současná převažující inženýrská praxe v řídících algoritmech směřuje ke automatickému generování kódu ze Simulinku. V mé verzi jsem se tedy vydal tímto směrem.

Pro BB existuje v Simulinku podpora ve formě *Simulink Coder Support Package for BeagleBone Blue* add-onu. Tento balíček obsahuje základní podporu pro BB. Zahrnuje bloky pro tlačítka, ovládání DC motoru, odečítání kvardaturního enkodéru a zapnutí/vypnutí LED diod. Dále podporuje odesílání a přijímání UDP a TCP/IP paketů. Pro interakci s I^2C periferiemi bylo třeba vyvinout ovladače.

Kódy, stejně jako veškerá dokumentace byly spravovány pomocí populárního verzovacího programu Git. Repozitář je dostupný na adrese `https://gitlab.fel.cvut.cz/beckama3/Slotcar-platooning-matlab`.

3.1 Ovladače

Problém komunikace Simulinku s nepodporovanými periferiemi jsem vyřešil pomocí bloků System Object. Tento bloček umožňuje spustit na BB kód v jazyce C a výsledek předat do prostředí Matlabu. Zdrojové kódy k ovladačům jsou ve složce `/drivers`. Vlastnosti a vzhled bloku se definují v **.m** souboru, zdrojové kódy jsou ve složce `/src` a hlavičkové soubory v `/include`. Ovladače byly vyvinuty z šablony `Source.m` získané Matlab příkazem `realtime.createDriverProject('my_driver')`.

3.1.1 Senzor přiblížení

Zdrojový soubor ovladače k tomuto senzoru se nachází ve složce `/drivers/I2Cdistancesensor/src/i2cdriver.c`. Ke komunikaci se senzorem používám SMBus příkazy, které čip senzoru podporuje.

Senzor **Adafruit VCNL4010** umožňuje nastavit proud IR LED diodou. Aby byl dosah senzoru co největší, tak je žádoucí proud nastavit na maximální hodnotu. Tuto hodnotu nastavuje metoda `set_current`, která je volána před prvním měřením. Metoda používá příkaz `i2c_smbus_write_byte_data`, který do registru 0x83 zapisuje hodnotu předanou ze Simulinku. Platný rozsah je 0d až 20d, jako výchozí hodnotu v nastavení bloku Simulinku jsem tedy nastavil 20d.

Samotné měření pak probíhá metodou `i2c_measure`. Nejprve je třeba dát senzoru pokyn k měření, což provedeme zápisem hodnoty 0x18 do registru 0x80. Výsledek lze následně vyčíst `(i2c_smbus_read_word_data)` z registru 0x87.

3.1.2 Senzor IMU (Inertial Measurement Unit)

Data z tohoto senzoru sice zatím nevyužívám, avšak v budoucích experimentech zřejmě najdou uplatnění. Údaje z tříosého akcelerometru a gyroskopu by se dala využít např. spolu s odometrií pro odhadování přesné polohy autíčka na trati. Zdrojový soubor ovladače se nachází v `/drivers/IMU/src/imu.c`.

Čip **MPU9250** komunikuje s procesorem pomocí interní I^2C sběrnice. Pro odečítání dat jsem využil existující API xxxxx odkaz xxxxxxx Robotics Cape. Při startu ovladače inicializuji struct `rc_mpu_data_t` a poté z ní jen odečítáme data (např `return data.gyro[0];`). Po zastavení Simulinkového kódu se čip vypne metodou `imu_terminate`.

3.2 Simulink

V této kapitole popíši Simulinkové schéma, ze kterého je generovaný program řídící autíčko. Zdrojové kódy tohoto programu generuje Simulink Coder, který je odešle na BB, kde se kompilují a spustí. Pokud použijeme externí mód, tak je část tohoto kódu zodpovědná za komunikaci se Simulinkem. Za běhu tak lze měnit některé části schématu, např. konstanty, používat přepínače apod. Zároveň lze online získávat data z probíhajícího experimentu. Velkou nevýhodou je, že tento mód lze spustit pouze pro jednu desku a tedy jedno autíčko.

V naší koloně na vedoucím autíčku běží obyčejný deploy-to-harware kód, zatímco na druhém je provozovaný zmíněný externí mód. Z toho důvodu jsem vyvinul dvě rozdílná schémata – `leader.slx` a `follower.slx`.

Experimentálně jsem zjistil, že nejvyšší možná frekvence Simulink kroků je 200 Hz, a tak používám právě tu. Solver Simulinku používám fixed-step discrete, který je nejrychlejší. Všechny signály v modelu všem musí být diskrétní.

3.2.1 Senzor přiblížení

V Simulinku dostáváme z bloku Distance Sensor číslo proximity, které musíme převést na vzdálenost v milimetrech. K tomu účelu je třeba nejdříve odečíst offset. Offset je u každého autíčka proměnný a záleží na velikosti vytištěné díry pro senzor a kvalitě přilepení. Pro účely testování jsem vyvinul blok Offset Detect, který průměruje hodnotu proximity první okamžiky simulace (0,2 - 1,2 s). Během této doby je nutné autíčko nechat v klidu na dráze a odstranit všechny překážky v okruhu 50 cm před kamionem.

Po odečtení offsetu dále signál zpracovávám running mean filtrem, který

průměruje poslední čtyři změřené hodnoty a vyhlazuje tak signál za cenu mírného zpoždění. Takto upravený signál linearizuji pomocí Matlab Function bločku. Vzorce pro linearizaci jsem získal rozdělením naměřených dat do tří intervalů (>30 - 11,5 cm; 11,5 - 4,5 cm; 4,5 - 0 cm) a postupnou linearizací pomocí Curve Fitting Toolboxu. Výstup z bloku saturuji na 250 mm, kdy se přesnost již ztrácí.

3.2.2 Senzor odometrie

Pro odometrii používám bloček Encoder poskytovaný z již zmiňovaného balíčku. Výstup bloku je poloha, kterou filtruji running mean filtrem z posledních 8 vzorků. Následně diskrétním ekvivalentem derivace, tedy diferencí, získávám rychlost. Tu následně saturuji na ± 10 Navzdory známým úskalím derivace signálů se tato metoda ukázala jako spolehlivá. Pro lepší odhad rychlosti by šlo v budoucnu navrhnout observer.

3.2.3 Komunikace

Komunikace mezi autíčky probíhá cestou auto-server-auto pomocí standardní wifi sítě. To je rozdíl oproti předešlé verzi, která používala ad-hoc wifi síť. Ta měla nevýhodu, že se někdy při výpadku více autíček rozpadla na dvě a nezbylo než autíčka restartovat. Na úrovni Simulinku je komunikace řešena bloky pro odesílání a přijímání TCP/IP a UDP paketů. Tato podpora je poskytnuta v rámci *Simulink Coder Support Package for BeagleBone Blue*.

3.2.4 Model autíčka

Pro účely rychlého návrhu regulátoru jsem systém identifikoval jako black-box. Experiment probíhal zadáním požadavku na 50% střídy PWM kontroléru motoru po dobu 0,7 s a sledováním odezvy. Autíčko jsem pouštěl po rovince. Systém jsem identifikoval pomocí System Identification Toolboxu jako diskrétní stavový model druhého řádu.

Existuje i přesnější metoda, kdy se vychází z naměřených fyzických parametrů motoru, hmotnosti vozidla, tření apod. Metodu podrobně popisuje Martin Lád v kapitole 5 [Lá17].

3.2.5 Regulace rychlosti

Pro udržování konstantní rychlosti (tempomat) jsem navrhl diskrétní PID regulátor. Regulátor byl navrhován pomocí identifikovaného modelu nástrojem PID tuning.

3.2.6 Regulace vzdálenosti autu vpředu

Pro regulaci vzdálenosti jsem experimentálně navrhl jednoduchý proporcionální regulátor. Vstupem je vzdálenost k autu vpředu minus požadovaná vzdálenost.

Kapitola 4

Zprovoznění

V této kapitole shrnu všechny kroky, které jsou potřeba pro stavbu nového autíčka a ke zprovoznění experimentu.

4.1 Update desky

Nejdříve stáhneme nejnovější Debian ze stránek výrobce[1]. Obraz operačního systému nahrajeme na micro SD (4GB+) kartu např. pomocí populárního nástroje Etcher[2]. Kartu vložíme do BeagleBone a pomocí micro USB kabelu desku připojíme k počítači. Beaglebone nyní nabootuje z karty.

Pro přepis interní eMMC paměti musíme upravit soubor na kartě. K Beaglebone se připojíme z linux terminálu pomocí Secure Shell (SSH). Zadáme příkaz `ssh debian@192.168.7.2` a po výzvě heslo `temppwd`. Pomocí známého příkazu `cd` se přesuneme do složky `/boot`. Příkazem `nano uEnv.txt` upravíme řádek

```
#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

na

[1] `https://beagleboard.org/latest-images`
[2] `https://etcher.io`

```
cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

Po restartu desky a vyčkání přibližně 15 minut bychom měli mít nejnovější operační systém v interní paměti. Vyjmeme micro SD kartu a pomocí wifi se připojíme k internetu příkazy

```
sudo connmanctl
connmanctl> scan wifi
connmanctl> services
connmanctl> agent on
connmanctl> connect wifi_ke_ktere_se_chceme_pripojit
connmanctl> quit
```

Status připojení k internetu, přidělenou IP apod. můžeme zkontrolovat příkazem `ip a`. Nyní aktualizujeme nainstalované balíčky a doinstalujeme potřebnou knihovnu k operacemi s I^2C příkazy

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install libi2c-dev
```

Tímto updatujeme i balíček Robotics Cape, z něhož v aktuální verzi vypadl hlavičkový soubor **`rc_usefulincludes.h`**. Tento soubor ovšem vyžaduje aktuální add-on pro podporu BB v Simulinku, a tak ho musíme opět vytvořit. Bohužel se mi z neznámého důvodu nepodařilo zprovoznit Secure Copy (scp), proto jsem vytvořil soubor přímo v konzoli. Příkazy `cd` se dostaneme do cesty **`/usr/include/`**. Soubor vytvoříme a editujeme příkazy

```
sudo touch rc_usefulincludes.h
sudo nano rc_usefulincludes.h
```

Do text editoru nano vložíme obsah souboru, který je přiložený k této práci. Odkaz na internetu je poněkud problematické najít, čerpal jsem z google web-cache.

Dále je třeba zkalibrovat gyroskop a akcelerometr, to provedeme následujícími příkazy a prováděním zobrazovaných instrukcí.

```
sudo rc_calibrate_accel
sudo rc_calibrate_gyro
```

Správnost kalibrace můžeme ověřit příkazem `rc_test_imu`.

4.2 Nastavení Simulinku

Při instalaci *Simulink Support Packege for BeagleBone Blue* je třeba mít připojenou desku. V průběhu instalace se nám zobrazí průvodce, který nakonfiguruje BB.

Po načtení Simulinkového schématu zvolíme Model Configuration Parameters a klepneme na kartu Hardware Implementation. Hardware Board je třeba nastavit na BeagleBone Blue, Target hardware resource/Board Parameters/Device Adress vyplníme IP adresu desky. Tu doporučuji zafixovat v routeru s MAC adresou, aby byla pokaždé stejná. V záložce external mode zvolíme Run external mode in a backgroud thread. Tím upřednostníme běh výpočetního vlákna algoritmu před komunikací desky se Simulinkem. V menu Solver/Solver options zvolíme type: Fixed-step, Solver: discrete, Fixed-step size: 1/200.

Samotné spuštění probíhá nahráním modelu `leader.slx` do vedoucího autíčka možností Deploy to Hardware. Následně načteme `follower.slx` a navolíme (nahoře v roletce) external mód. Spustíme tlačítkem Run.

Kapitola 5

Experimenty

V této kapitole prezentuji několik jednoduchých experimentů, které slouží především jako ukázka získávání dat během jízdy autíčka.

5.0.1 Udržování konstantní rychlosti - tempomat

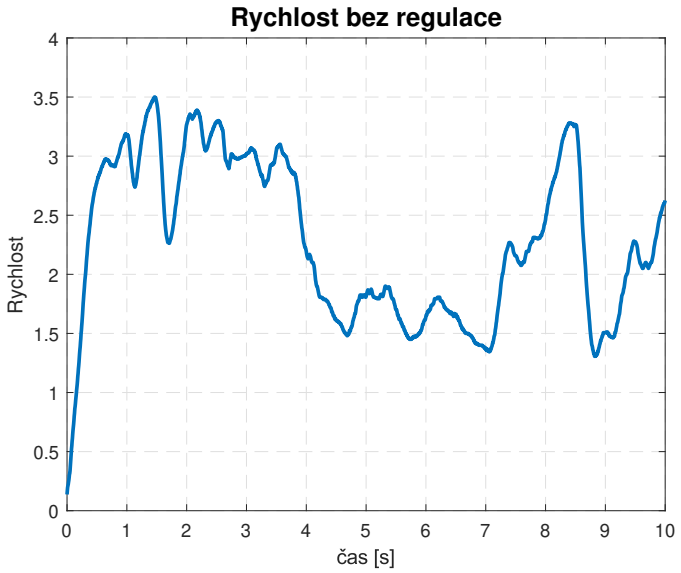
Obrázek (5.1) ukazuje průběh rychlosti autíčka po ujetí části testovacího okruhu, motor je regulace napájen konstantním PWM signálem o střídě 25%. Lze pozorovat propad v rychlosti od 4. sekundy, kdy autíčko najíždí do zatáčky.

Obrázek (5.2) ukazuje průběh rychlosti autíčka startujícího na stejném místě jako v předchozím experimentu. Tentokrát je zapojena regulační smyčka, a tak žádný velký propad v rychlosti nepozorujeme.

Obrázek (5.3) zobrazuje akční zásah regulátoru na motoru.

5.0.2 Akcelerometr

Pro test akcelerometru a gyroskopu jsem zvolil experiment, kdy se autíčko rozjíždí po rovince a projede ∼220° levotočivou zatáčku. Nezpracovaná data



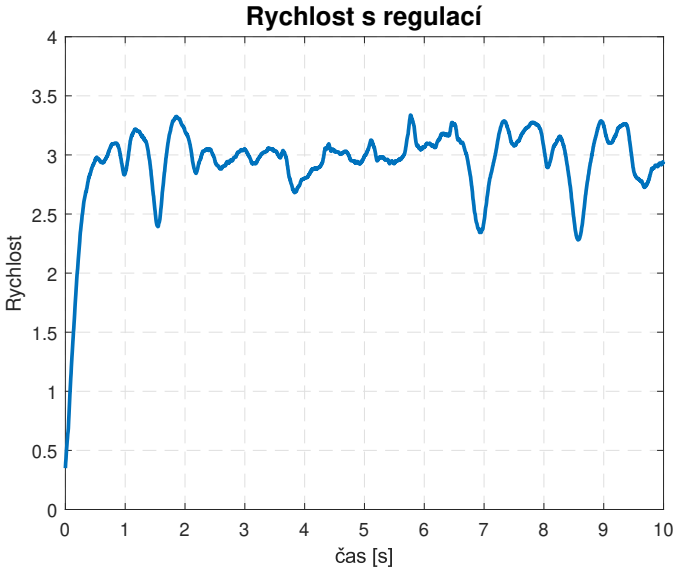
Obrázek 5.1: Průběh rychlosti bez regulace

z akcelerometru si můžeme prohléhnout na obrázku (5.4). Po filtraci plovoucím průměrem dostaneme čitelnější obrázek (5.5). Na grafech zobrazuji jen zajímavou složku x a y, složka z je konstatní a odpovídá gravitačnímu zrychlení.

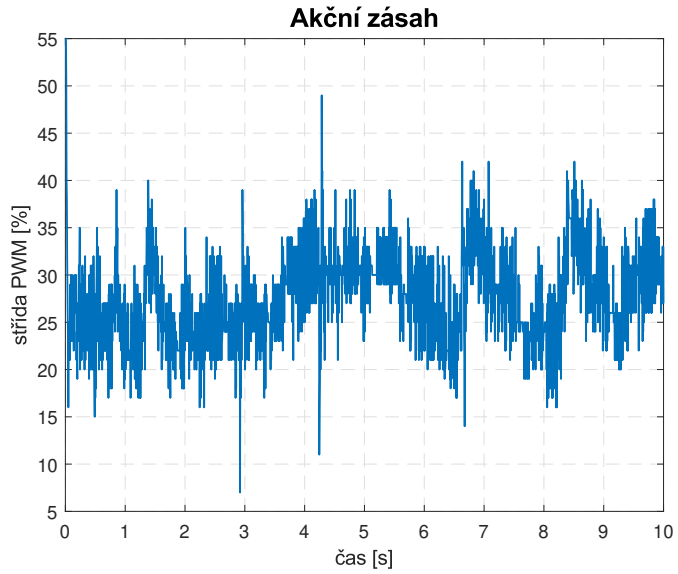
Data z gyroskopu (tentokrát bez filtrace) si můžeme prohlédnout na obrázku (5.6). Graf odpovídá experimentu, průjezd zatáčkou ovlivňuje pouze úhlovou rychlost složky x.

5.0.3 Senzor vzdálenosti

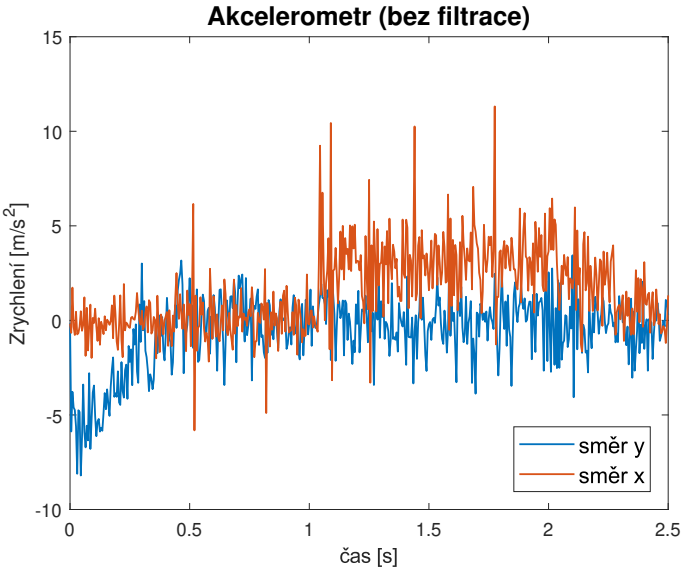
Pro test senzoru přiblížení jsem navrhl experiment, kdy autíčko jede konstantní rychlostí k překážce. První situace je na rovince (5.7), druhá v zatáčce (5.8). Z grafů vidíme, že senzor funguje rozsahu 0 - 15 cm spolehlivě.



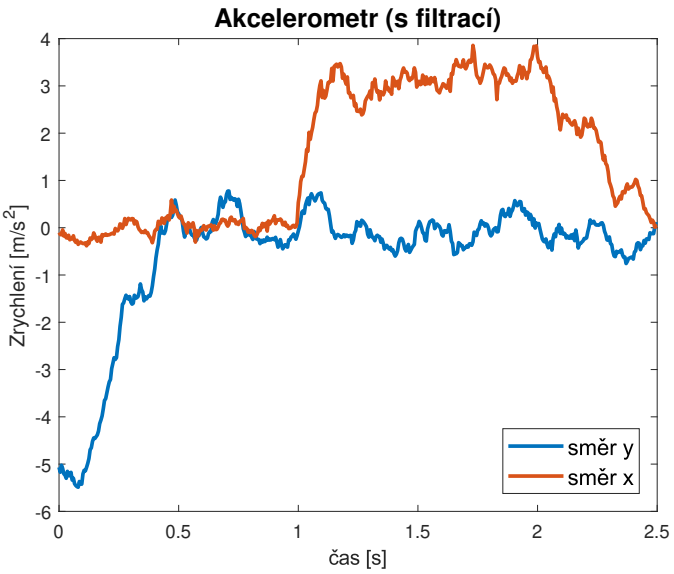
Obrázek 5.2: Průběh rychlosti s regulací



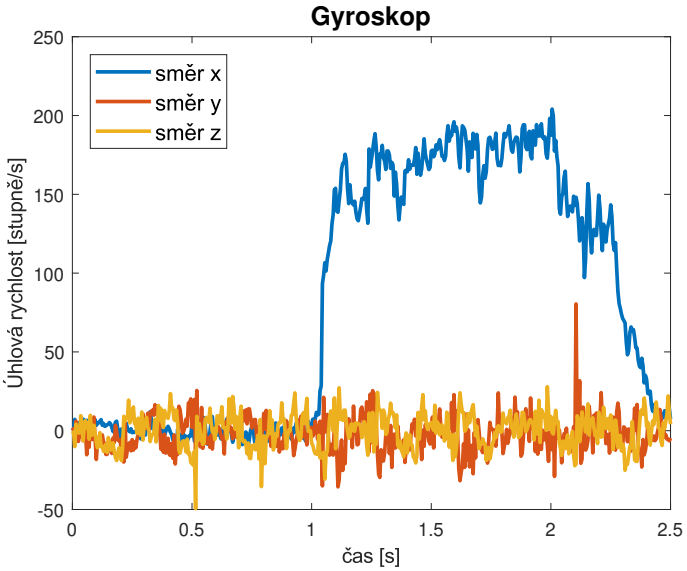
Obrázek 5.3: Akční zásah – regulace rychlosti



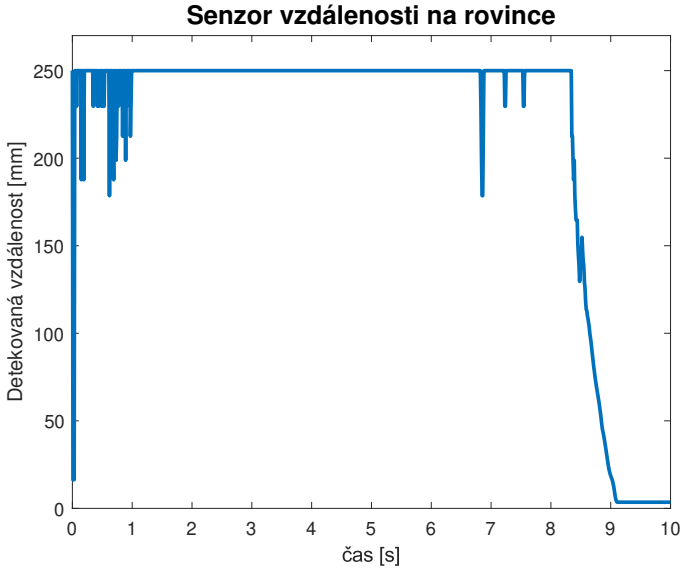
Obrázek 5.4: Akcelerometr (nefiltrovaný)



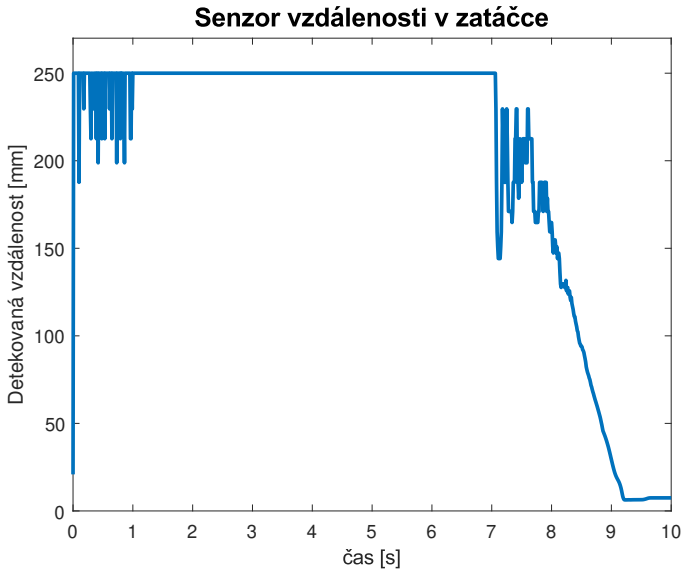
Obrázek 5.5: Akcelerometr (filtrovaný)



Obrázek 5.6: Gyroskop



Obrázek 5.7: Senzor vzdálenosti na rovince



Obrázek 5.8: Senzor vzdálenosti v zatáčce

Kapitola 6

Závěr

Cílem práce bylo navrhnout a realizovat hardware i software pro testování řídících algoritmů autonomních konvojů na autodráze. Tento problém již byl řešen minulými studenty a tato práce na ně volně navazuje.

Na míru navrhnutá a vytištěná karoserie dobře plní účel ochrany komponent. Navíc je dle mého subjektivního názoru i vizuálně atraktivnější než karoserie předchozí verze. Deska BeagleBone Blue, která je základem autíčka, se osvědčila. Pro externí senzory jsem napsal ovladače v jazyce C, které jsou pomocí bloku Matlab System Object automaticky odesílány i kompilovány na desce. Všechen ostatní kód je také automaticky generován ze bloků v Simulinku. Oproti zadání se k BeagleBone podařilo připojit pouze jeden senzor vzdálenosti, a to kvůli fixní I^2C adrese čipu.

Funkčnost senzorů jsem ověřil v jednoduchých experimentech a vybrané průběhy jsem přiložil do práce. Práce obsahuje i kapitolu zprovoznění autíčka, kde jsem shrnul postup oživení platformy. Případným zájemcům tak pomůže rychle se seznámit s platformou a soustředit se na vývoj řídících algoritmů.

Možné úskalí vidím distribuci kódu do více autíček. Bohužel Simulink v externím módu nepodporuje provoz více desek současně. V současné verzi je nutné manuálně rozeslat kódy (Deploy to hardware) do všech autíček. To je pracné a zdlouhavé, jelikož odeslání kompilace pro jedno autíčko trvá asi dvě minuty. Pokud z těchto autíček chceme dostávat online data do Simulinku, je třeba je odesílat do jednoho hlavního, kde simulace běží v externím módu. Já jsem zatím experimentoval jen se dvěma vozy, a tak to nebyl velký problém. Zde je určitě prostor pro vylepšení.

Příloha A

Literatura

[Gri17] Dr. Richard Grimmett, *Beaglebone robotic projects, 2nd edition*, Packt Publishing, 2017.

[Lá17] Martin Lád, *Experimental slotcar-based platform for distributed control of vehicular platoons*, České vysoké učení technické v Praze, 2017.

[Mar12] Dan Martinec, *Distributed control of platoons of racing slot cars*, České vysoké učení technické v Praze, 2012.

[MLH17] Ivo Herman Martin Lád and Zdeněk Hurák, *Vehicular platooning experiments using autonomous slot cars*, In 20$^{\text{th}}$ IFAC World Congress. Tolouse, France, 2017.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bečka** Jméno: **Marek** Osobní číslo: **434721**

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:

Návrh a realizace řídicího systému pro autonomní autodráhové autíčko určené pro experimentování s inteligentními konvoji

Název bakalářské práce anglicky:

Design and realization of a control system for an autonomous slot car intended for experimenting with vehicular platoons

Pokyny pro vypracování:

Navrhněte a realizujte hardware i software pro řídicí systém autonomního autodráhového autíčka, který umožní experimentování s algoritmy pro distribuované řízení konvojů (či kolon) těchto autíček. Oproti stávající verzi navrhněte nové hardwarové řešení, které bude v maximální míře využívat již hotových univerzálních vývojových kitů pro mechatronické systémy jako je kupříkladu populární BeagleBone Blue. Nové sofwarové řešení postavte do maximální možné míry na kódu automaticky generovaném z modelů v Simulinku. Dílčí úkoly:
1.) Pro autíčko samotné použijte zakoupený podvozek s motorem. Karosérii navrhněte vlastní, aby umožňovala umístění řídicí a komunikační jednotky a senzorů. Vyrobte ji s využitím 3D tisku.
2.) Základem palubního řídicího systému nechť je populární vývojový kit BeagleBone Blue. Pokud to bude možné, tak i pro měření rychlosti autíčka a vzdálenosti k předchozímu a následujícímu autíčku použijte nějaké existující modulární řešení (breakout boards), aby tak reprodukce navrženého řešení byla i pro zájemce co nejpřímočařejší.
3.) Veškerý (a nebo aspoň většina) software nechť je generován automaticky z grafických modelů v prostředí Simulink.
4.) Funkčnost předveďte s využitím alespoň dvou autíček jedoucích těsně za sebou.

Seznam doporučené literatury:

[1] Lád, Martin, Ivo Herman, and Zdeněk Hurák. 'Vehicular platooning experiments using autonomous slot cars.' In 20th IFAC World Congress. Toulouse, France, 2017.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Zdeněk Hurák, Ph.D., katedra řídicí techniky FEL

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

Datum zadání bakalářské práce: **01.02.2018** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2019**

doc. Ing. Zdeněk Hurák, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Michael Šebek, DrSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. 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

