FAKULTA
STROJNÍ
ČVUT V PRAZE

Bakalářská práce

Příprava experimentálního modelu automobilového podvozku pro počítačové řízení

Vít Pawlik

Ústav mechaniky, biomechaniky a mechatroniky
Vedoucí práce: Ing. Petr Beneš, Ph.D.

2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pawlik** Jméno: **Vít** Osobní číslo: **456380**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Příprava experimentálního modelu automobilového podvozku pro počítačové řízení

Název bakalářské práce anglicky:

Modification of an experimental car chassis model for computer control

Pokyny pro vypracování:

1) Seznamte se konstrukcí podvozkové platformy Henes-Broon.
2) Prostudujte možné přístupy k řešení vzdáleného ovládání jízdy z počítače.
3) Pro zvolené řešení připravte potřebný hardware a software.
4) Funkčnost celého systému ověřte experimentálně.

Seznam doporučené literatury:

1) Solmaz, S., Coskun, T., An automotive vehicle dynamics prototyping platform based on a remote control model car, Turkish Journal of Electrical Engineering and Computer Sciences. Vol. 21, 2013, pp. 439-451.
2) Škvor, J., Autonomní jízda zmenšeného modelu Small, diplomová práce, Praha, ČVUT v Praze, 2017.
3) Vrbský, L., Experimentální model vozidla s asistentem pro couvání s přívěsem, bakalářská práce, Praha, ČVUT v Praze, 2016.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Beneš, Ph.D., odbor mechaniky a mechatroniky FS

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

Datum zadání bakalářské práce: **25.04.2018** Termín odevzdání bakalářské práce: **17.08.2018**

Platnost zadání bakalářské práce: _____

Ing. Petr Beneš, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Milan Růžička, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. 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

Anotační list

Jméno autora:	Vít Pawlik
Název bakalářské práce:	Příprava experimentálního modelu automobilového podvozku pro počítačové řízení
Anglický název:	Modification of an experimental car chassis model for computer control
Akademický rok:	2017/2018
Ústav/odbor:	Ústav mechaniky, biomechaniky a mechatroniky, Odbor mechaniky a mechatroniky
Vedoucí práce:	Ing. Petr Beneš, Ph.D.
Bibliografické údaje:	Počet stran: 49 Počet obrázků: 23 Počet příloh: 1x CD
Klíčová slova:	počítačové řízení, model automobilu, MATLAB, Simulink, Raspberry Pi, Python, Arduino
Keywords:	computer control, car model, MATLAB, Simulink, Raspberry Pi, Python, Arduino
Abstrakt:	Cílem této práce je navrhnout, realizovat a experimentálně ověřit řešení pro počítačové řízení experimentálního modelu automobilového podvozku. Je v ní prozkoumán vhodný hardware, jako platformy Raspberry Pi a Arduino a možnosti jejich použití.
Abstract:	The goal of this thesis is to design, implement and verify a solution for computer control system for an experimental car chassis model. This work explores suitable hardware, such as Raspberry Pi and Arduino and their scope of use.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje.

V Praze dne 16. srpna 2018 Vít Pawlik

Abstrakt

Cílem této práce je navrhnout, realizovat a experimentálně ověřit řešení pro počítačové řízení experimentálního modelu automobilového podvozku. Je v ní prozkoumán vhodný hardware, jako platformy Raspberry Pi a Arduino a možnosti jejich použití.

Klíčová slova počítačové řízení, model automobilu, MATLAB, Simulink, Raspberry Pi, Python, Arduino

Abstract

The goal of this thesis is to design, implement and verify a solution for computer control system for an experimental car chassis model. This work explores suitable hardware, such as Raspberry Pi and Arduino and their scope of use.

Keywords computer control, car model, MATLAB, Simulink, Raspberry Pi, Python, Arduino

Obsah

Úvod

Motivací pro vznik této práce byla potřeba experimentálního zařízení pro testování algoritmů souvisejících s autonomním řízením automobilu. Pro tento účel by bylo možno použít zařízení založené na malém autu na dálkové ovládání, to by však přineslo problémy například s použitím kamer, dálkoměrů a dalších senzorů, které pro své fungování a použití potřebují určité minimální rozměry a vzdálenosti, a znesnadnilo simulování reálných situací při experimentech. Proto jsme se rozhodli použít elektrické dětské vozítko, které je výrazně větší než RC modely, což zajistí jak bezproblémové umístění potřebné elektroniky na zařízení, tak i přiblížení k reálným automobilům.

Hlavním cílem této práce je zjistit, jaké jsou možnosti počítačového řízení takového experimentální zařízení a jaký hardware a software je k tomu možné použít. Poté z dostupných možností zvolit nejvhodnější variantu a vytvořit k ní potřebné programy pro řízení. Vedlejším cílem je popsat postup vedoucí k vytvoření těchto programů, což může v budoucnu sloužit jako návod pro pokračování v rozvíjení tohoto experimentální zařízení nebo pro vytvoření nového.

V první části popíšu použitý podvozek, jeho mechanické vlastnosti a elektronické součásti. V druhé části se budu zabývat realizací řízení a možnostmi a praxí realizace řízení s použitím počítače. Dále popíšu některé zástupce používaného hardware a jejich možnosti použití. Ve třetí části provedu výběr zařízení pro realizaci ovládání, popíši postup zprovoznění vybraného hardware a demonstruji některé základní příklady použití. Zabývat se budu jak použitím prostředí Matlab/Simulink, tak i dalšími možnostmi ovládání zvoleného hardware. Následně vypracuji programy a kód potřebný pro příjem příkazů z počítače a obsluze elektronických součástí podvozku. Ve čtvrté části provedu experimentální ověření funkce navrženého řešení.

Kapitola 1

Podvozková platforma Henes Broon

Použitá podvozková platforma vznikla z komerčního dětského elektrického vozítka s modulární strukturou, ze kterého byla sundána veškerá kapotáž a demontována řídící elektronika. Zůstal tak pouze podvozek s koly, zavěšením, zatáčením, motory a baterií. Fotky vozítka, podvozku a součástí jsou na konci této kapitoly.

1.1 Ovládání zatáčení

Zatáčení je poháněno stejnosměrným motorem s vyvedenými dvěma vodiči. Rotace motoru je přes převody a hřeben převedena na posuv tyče, která natáčí kola. Aby tímto motorem mohlo být provedeno natočení kol na určitý úhel, je nutné k němu připojit senzor pro zpětnou vazbu řídícímu systému. Tato zpětná vazba je realizována jednoduchým senzorem natočení, který je tvořen potenciometrem.

Motor zatáčení má vyvedeny dva vývody. Při přivedení napětí +12V na zelený vodič a nulového napětí na bílý vodič zatáčí motor ve směru doprava a v opačném případě doleva. Detail motoru je na obrázku 1.3

1.2 Ovládání pohonu

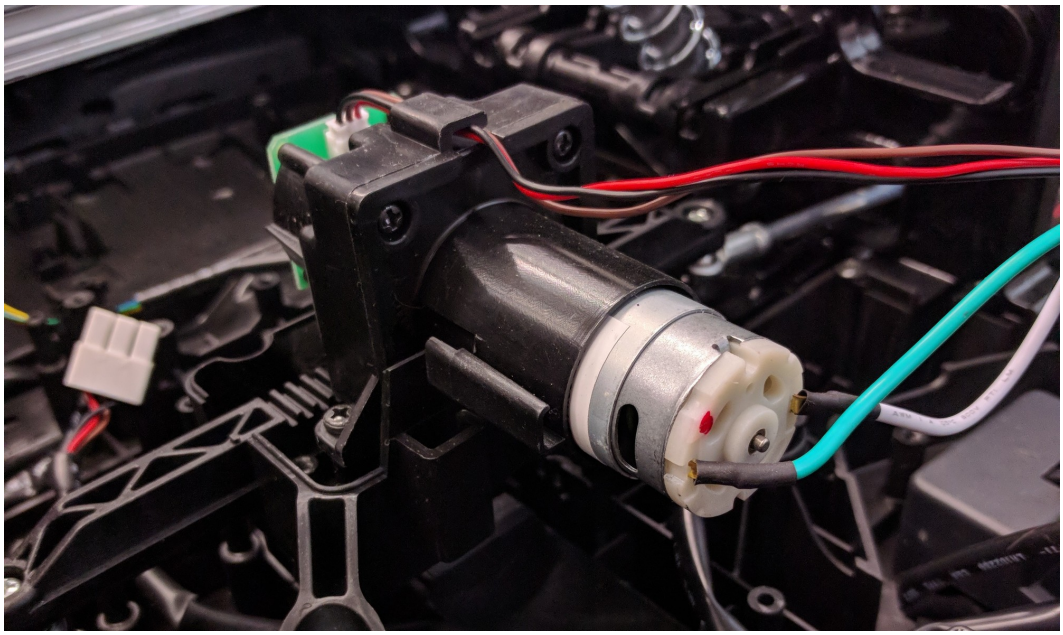
S původním řídícím systémem pohon funguje tak, že při stlačení akcelerátoru je na motory přivedeno napětí a ty začnou vyvíjet kroutící moment, úměrný stlačení akcelerátoru. Po chvíli se systém řízení přepne do zpětnovazebného režimu, který pro měření rychlosti využívá Hallovy sondy na levém zadním kole. Pokud tedy počáteční kroutící moment byl příliš malý na to, aby vozítko uvedl do pohybu, a kola se nezačala točit, řídící systém zvýší kroutící moment a udržuje konstantní rychlost otáčení kol.

Pohon je tvořen dvěma motory, každé zadní kolo je vybaveno jedním motorem s převodem. Z každého motoru jsou vyvedeny dva vodiče, které jsou spojeny rozdvojkou do jednoho vývodu s dvěma kontakty. Po přivedení napětí +12V na červený vodič a nulového napětí na žlutý vodič se oba motory otáčejí ve směru dopředu, při opačném zapojení se otáčí dozadu. Na obrázku 1.4 je detail motoru pohonu.

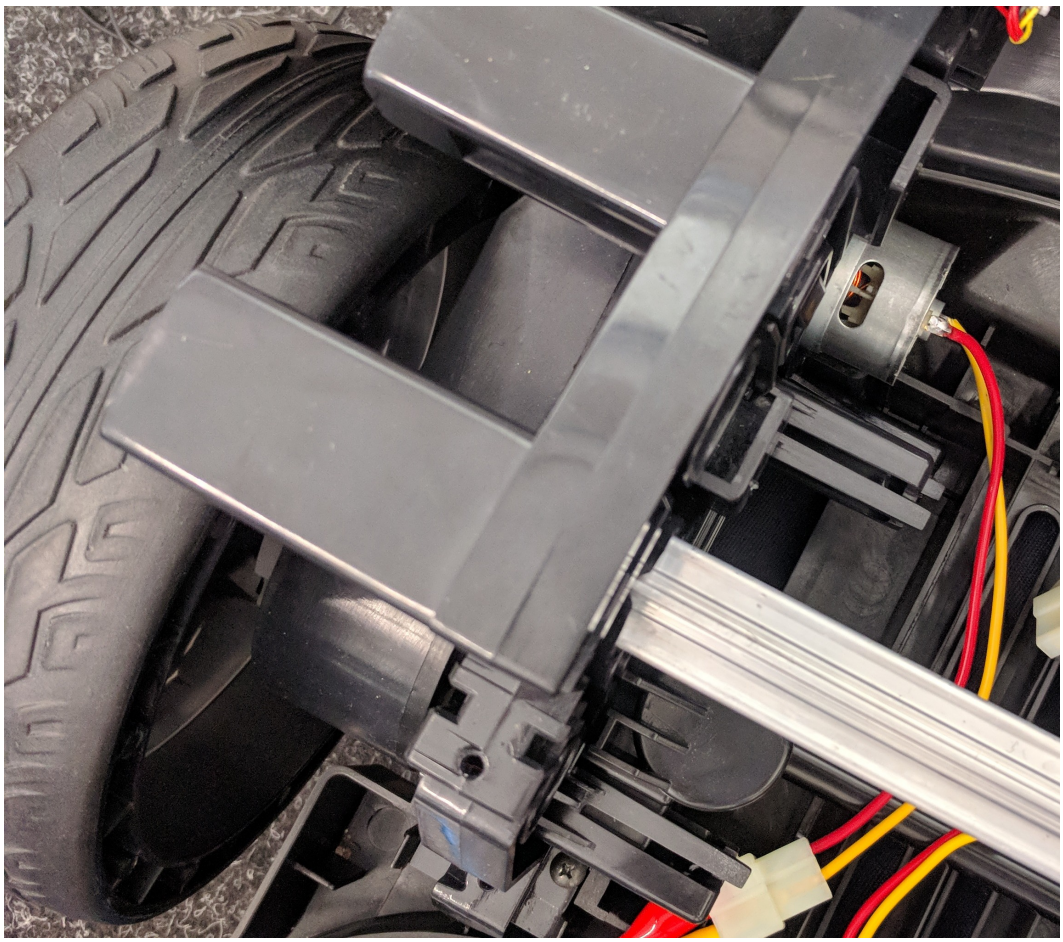
Podle specifikace výrobce mají motory nominální výkon 120 W, a tedy nominální proud 10 A.

Obrázek 1.1: Henes Broon F830. Převzato z [1]

Obrázek 1.2: Henes Broon F830 - podvozek. Převzato z [2]



Obrázek 1.3: Detail motoru zatáčení s potenciometrem a hřebenovým převodem rotace na posuv



Obrázek 1.4: Detail pravého motoru pohonu a převodu na kolo

Kapitola 2

Počítačové řízení

Existuje velmi mnoho možností realizace řízení. Publikace [3, The Interface] vyjmenovává a charakterizuje tři základní typy hardware:

- osobní počítač (PC),
- jednodeskový počítač,
- mikrokontrolér.

PC nabízí vysoký výkon, velkou kapacitu paměti a velké množství portů, které ale většinou není možné využít přímo k ovládání, ale je nutné přidat specializované zařízení, které toto umožní. Je však vhodné k použití v roli řídícího prvku (master), který ovládá další hardware (slave), který je k němu připojen přes některé rozhraní. Další nevýhodou PC jsou obvykle výrazně větší rozměry a nutnost síťového napájení.

Jednodeskový počítač je kompletní počítač, který je tvořen pouze jednou deskou plošných spojů. Na této desce je System on a Chip (SoC), což je jeden integrovaný obvod, obsahující všechny základní prvky pro fungování počítače, tedy procesor, paměť, grafické jádro a obvody pro obsluho portů a rozhraní. Může také obsahovat porty Universal Serial Bus (USB) nebo General-purpose Input/Output (GPIO). GPIO piny tvoří univerzální rozhraní, umožňující rozšíření funkcionalit počítače prostřednictvím digitální komunikace s dalšími obvody. Pro obsluhování portů a rozhraní na desce bývají dedikované integrované obvody. Nevýhodou jednodeskových počítačů je relativně nižší výkon, který však přináší nízké nároky na napájení. Na jednodeskovém počítači většinou běží kompletní operační systém, jako třeba některá distribuce Linuxu nebo Windows 10 IoT.

Mikrokontrolér je nejjednodušší a nejméně výkonné zařízení, obsahující pouze procesor, paměti pro program a pro dočasná data, a dále obvody pro obsluhu vstupních a výstupních portů. Na desce vedle mikrokontroléru dále bývá zdroj hodinového signálu, napájecí obvody, může také obsahovat analogově digitální převodníky (ADC). Tyto kontroléry bývají programovány nízkoúrovňovými jazyky symbolických adres (assembly language) nebo jazykem typu C, který je přeložen překladačem (kompilátorem). S nízkým výkonem souvisí i nízká spotřeba a nízké nároky na napájení, takže je mnohdy možné mikrokontrolér provozovat například s napájením z baterie.

V následující části popíšu některé nejběžnější zástupce jednodeskových počítačů [4] a mikrokontrolérů [5].

2.1 Arduino Uno

Obrázek 2.1: Arduino Uno Rev3. Převzato z [6]

Arduino Uno je deska s mikrokontrolérem ATmega328P [7] s taktem 16 MHz a pamětí 32 kB, z níž je 0.5 kB obsazeno zavaděčem. Má provozní napětí 5 V a napájena může být buď z USB portu nebo z napájecího jack konektoru. Obsahuje 16 digitálních Input/Output (I/O) pinů, z nichž 6 podporuje pulzně-šířkovou modulaci (PWM), a 6 analogových I/O pinů. Programuje se prostřednictvím Arduino IDE upravenou verzí jazyka C, Wiring. [6]

Komunikace s PC probíhá prostřednictvím sériové linky pomocí USB spojení. Pro bezdrátovou komunikaci je nutné k Arduinu připojit modul s radiovým, Wi-Fi nebo Bluetooth vysílačem a přijímačem. Některé verze Arduina již obsahují Wi-Fi modul přímo na desce.

Kvůli nízkému výkonu je Arduino vhodné pro jednoduché elektronické projekty, které nevyžadují složité výpočty. Kromě Una existují i další verze, například Mega, která obsahuje více portů a druhý sériový port, nebo třeba Nano, které má stejný mikrokontrolér jako UNO, ale má jednodušší napájecí obvod a menší velikost.

Jedním z možným použití Arduina je jako rozhraní mezi (jednodeskovým) počítačem a senzorem s digitálním nebo analogovým výstupem.

2.2 NodeMCU (ESP8266)

Nejprve byl čip ESP8266 prezentován jako levný Wi-Fi modul pro Arudino, ale poté si komunita uvědomila, že lze postavit desku kolem samotného ESP8266, bez potřeby dalšího mikrokontroléru, neboť sám o sobě obsahuje ovladač pro sériovou komunikaci potřebný pro programování, regulátor napájení i množství I/O pinů, a tak vzniklo NodeMCU. [8]

Procesor má takt 80 MHz, 96 kB operační paměti a 4 MB paměti flash. Kromě programování ve Wiring, jako Arduino, nabízí dále programování v jazyce Lua. Dále nabízí připojení různých rozšíření v podobě shieldů. [9]

2.3 STM32

Mikrokontroléry STM32 s architekturou ARM stojí někde mezi Raspberry Pi a 8bitovými mikrokontroléry typu Arduino. Výkonem nedosahují k Raspberry Pi, avšak nabízí vyšší výkon než Arduino, při zachování úzké vazby na hardware a spotřeby v řádech mikroampér. Komplexnost platformy ale přináší náročnější konfiguraci. Existuje velké množství variant

STM32, mezi kterými jsou varianty zaměřené na výkon, zaměřené na nízkou spotřebu, nabízející bezdrátovou komunikaci, nebo nabízející přesné ADC. [10]

2.4 Raspberry Pi

Platforma Raspberry Pi vznikla na půdě Cambridské univerzity ve snaze dr. Ebena Uptona vytvořit malý počítač, který by byl velmi levný a který by šel použít k výuce programování, a vzbudit v mladé generaci studentů zájem o informatiku a programování. Z této iniciativy v roce 2012 vznikl jednodeskový počítač s cenovkou £25 (u nás okolo 600 Kč) osazený SoC typu ARM, který se používá v mobilních telefonech, s ethernetovým konektorem, konektory USB a HDMI grafickým výstupem, a také GPIO piny a proprietárním konektorem pro připojení kamery. Tento jednodeskový počítač s operačním systémem Linux (nejčastěji upravenou verzí distribuce Debian, Raspbian) pak lze použít jako osobní počítač. [11, Raspberry Pi]

2.4.1 Raspberry Pi 3 Model B+

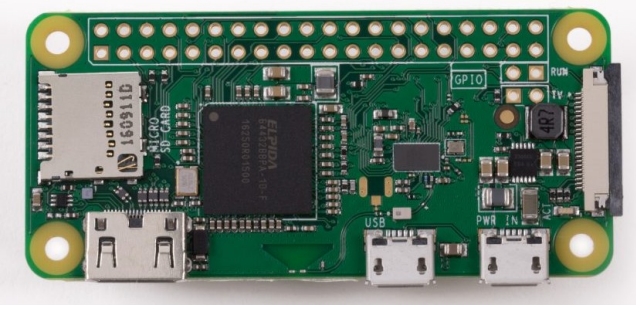
Aktuální verze 3 Model B+ (vydaná 13. března 2018) obsahuje výkonnější 64bitový 4jádrový procesor, dále taky disponuje Wi-Fi a Bluetooth konektivitou, čtyřmi porty USB, jedním ethernetovým a CLI portem pro displej. Pro výstup je k dispozici jeden HDMI port, jeden TRRS jack pro audio a video a CLI port pro modul s kamerou. Dále obsahuje 40 GPIO portů. [12]

Obrázek 2.2: Raspberry Pi 3 Model B+. Převzato z [12]

2.4.2 Raspberry Pi Zero

Zero je velmi malá a levná[1] verze Raspberry Pi. Obsahuje jednojádrový 32bitový procesor s taktem 1 GHz a 512 MB RAM, microHDMI výstup, jeden univerzální microUSB port a jeden dedikovaný pro napájení. Nedisponuje Ethernetovým portem, ale v roce 2017 bylo vydáno Pi Zero W, které nabízí Wi-Fi konektivitu [14].

[1] Vydáno bylo jako bezplatná příloha vánočního čísla časopisu *The MagPi* v roce 2015 a následně se začalo prodávat s cenovkou $5. Časopis *The MagPi* je oficiální magazín pro uživatele Raspberry Pi. Velikost Pi Zero je 65 mm x 30 mm. [13, Raspberry Pi Zero: the $5 computer]



Obrázek 2.3: Raspberry Pi Zero W. Převzato z [13]

2.4.3 GPIO

Existuje více možností ovládání GPIO pinů. V článku *Sparkfun: Raspberry gPIo* [15] jsou popsány některé nejpoužívanější, což je buď použití programovacího jazyka C/C++ s knihovnou WiringPi, nebo použití programovacího jazyka Python a knihovny RPi.GPIO. Tyto piny mohou plnit funkci vstupu nebo výstupu. Dále obsahují integrované rezistory, které lze použít jako pull-up nebo pull-down rezistory, a také podporují funkci interrupt, která umožňuje zaručení reakce procesoru na událost (např. vykonání akce při změně stavu z 0 na 1 na některém vstupu). Také umožňují komunikaci se zařízením či integrovaným obvodem s protokoly I^2C [14, GPIO] nebo SPI [14, SPI].

2.4.4 Matlab/Simulink

Společnost MathWorks nabízí k prostředí MATLAB a Simulink balíky s podporou pro Raspberry Pi. Tyto balíky umožňují ovládání funkcí Raspberry Pi, buď z prostřední MATLAB pomocí příkazů, nebo z prostředí Simulink prostřednictvím funkčních bloků. Takto lze ovládat GPIO piny, jejich prostřednictvím dále I^2C a SPI zařízení nebo servo motor. Dále umožňují použití *CameraBoard*, což je oficiální deska s kamerou používající standardizovaný CSI konektor. Dále nabízí podporu pro USB kamery, přehrávání nebo nahrávání audia použitím komponenty ALSA a síťovou komunikaci prostřednictvím protokolů UDP nebo TCP/IP. Simulink při použití toolboxu Simulik Coder umožňuje nahrání vytvořeného modelu přímo do zařízení a jeho samostatný běh. Všechny funkce a použití těchto balíků jsou detailně popsány v [16, MATLAB Support Package for Raspberry Pi Hardware] a [16, Simulink Support Package for Raspberry Pi Hardware].

2.5 Rozšíření základní možností

Možnosti Raspberry Pi i Arduina lze dále rozšířit přidáním další desky, která se připojí pomocí GPIO pinů. Toto se v případě Arduina nazývá *shield* a v případě Raspberry Pi *HAT*. Toto rozšíření může být jednoduché pájivé nebo nepájivé pole s vyvedenými GPIO piny, zvuková karta s kvalitním DAC, síťová karta, karta pro bezdrátovou komunikaci (např. GSM), obvody pro ovládání motorů nebo třeba kompletní PLC. Ovládání tohoto zařízení pak probíhá prostřednictvím GPIO, užitím protokolů I^2C, SPI nebo sériové linky. [17] [18]

2.6 Beaglebone Black

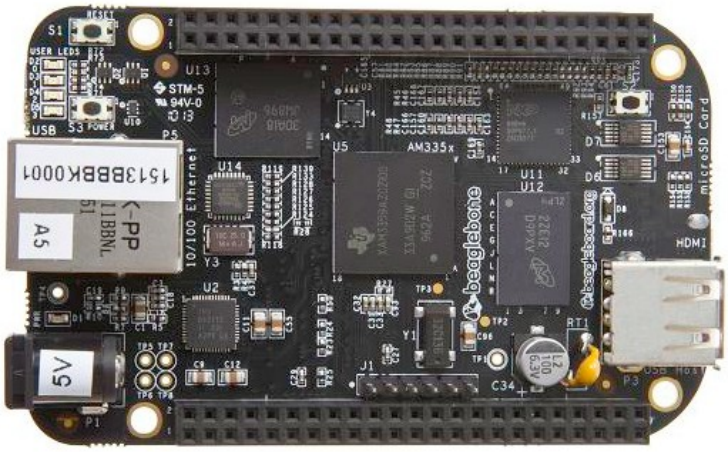
Stejně jako Raspberry Pi je Beaglebone Black (BBB) jednodeskový počítač. Obsahuje procesor architektury ARM s taktem 1 GHz a 512 MB RAM, 2 GB flash paměti a slot na microSD kartu. Oproti Raspberry Pi má BBB po koupení operační systém již nahraný ve vnitřní paměti. Dále obsahuje standardní napájecí jack konektor. Sice nabízí pouze jeden port

pro přípojení USB zařízení, ale zato disponuje 92 GPIO piny, z nichž některé nabízejí vstup s ADC, což umožňuje připojení různých senzorů a zařízení s analogovým výstupem. Dále je k dispozici miniHDMI port pro výstup videa. Velké množství GPIO pinů činí BBB ideálním pro rozsáhlejší projekt. [19]

Velkou výhodou BBB je Programmable Real-Time Unit (PRU), neboli programovatelná jednotka reálného času, což je 32bitový procesor s taktem 200MHz, který umožňuje ovládání I/O portů a periferií a přístup paměti v reálném čase [20, PRU-ICSS Resources].

Ve verzi Wireless nabízí místo Ethernetového portu Wi-Fi konektivitu.

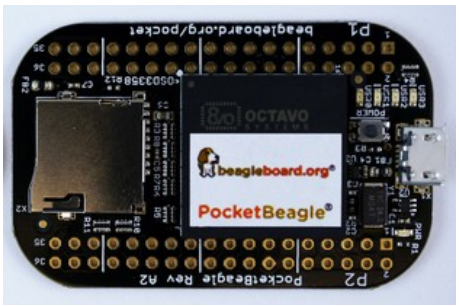
MathWorks nabízí balík hardwarové podpory *MATLAB Support Package for BeagleBone Black Hardware* pro ovládání BBB z prostředí MATLAB a taky balík *Embedded Coder Support Package for BeagleBone Black Hardware*, umožňující běh Simulink modelů na BBB [16].



Obrázek 2.4: Beaglebone Black. Převzato z [20]

2.7 PocketBeagle

PocketBeagle je zmenšená verze Beaglebone Black o velikosti 56 mm x 35 mm, neobsahující téměř žádné porty pro konektory, pouze GPIO porty, microUSB pro napájení a programování a slot pro microSD kartu. Stejně jako Raspberry Pi neobsahuje flash paměť, operační systém je umístěn na microSD kartě.



Obrázek 2.5: PocketBeagle. Převzato z [20]

Kapitola 3

Příprava hardware a software

3.1 Volba hardware

Jako základní prvek pro realizaci použiji Raspberry Pi vybavené Wi-Fi konektivitou. Není třeba specifikovat verzi, neboť všechny aktuálně prodávané verze jsou kompatibilní.[2] Použiji nejnovější a nejvýkonnější verzi 3 Model B+ pro finální realizaci a menší verzi Zero W pro testování dílčích částí.

Hlavním důvodem volby Raspberry Pi je možnost použití množství různých přístupů při programování, ať už je to z prostředí MATLAB/Simulink, některého z běžných programovacích jazyků nebo použitím vizuálního programovacího jazyka [21].

K řízení zatáčení je nutné číst analogovou hodnotu natočení potenciometru, a proto k tomu použiji Arduino vybavené ADC. Navíc Arduino nabízí několik možností komunikace s Raspberry Pi.

3.2 Oživení Raspberry Pi

Raspberry Pi se prodává jako *barebones*, tedy pouze deska, bez jakýchkoliv periferií a bez operačního systému. Nabízí více možností bootování [14, Raspberry Pi Boot Modes], nejběžnější je bootování z SD karty s některou distribucí Linuxu. Jako operační systém použiji Raspbian, což je verze distribuce Debian upravená speciálně pro Raspberry Pi. Rovněž je vhodná pro použití s MATLABem.

Použiji Raspbian ve verzi Lite, která obsahuje pouze základní programy. Neobsahuje například grafické rozhraní, což ale nevadí, neboť standardním způsobem používání je připojení se z jiného počítače prostřednictvím SSH a ovládání textovým terminálem. Tento protokol je běžný na linuxových distribucích a od konce 2017 ho nativně podporuje i Windows 10. V případně nutnosti je možné grafické prostředí doinstalovat.

Pro oživení je potřeba následující:

- Některá verze Raspberry Pi, vhodný napájecí zdroj a SD karta.
- PC se čtečkou SD karet a vhodný program pro zapsání obrazu Raspbianu na SD kartu[3].

[2] Při použití microSD karty s Raspbianem stačí kartu vyjmout z jednoho Raspberry Pi a vložit do druhého a vše bez problému funguje naprosto stejně. Jediná nekompatibilita by mohla nastat při použití původního Raspberry Pi z roku 2012, které má pouze 26 GPIO pinů. Nicméně při použití pouze prvních 26 pinů kompatibilita zůstává.

[3] Například svobodný program Win32 Disk Imager `https://sourceforge.net/projects/win32diskimager/`, nebo modernější program Etcher `https://etcher.io/`

- Obraz operačního systému Raspbian[4,5].
- Ethernetový kabel, pokud Raspberry Pi nedisponuje Wi-Fi konektivitou.

3.2.1 Postup oživení

Pro základní oživení je nutné stáhnout obraz Raspbianu a extrahovat obraz `img` ze zip souboru a zapsat ho na SD kartu.

Zapsáním obrazu jsou na SD kartě vytvořeny dva oddíly, z nichž pouze jeden je pro Windows čitelný. Proto pro případné formátování SD karty je vhodné použít specializovaný program[6], například podle návodu *How To Format Pi SD Cards Using SD Formatter*[22].

Pokud vše proběhlo bez problémů, z takto připravené SD karty by Raspberry Pi mělo nabootovat. V tomto stavu by bylo možné Raspberry Pi použít s externím monitorem a klávesnicí. Já však chci Raspberry Pi ovládat vzdáleně přes síť prostřednictvím Secure Shell (SSH). Z důvodů bezpečnosti je SSH implicitně zakázáno [23] a je nutné ho povolit. Toto se provede snadno, stačí v oddílu `boot` na SD kartě vytvořit prázdný soubor s názvem `ssh`.

Dále je vhodné přidat nastavení Wi-Fi. Toto se provede vytvořením souboru `wpa_supplicant.conf` na SD kartě. Příklad tohoto souboru je uveden jako kód 3.1.

```
1  ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
2  update_config=1
3  country=CZ
4
5  network={
6          ssid="raspberry_wifi"
7          scan_ssid=1 #pokud je SSID neviditelne
8          psk="raspberry"
9          priority=1
10         id_str="primarni"
11 }
12
13 network={
14         ssid="raspberry_zalozni"
15         psk=e4df261a952e18235bf627664c3205e8b7
           ↪  6c0f9530526b6669f6e683150029f6
16         priority=2
17         id_str="zalozni"
18 }
```

Kód 3.1: wpa_supplicant.conf

[4] `https://www.raspberrypi.org/downloads/raspbian/`

[5] Je možné SD kartu nakonfigurovat přímo balíkem hardwarové podpory v MATLABu, avšak ten v současnosti (červen 2018) obsahuje starší verzi Raspbianu, a například Raspberry Pi 3 Model B+ s ní vůbec nenabootuje.

[6] Ve Windows je možné kartu naformátovat programem *Správa počítače*. Nejprve je nutné odstranit oba svazky na SD kartě, a následně vytvořit nový, typu FAT32.

Tento soubor obsahuje název Wi-Fi sítě `ssid` a heslo `psk`[7]. Při používáni lze pak tento soubor editovat z terminálu příkazem:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Soubory `ssh` a `wpa_supplicant.conf` jsou přiloženy na médiu ve složce `raspberrypi/boot`.

Po těchto operacích je SD karta připravena, a po vložení do Raspberry Pi a připojení zdroje by mělo Raspberry Pi nabootovat a poté se připojit k zadané Wi-Fi síti. Pro testování Raspberry Pi doporučuji použít funkci Windows *Mobilní hotspot* a vytvořit síť s názvem a heslem odpovídajícím konfiguraci Raspberry Pi, ke které se tak automaticky připojí, a v nastavení *Mobilního hotspotu* se pak zobrazení IP adresa připojeného zařízení. Následně se k němu lze připojit z konzole nebo příkazové řádky příkazem `ssh pi@ip_adresa`, kde `ip_adresa` je IP adresa Raspberry Pi.

V příloze D jsou uvedeny některé užitečné příkazy.

3.3 Použití Raspberry Pi v prostředí MATLAB/Simulink

3.3.1 Zprovoznění podpory

Po oživení podle návodu v sekci 3.2.1 je pro použití s MATLABem a Simulinkem nutno doinstalovat podporu. To lze udělat provedením série příkazů:

```
cd ~
wget https://github.com/mathworks/Raspbian_OS_Setup/
↪  raw/master/RaspbianOSsetup_Stretch.sh
sudo chmod +x RaspbianOSsetup_Stretch.sh
sudo ./RaspbianOSsetup_Stretch.sh
```

Kód 3.2: Doinstalování podpory MATLAB/Simulink

- `cd ~`: přesun do domovského adresáře,
- `wget ...`: stažení konfiguračního skriptu z GitHubu MathWorks,
- `sudo`: provedení příkazu nebo spuštění programu s oprávněním správce,
- `chmod +x`: povolení spuštění skriptu (přidání spustitelného příznaku),
- `./RaspbianOSsetup_Stretch.sh`: spuštění skriptu.

Tento skript stahuje, kompiluje, instaluje a konfiguruje velké množství prostředí, programů a funkcí, a provedení může trvat velmi dlouho.

[7] Místo hesla je možno použít jeho zašifrovanou podobu. Tu jde získat pod Linuxem příkazem `wpa_passphrase [ ssd_site ] [ heslo_site ]`, případně online utilitou `https://www.wireshark.org/tools/wpa-psk.html`. Toto řešení je vhodné pro zabránění zveřejnění hesla, když k zařízení nebo konfiguračním souborům má přístup někdo další.

3.3.2 Ukázka použití Raspberry Pi v prostředí MATLAB

Pro ovládání Raspberry Pi z MATLABu je nutné nejprve stáhnout a nainstalovat balík hardwarové podpory *MATLAB Support Package for Raspberry Pi Hardware*, a to buď přímo v MATLABu z menu *Home →Environment →Add-Ons →Get Hardware Support Package*, nebo z webu [24]. Obsažené funkce jsou uvedeny v dokumentaci [16, MATLAB Support Package for Raspberry Pi Hardware Functions - By Category].

Nejprve je nutné připojit Raspberry Pi k MATLABu. K tomu je zapotřebí, aby Raspberry Pi bylo ve stejné lokální síti, jako PC s MATLABem, a je nutné znát jeho IP adresu. Dále budu předpokládat, že IP adresa Raspberry Pi je `192.168.1.163`. Připojení se provede vytvořením objektu, například s názvem *rpi*:

```
rpi = raspi('192.168.1.163', 'pi', 'raspberry')
```

Pokud se funkce `raspi` zavolá bez parametrů, použijí se poslední použité parametry.

V následujících kódech demonstruji použití některých základních funkcí.

```
rpi_ip='192.168.1.163';
rpi = raspi(rpi_ip, 'pi', 'raspberry')

while 1
  rpi.writeLED('led0',1)
  pause(0.5)
  rpi.writeLED('led0',0)
  pause(0.5)
end
```

Kód 3.3: Blikání LED diodou na desce Raspberry Pi z MATLABu

```
rpi = raspi;
% nastaveni frekvence PWM signalu na pinu 25
setPWMFrequency(rpi, 25, 2000)
% zapnuti PWM signalu na pinu 25 s duty-cycle o hodnote 42 %
writePWMDutyCycle(rpi, 25, 0.42)
```

Kód 3.4: Generování PWM signálu na pinu 25

```
rpi = raspi;
servo_pin = 18;
% vytvoreni objektu serva
rpi_servo = servo(rpi, servo_pin)
uhel=90;
% nastaveni pozice serva
writePosition(rpi_servo,uhel)
```

Kód 3.5: Ovládání serva na pinu 18

```
rpi = raspi;
% vytvoreni objektu kamery
kamera = cameraboard(rpi,'Resolution','1920x1080', 'Rotation',0,'ImageEffect','denoise')
figure
img = snapshot(kamera);
imagesc(img)
drawnow
```

Kód 3.6: Ovládání modulu kamery na Raspberry Pi z MATLABu

3.4 Použití Raspberry Pi v prostředí Simulink

Nejprve je nutné nainstalovat balík podpory *Simulink Support Package for Raspberry Pi Hardware*, opět přímo z MATLABu, nebo z webu [25]. Bloky, které tento balík nabízí, jsou uvedeny v dokumentaci [16, Simulink Support Package for Raspberry Pi Hardware > Modeling]. Jsou to například bloky pro řízení logického nebo PWM signálu na pinech, ovládání serva, ovládání kamery nebo UDP a TCP komunikaci. Dále nabízí možnost použití *S-Function*, pomocí které lze naprogramovat ovladač pro libovolné zařízení připojené k Raspberry Pi.

Před spuštěním modelu je nutné nastavit cílový hardware. To se provede přes menu modelu *Tools →Run on Target Hardware →Options…*, kde je potřeba pod *Hardware board* zvolit možnost *Raspberry Pi*. Model může běžet ve dvou různých módech: *externí* a *nasazený*[8]. V externím módu je možné některé parametry bloků modelu upravovat i při běhu. Běh v nasazeném módu se provede funkcí *Deploy to hardware*, kdy je model zkompilován na cílovém zařízení a spuštěn. Takto běžící model běží, dokud není spuštěn další model nebo model není zastaven vytvořením objektu `raspberrypi` a použitím metody `stopModel('jmeno_modelu')`[9], případně vypnutím zařízení. Funkcí `addToRunOnBoot` lze povolit spuštění modelu při startu zařízení.

3.4.1 Úskalí řízení Raspberry Pi ze Simulink

Jedním z problémů je, že s výchozím nastavením model neběží v reálném čase, ale s krokem dosahujícím až tisíců sekund. Toto lze částečně ošetřit pevným nastavením velmi malého kroku, případně vložením MATLAB funkce, ve které bude použita funkce `pause(t)`. Příklad takové funkce je v kódu 3.7. Funkce má jako vstup dobu pauzy v sekundách. Ukázka použití je na obrázku 3.1. Stále však nelze docílit časově deterministického běhu, což například znemožňuje použití bloku *Signal Builder* a podobných bloků, protože čas poběží po náhodných skocích.

Běhu v reálném času by mělo být možné dosáhnout použitím bloku *Real-Time Sync* z toolboxu *Simulink Desktop Real-Time* [16], ale ani tak se mi toho nepodařilo dosáhnout, a při testování na dostupném hardware model běžel naopak velmi pomalu. To je způsobeno nejspíš nekompatibilitou. Běhu v reálném čase lze patrně dosáhnout pouze při použití specializovaného hardware.

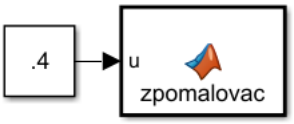
Dalším problémem je blok *UDP Send*, který je v příkladech [16, Simulink Support Package for Raspberry Pi Hardware Examples] použit k přenosu videa z Raspberry Pi do modelu běžícího v PC. Tento blok podporuje pakety nanejvýš o velikosti 65507 elementů, a tedy umožňuje přenášet obraz o nejvyšším rozlišení 160 x 120 pixelů. Řešením by mohlo být použití funkce, která by pole s obrazovými daty rozdělila do několika menších polí, poslat každé zvlášť jedním *UDP Send* blokem a v PC je následně opět složit. Nativní rozlišení

[8] Anglicky *External* a *Deployed*.

[9] Například `rpi=raspberrypi; rpi.stopModel('bezici_model');`.

```
1 function zpomalovac(u)
2 pause(u)
```

Kód 3.7: Funkce `zpomalovac`



Obrázek 3.1: Ukázka použití MATLAB Funkce zpomalovač v Simulinku

modulu *Camera Module V2* je 1080p (1920 x 1080 pixelů), což představuje 2 073 600 prvků v jednom kanálu, a tedy pro přenos barevného obrazu o třech kanálech by bylo potřeba 95 *UDP Send* bloků.

3.5 Použití Raspberry Pi s jazykem Python

Python je programovací jazyk vytvořený s důrazem na jednoduchost a stručnost se zachováním použitelnosti a univerzality [26], což ho činí ideálním pro použití na zařízení jako Raspberry Pi. Základní možnosti Pythonu je možné rozšířit doinstalováním dalších knihoven, které umožní použití GPIO portů, kamery, sériovou komunikaci, síťovou komunikaci, nebo třeba zpracovaní dat nebo obrazu.

S použitím návodu [27] jsem sestavil jednoduchou ukázku 3.8. Tento program nastaví pin 26[10] jako input, přiřadí mu interrupt, a při detekování klesající hrany průběhu napětí provede funkci `buttonPressed`. Jednotlivé částí jsou:

- `import`: načtení knihovny, použitím funkce `from` se načtou pouze ty funkce, které plánuji použít,
- `GPIO.setmode(GPIO.BCM)`: nastavení číslování pinů dle číslování procesoru,
- `GPIO.cleanup()`: nastavení všech pinů do výchozího nastavení, které je vhodné provádět vždy na konci programu, protože zabrání neočekávaným stavům při příštím použití GPIO,
- `GPIO.add_event_detect()`: provedení funkce při detekci události na pinu, s eliminaci záchvěvu,
- `try; except; finally`: tato konstrukce ošetřuje neočekávané stavy, případně ukončení programu stiskem CTRL+C, a zajišťuje provedení `GPIO.cleanup()` před ukončením programu[11].

Bližší informace je možné dohledat v dokumentaci jazyka Python [29] a dokumentaci knihovny `RPi.GPIO` [30].

[10] V této práci číslování pinů vždy odpovídá číslování dle procesoru (*BCM*, podle výrobce procesoru *Broadcom*).

[11] Více informací ve článku [28].

```python
from time import sleep
import RPi.GPIO as GPIO

# nastaveni cislovani pinu
GPIO.setmode(GPIO.BCM)

# nastaveni pouziteho pinu
GPIO.setup(26, GPIO.IN, pull_up_down=GPIO.PUD_UP)

STOP = 0

# funkce volana pri interruptu
def buttonPressed(channel):
    # globalni promennou pro zapis nutno volat pres global
    global STOP
    STOP = 1

# detekce udalosti
GPIO.add_event_detect(26, GPIO.RISING, callback=buttonPressed, bouncetime=150)

# telo programu
try:
    while not STOP:
        sleep(1)

except Exception as e:
    print(e)
finally:
    GPIO.cleanup()
    print "All cleaned up."
```

Kód 3.8: Příklad programu v Pythonu. K pinu 26 je připojeno tlačítko, které tento pin při stisku spojí se zemi, což procesor okamžitě zaregistruje, přeruší prováděnou činnost a zavolá funkci `buttonPressed()`.

3.5.1 Ultrazvukový dálkoměr

Pro zabránění nárazu do překážky lze použít ultrazvukový dálkoměr, například modul *HC-SR04*. Ten při přivedení napětí na vstup *Trig* vyšle skupinu impulzů, následně při čekání na ozvěnu impulzů přivede na výstup *Echo* napětí – logická jednička. Změřením doby po kterou je na výstupu logická jednička a vydělením rychlostí zvuku lze získat vzdálenost, kterou pulz urazil [31]. Modul používá 5V úroveň, pro bezpečné připojení k Raspberry Pi je nutné ho připojit přes jednoduchý napěťový dělič tvořený dvěma stejnými rezistory, který tak sníží napětí na polovinu. Pro jeho ovládání jsem podle datového listu [31] s použitím knihovny `RPi.GPIO` vytvořil jednoduchý program `/raspberrypi/test/dalkomer_test.py`.

3.5.2 Použití knihovny `gpiozero`

Kromě knihovny `RPi.GPIO` existuje ještě knihovna `gpiozero`, která pro svou funkci implicitně používá knihovnu `RPi.GPIO`, ale snaží se použití GPIO pinů učinit přístupnější, a pro nejběžnější elektronické součástky nabízí objekty, které lze snadno ovládat prostřednictvím

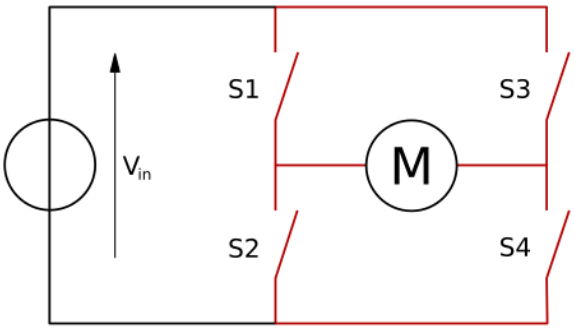
metod[12] [32]. Všechny informace k možnostem této knihovny jsou uvedeny v její dokumentaci [33]. V kódu 3.9 je příklad použití převzatý z dokumentace [32].

```
1  from gpiozero import LED
2  from time import sleep
3
4  # vytvoreni objektu LED diody na pinu 17
5  led = LED(17)
6
7  while True:
8      led.on()
9      sleep(1)
10     led.off()
11     sleep(1)
```

Kód 3.9: Příklad použití knihovny `gpiozero`. Nejprve je vytvořen objekt LED diody připojené k portu 17 a následně se v cyklu střídavě rozsvěcí a zhasíná s jednosekundovou pauzou.

3.5.3 Ovládání DC motoru s knihovnou `gpiozero`

Knihovna `gpiozero` nabízí objekt `Motor` pro ovládání motoru užitím H-můstku. Ten je pro ovládání motoru nutno použít, neboť připojení motoru přímo na piny Raspberry Pi by způsobilo tok velkého proudu a poškození procesoru. Princip H-můstku je zobrazen na obrázku 3.2. Při sepnutí spínačů S1 a S4 se motor otáčí v jednom směru a při sepnutí spínačů S2 a S3 v druhém. H-můstek taky často obsahuje cirkulační diodu, která zabraňuje vzniku napětí při odpojení motoru způsobenému cívkami motoru [34, Using motors].



Obrázek 3.2: Schéma H-můstku. Převzato z [35].

Kompletní kód programu je přiložen na médiu jako `/raspberrypi/test/DCmotor_test.py`. Program byl sestaven podle dokumentace k `gpiozero` [33] a dokumentace k jazyku Python [29]. Funkce jednotlivých částí je následující:

- V těle programu se nejprve inicializuje používaná proměnná a vytvoří se objekt motoru. Následně probíhá hlavní smyčka, ve které program čeká na vstup z klávesnice ve formátu reálného čísla. Po zadání vstupu se zavolá funkce `motor_ctrl()`.
- Funkce `motor_ctrl()` slouží k nastavení rychlosti motoru. To se provede tak, že podle znaménka rychlosti funkce zavolá metody objektu motoru pro přímý nebo zpětný chod.

[12] Více informací o objektech, třídách a metodách například v dokumentaci [29, Classes].

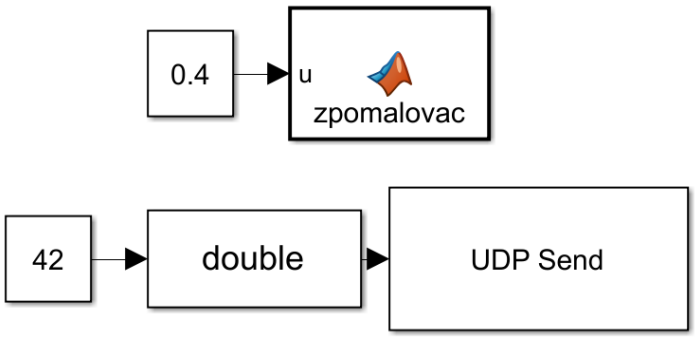
Pokud se nová rychlost od stávající výrazně liší, tedy pokud se liší znaménkem nebo o určitou hodnotu, zavolá se funkce `motor_rozjezd()`.

- Funkce `motor_rozjezd()` provede změnu rychlosti z předchozí rychlosti na novou. Pokud se liší znaménkem, nejprve nastaví nulovou rychlost a počká zadaný čas na zastavení motoru. Následně se vypočte potřebné zrychlení podle zadaného času na rozjezd a podle počtu kroků. Poté se v cyklu nastavuje vzrůstající rychlost podle vypočteného zrychlení a zadaného času na rozjezd a počtu kroků.

- Po proběhnutí volaných funkcí čeká hlavní cyklus na novou hodnotu rychlosti.

3.5.4 Komunikace Pythonu na Raspberry Pi se Simulinkem

Nejjednodušší možností komunikace je užitím protokolu UDP, který pro vysílání vyžaduje pouze cílovou adresu a port a pro příjímání pouze port. V Simulinku stačí vytvořit jednoduchý model s blokem *UDP Send* z knihovny *DSP System Toolbox*, zadat IP adresu Raspberry Pi a zvolit port v rozsahu 49152 až 65535[13]. Vstupem do toho bloku může být jakékoli číslo, v příkladu na obrázku 3.3 je použit blok *Data Type Conversion*, který zajistí, že vstup do bloku *UDP Send* bude vždy typu double, na který je připraven program běžící na Raspberry Pi. Model jsem dále doplnil o zpomalovač, který spolu s vhodným nastavením kroku simulace na pevnou hodnotu umožní běh modelu dostatečně blízký reálnému času.



Obrázek 3.3: Model *UDPtest*

Podle dokumentace Pythonu [29] jsem sestavil jednoduchý program, který poslouchá na odpovídajícím portu a při přijmutí zprávy ji dekóduje a vypíše. Program je uveden jako kód 3.10 a přiložen na médiu jako `/raspberrypi/test/dalkomer_test.py`.

Pro demonstraci komunikace z Pythonu do Simulinku jsem upravil program pro ovládání dálkoměru a vytvořil nový program, který změří dálkoměrem vzdálenost, poté užitím funkce `struct.pack()` číslo zabalí do paketu se strukturou odpovídající typu double a nakonec odešle zadaným portem na IP adresu počítače. Pro příjem v Simulinku lze použít blok *UDP Recieve* s nastaveným typem double. Program je přiložen jako `/raspberrypi/test/dalkomer_test_s_udp.py`.

3.5.5 Ovládání motoru na Raspberry Pi ze Simulinku

Následující kombinace simulinkového modelu a pythonového programu umožňuje ze Simulinku interaktivně ovládat motor.

[13] Tyto porty jsou dle specifikace [36] vyhrazeny pro dynamické přidělování a soukromé využití a nejsou pevně přiděleny žádné aplikaci.

Simulinkový model

Model odesílá příkazy pro nastavení rychlosti a zatáčení prostřednictvím protokolu UDP ve formátu reálného čísla v rozsahu $\langle -1, 1 \rangle$. V případě rychlosti odpovídá $+1$ nejvyšší rychlosti dopředu a -1 nejvyšší rychlost dozadu, v případě zatáčení odpovídá -1 největšímu zatočení doleva a $+1$ největšímu zatočení doprava. Při běhu modelu lze zatáčení a rychlost plynule ovládat pomocí posuvníků.

Pro rychlost je použit port 14550 a pro zatáčení port 14551. V blocích *UDP Send* je nutné nastavit aktuální adresu Raspberry Pi. Pro regulaci časového kroku modelu jsem použil funkci *zpomalovac*. Model je zobrazen na obrázku 3.4 a je přiložen na médiu jako `/PC/simulink/UDPrizeni.slx`.

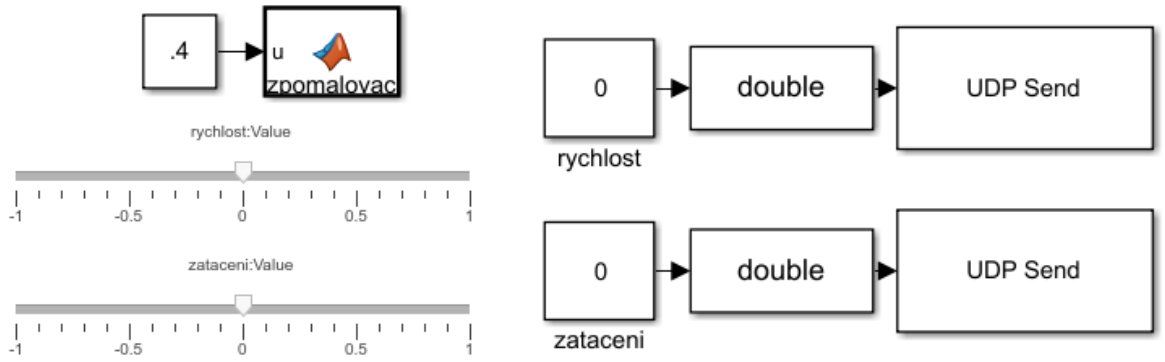
Kód pro Raspberry Pi

Program běžící na Raspberry Pi je složen z hlavního souboru `motor_rizeni.py` a souboru obsahujícího funkce `motor_rizeni_funkce.py`. Soubory jsou přiloženy na médiu ve složce `/raspberrypi/`.

Program přijme z UDP portem 14550 hodnotu rychlosti a příslušnou funkcí přivede odpovídající PWM signál na piny H-můstku, který přivede odpovídající napětí na motor. Hodnota natočení není využita, snadnou úpravou programu by však šla použít například pro druhý motor. Při připojení pinu GPIO 26 k zemi se program ukončí. To lze realizovat například tlačítkem. Motor je připojen k pinům GPIO 27 a GPIO 22.

```
1  import socket
2  from struct import unpack
3
4  # poslech na vsech interface
5  HOST = ''
6  # port pro prijem
7  PORT = 25000
8
9  # vytvoreni objektu pro UDP prijem
10 # AF_INET ... internet, SOCK_DGRAM ... UDP
11 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
12 # svazani socketu s adresou a portem
13 sock.bind((HOST, PORT))
14
15 try:
16     while True:
17         # prijem paketu a zaznamenani adresy odesilatele
18         data,adresa = sock.recvfrom(256)
19         # rozbaleni prijateho paketu typu double
20         data = unpack('d',data)
21         print(data[0])
22 except KeyboardInterrupt:
23     print("preruseno")
24 finally:
25     sock.close()
```

Kód 3.10: Program vypisující zprávu přijatou přes UDP



Obrázek 3.4: Model *UDPrizeni*

3.6 Řízení zatáčení Arduinem

Zatáčení podvozku realizované stejnosměrným motorem je astatická soustava – při přivedení určitého konstantního napětí na vývody motoru se bude úhel natočení kol zvyšovat, dokud nenarazí na mechanický doraz. Přidáním regulátoru s proporcionální složkou soustava získá statickou charakteristiku, a přidáním zpětné vazby vznikne díky integračnímu charakteru soustavy Uzavřený regulační obvod (URO) s nulovou trvalou regulační odchylkou $e(\infty) = w(\infty) - y(\infty)$, což je rozdíl mezi hodnotou žádané veličiny w a hodnotou regulované veličiny y po ustálení [37]. Aktuální hodnotu regulované veličiny získám čtením natočení potenciometru.

K realizace regulátoru na Arduinu jsem použil knihovnu *Arduino PID Library*[14] a implementoval podle návodu v dokumentaci knihovny [38]. Pro realizaci příjmu žádané hodnoty přes sériovou linku jsem použil návod [39]. Výsledný program je na přiloženém médiu jako *arduino/ovladani_zataceni/ovladani_zataceni.ino*.

Program je navržen tak, že nadřazený program mu neustále posílá žádanou hodnotu, ale regulátor přijme novou žádanou hodnotu až po dokončení regulace, kdy velikost regulační odchylky leží v tolerančním pásmu. Nastavení požadovaného zatočení probíhá velmi rychle, takže čekání na konec regulace před přijetím nové hodnoty neznemožňuje plynulé zatáčení. Komentáře k jednotlivým částem jsou přímo v programu. Princip fungování je následující:

- Na začátku programu jsou definovány používané konstanty, jako je hodnota natočení pro přímou polohu a pro krajní polohy, toleranční pásmo pro regulační odchylku a minimální dobu regulace.

- Následně se inicializují používané proměnné. Úvodní žádaná hodnota je nastavena na nulu, a poté začne hlavní smyčka.

- Cyklicky se z žádané a měřené skutečné hodnoty zatočení vypočítává aktuální hodnota akční veličiny u, která je funkcí `zapis_u()` přivedena na H-můstek, a ten na motor přivede napětí s příslušnou polaritou.

- Po dosažení velikosti e ležící v tolerančním pásmu se smyčka zastaví a program přijme ze sériové linky novou žádanou hodnotu. Poté smyčka opět pokračuje s novou žádanou hodnotou.

[14] Knihovnu je nutné v Arduino IDE doinstalovat přes nabídku *Projekt →Přidat knihovnu →Spravovat knihovny...* a vyhledat knihovnu *PID by Brett Beauregard*.

Žádaná hodnota je z intervalu $\langle -1, 1 \rangle$ a program ji přijímá ve formě reálného čísla se znaménkem o celkové délce šesti znaků, například $+0.420$ nebo -0.233. Pokud reálný rozsah natočení je nižší, je třeba v programu změnit odpovídající konstanty.

Pro testovací účely jsem použil regulátor pouze typu P. V následující, čtvrté kapitole práce vyzkouším, jestli lze přidáním složek I a D zlepšit vlastnosti regulačního pochodu.

3.6.1 Připojení Arduina k Raspberry Pi

Komunikaci Arduina s Raspberry Pi je možné realizovat prostřednictvím sériového rozhraní, kdy je Arduino připojeno do USB konektoru Raspberry Pi. K obsluze sériové komunikace lze použít Python. Nejprve je nutné doinstalovat modul pro komunikaci přes sériový port. To se provede v terminálu příkazem `pip install pyserial`. Pro otestování komunikace po sériové lince přes USB jsem vytvořil testovací programy. Podle návodu [40] jsem vytvořil pythonový program pro Raspberry Pi 3.12, na médiu jako `/raspberrypi/test/usb_serial_test.py`, který komunikuje přes sériovou linku. Ta mívá adresu `/dev/ttyUSB0`, číslo se však může lišit. Aktuální číslo lze po připojení Arduina do USB zjistit v terminálu příkazem `ls /dev/ttyUSB*`.

Pro Arduino jsem vytvořil program 3.11, na médiu jako `/arduino/serial_test/serial_test.ino`, podle příkladu v dokumentaci Arduina [41, read()], který čeká na zprávu na sériové lince, a když nějakou přijme, přečte první číslo[15] zprávy (první byte) a pošle toto číslo zpět.

Obecný návod k programování Arduina je například na oficiálních stránkách [41, Getting Started with Arduino and Genuino products].

```
1  int prichozi = 0;
2
3  void setup() {
4          // otevreni seriove linky
5          Serial.begin(9600);
6  }
7
8  void loop() {
9          // posle zpravu po prijmuti zpravy
10         if (Serial.available() > 0) {
11                 // prijmuti prichoziho bytu
12                 prichozi = Serial.read();
13                 // odeslani prijmuteho bytu
14                 Serial.println(prichozi);
15         }
16 }
```

Kód 3.11: Program přijme zprávu ze sériové linky a její první písmeno pošle zpět.

[15] Zprávy při komunikaci přes sériovou linku jsou často typu char a pak tedy na zprávu obsahující číslo 0 přijde odpověď 48, což v tabulce znaků ASCII odpovídá číslu 0.

```
import serial
from os import listdir

# pripojeni k seriove lince
if 'ttyUSB0' in listdir('/dev/'):
    cesta = '/dev/ttyUSB0'
else:
    cesta='/dev/ttyUSB1'
# vytvoreni objektu na seriove lince
arduinoSerialData = serial.Serial(cesta,timeout=1,baudrate=9600)

def posli_a_cekej(zprava,serial):
    '''Posle zpravu, nasledne vycisti zasobnik a ceka na odpoved. '''
    # promenna pro odpoved
    vystup = ''
    predchozi = 'xxx'
    # vycisteni zasobniku
    serial.flushInput()
    serial.flushOutput()
    # poslani zpravy
    serial.write(zprava)
    while True:
        vystup+=serial.read()
        #pokud se vystup uz nezmeni, tak ho funkce vrati
        if vystup==predchozi:
            return vystup
        predchozi=vystup

# telo programu
while True:
    print posli_a_cekej(raw_input('zprava na poslani: '),arduinoSerialData)
```

Kód 3.12: Ukázka sériové komunikace s Arduinem připojeným přes USB. Program se připojení k Arduinu jako sériovému zařízení, odešle zprávu a poté čeká na odpověď.

3.7 Finální programy pro řízení

Spojením programů ovládajících jednotlivé části vznikne řešení schopné řídit podvozek. Skládá ze simulinkového modelu běžícího na PC, pythonovému programu běžícího na Raspberry Pi a programu běžícího na Arduinu. Z PC jsou odesílány příkazy do Raspberry Pi. Raspberry Pi příkaz přijme a pokud nedetekuje překážku, pošle do Arduina, které zprostředkovává ovládání motorů. V následující části uvedu jednotlivé programy a popíšu v sekcích podle platforem. Zapojení je zobrazeno na obrázku 3.6.

3.7.1 PC

Simulinkový model běžící na PC odesílá příkazy pro nastavení rychlosti a zatáčení protokolem UDP. Hodnoty jsou v rozsahu $\langle -1, 1 \rangle$, což odpovídá *nejvíce doleva* až *nejvíce doprava* pro zatáčení, respektive *nejrychleji dozadu* až *nejrychleji dopředu* pro rychlost. Pro rychlost je použit port 14550 a pro zatáčení port 14551. Dále je použit port 25000 pro příjem hodnoty naměřené vzdálenosti.

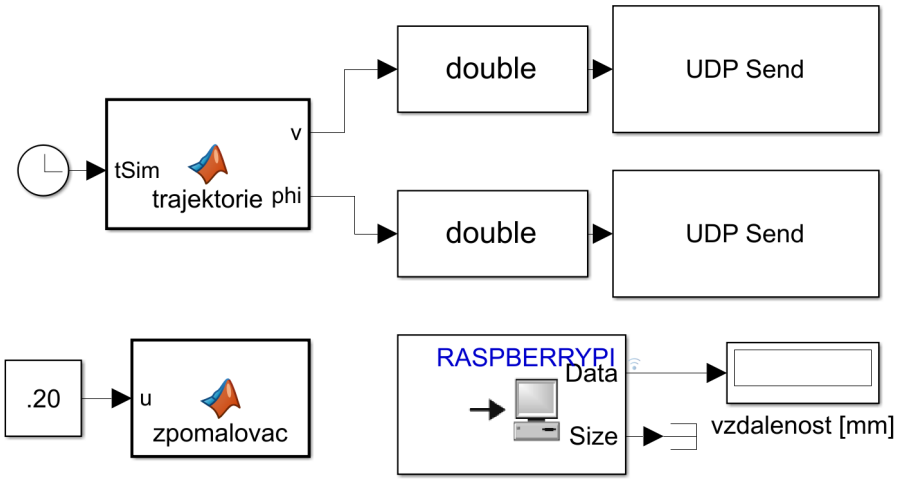
Aktuální hodnoty zatočení φ_i a rychlosti v_i jsou pomocí funkce *trajektorie* vybírány z tabulky na základě aktuálního simulačního času t_{sim}, kde hodnoty t_i udávají dobu setrvání na hodnotě zatočení a rychlosti. Hodnota indexu i je nejnižší taková hodnota, že v daném okamžiku t_{sim} platí

$$\sum_{j=1}^{i} t_j > t_{\text{sim}}.$$

Tabulka má formát

$$\begin{bmatrix} \varphi_1 & v_1 & t_1 \\ \varphi_2 & v_2 & t_2 \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

Model je zobrazen na obrázku 3.5.



Obrázek 3.5: Model *UDPtrajektorie*

3.7.2 Raspberry Pi

Program běžící na Raspberry Pi je pro přehlednost složen ze dvou částí – hlavní část je tělo programu *raspberry_rizeni.py*, ve kterém jsou ze souboru *raspberry_rizeni_funkce.py* importovány používané funkce. Oba soubory jsou na médiu ve složce `/raspberrypi/`. Pro funkčnost je nutné nastavit IP adresu PC, na kterém běží řídící Simulinkový model.

Program nejprve provede nutné nastavení UDP komunikace, dálkoměru, relé, tlačítka pro ukončení a sériové linky Arduina. Poté jsou do Arduina poslány nulové hodnoty pro rychlost a natočení a je sepnuto relé.

Následně probíhá cyklus, ve kterém jsou volány jednotlivé funkce. Raspberry Pi nejprve přijme přes UDP hodnoty rychlosti a zatočení. Následně pomocí dálkoměru změří, jestli před podvozkem není překážka. Pokud detekuje překážku, je hodnota rychlosti nastavena na nulu. Poté sériovou linkou pošle do Arduina hodnoty natočení a rychlosti. Naměřenou hodnotu vzdálenosti pošle přes UDP do PC.

Pokud Raspberry Pi přes UDP po určitou dobu nedostane novou hodnotu natočení nebo rychlosti nebo při běhu dojde k jiné chybě, je tato hodnota nastavena na nulu. Pokud dojde k chybě při posílání do Arduina, je relé rozepnuto. Pokud dojde k nějaké jiné chybě, je relé rovněž rozepnuto a program se ukončí.

Pokud Raspberry Pi detekuje stisk tlačítka *STOP*, relé je rozepnuto a podvozek se zastaví.

Arduino je k Raspberry Pi připojeno pomocí USB kabelu.

Nastavitelné parametry jsou:

- timeout_cas = 0.5: doba čekání na přijmutí nové hodnoty přes UDP v sekundách,
- vzdalenost_pro_zastaveni = 500: vzdálenost překážky, pod kterou Raspberry Pi nastaví rychlost na nulu,
- pin_trigger = 23, pin_echo = 24, pin_rele = 2, pin_tlacitka = 21: čísla pinů pro připojení jednotlivých elektronických součástek.

3.7.3 Arduino

Na Arduinu běží program *arduino_ovladani.ino*, na médiu přiložen ve složce `/arduino/arduino_ovladani/`, starající se o řízení motorů. Program nejprve inicializuje používané proměnné, PID regulátor a používané piny. Žádaná hodnota zatočení je na začátku nastavena na nulu. Jednotlivé prováděné činnosti jsou realizovány prostřednictvím funkcí, které jsou volány při běhu hlavní smyčky.

V hlavní smyčce je nejprve načtena hodnota aktuálního zatočení kol a následně vypočtena hodnota akční veličiny, která je funkcí přenesena na odpovídající napětí na motoru zatáčení. Jakmile je regulační odchylka žádané a reálné hodnoty natočení dostatečně malá, je zavolána funkce, která přijme novou žádanou hodnotu natočení a hodnotu rychlosti. Následně je zavolána funkce pro nastavení rychlosti `zapis_rychlost()`. Poté se cyklus opakuje.

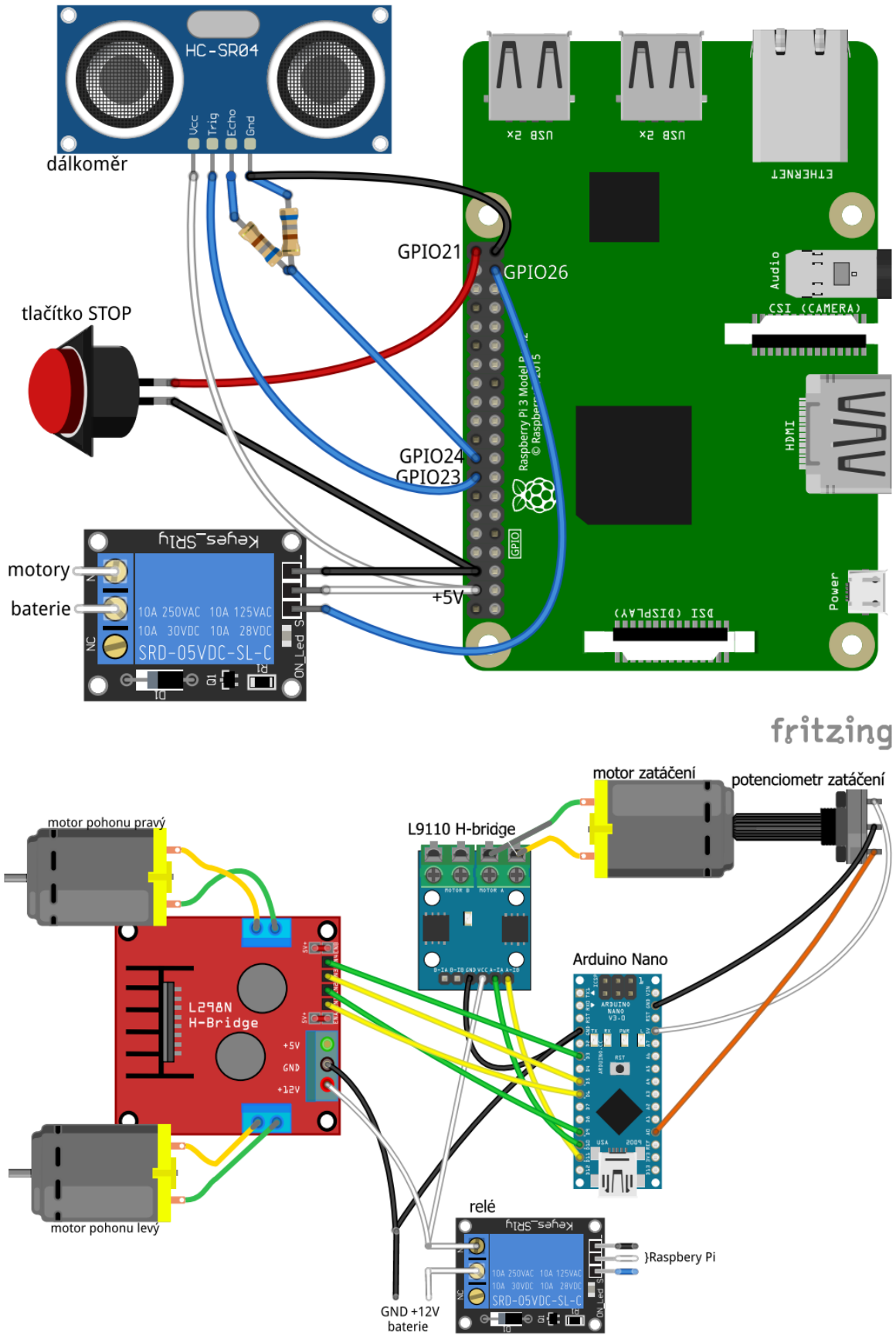
Rozjíždění a zastavování probíhá podle nastavené sekvence a obstarávají je funkce `zastaveni()` a `rozjezd()`, které jsou volány funkcí `zapis_rychlost()`.

Piny pro připojení jsou:

- pinZataceniDoleva = 10: pin motoru zatáčení pro natočení kol doleva,
- pinZataceniDoprava = 11,
- pinLevyDopredu = 6: pin levého motoru pohonu pro jízdu dopředu,
- pinLevyDozadu = 9,
- pinPravyDopredu = 5,
- pinPravyDozadu = 3.

Nastavitelné parametry modelu jsou:

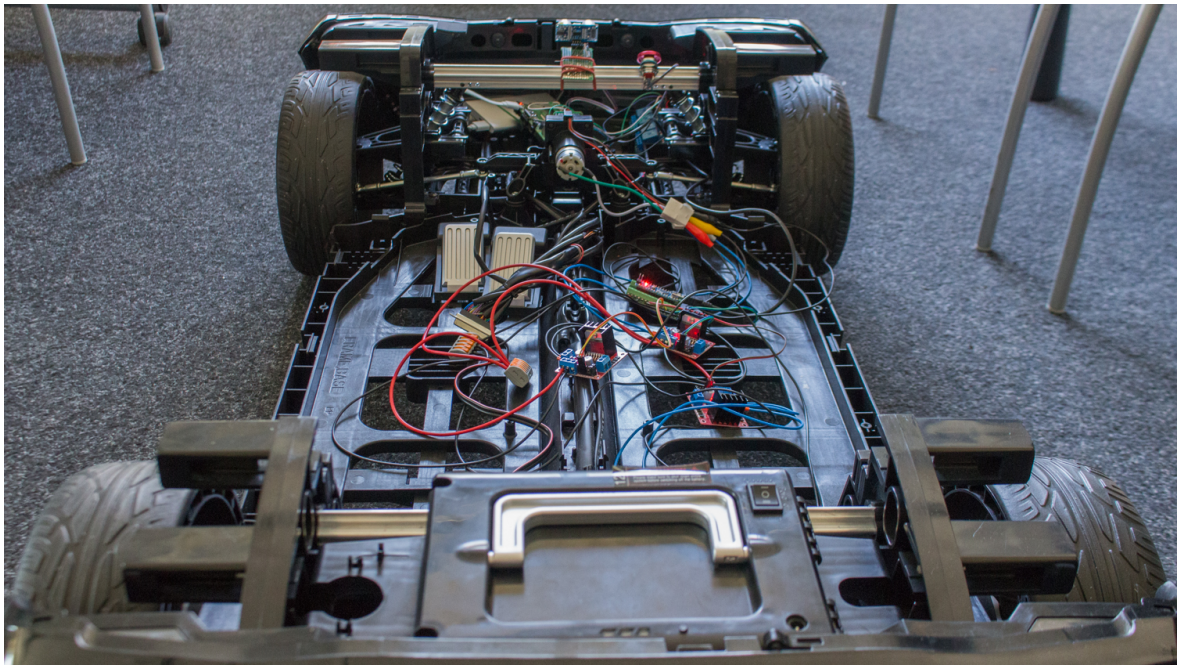
- nuloveNatoceni: hodnota natočení pro přímý směr,
- pravy_doraz: hodnota natočení pro pravý mechanický doraz,
- levy_doraz,
- mrtve_pasmo: relativní hodnota napětí na motoru zatáčení, když už je motor schopen překonat mechanické odpory,
- tolerancni_pasmo: maximální relativní odchylka žádané hodnoty od skutečné, při které je regulace ukončena,
- doba_regulace: minimální doba v milisekundách, po kterou probíhá regulace,
- r0, rI, rD: parametry PID regulátoru,
- pocet_vzorku: počet vzorků měření aktuální hodnoty regulované veličiny, jejichž průměr je použit jako aktuální hodnota,
- pocetRychlosti: počet prvků sekvence pro rozjezd a zastavení,
- rozjezdRychlost[], rozjezdCas[]: mezihodnoty rychlosti při rozjíždění a příslušné doby setrvání.



Obrázek 3.6: Schéma zapojení. Vytvořeno v programu *Fritzing*.

Kapitola 4

Experimentální ověření funkčnosti



Obrázek 4.1: Fotografie výsledného řešení

4.1 Zatáčení

Nejprve je nutné změřit rozsah zatáčení. K tomu účelu jsem vytvořil jednoduchý program pro Arduino, který měří natočení potenciometru a naměřenou hodnotou vypisuje do sériové linky. Pro zlepšení přesnosti měření je hodnota průměrována ze sta měření. Program je přiložen na médiu jako `/arduino/mereni_natoceni/mereni_natoceni.ino`. Naměřený rozsah řízení je

$$\langle -0.600, +0.580 \rangle$$

s nulovou polohou +0.010.

Pro lepší funkci jsem zvýšil proporcionální složku na $r_0 = 3$, což snížilo mrtvé pásmo, kdy na motoru není dostatečně vysoké napětí pro překonání mechanického odporu. Zjištěné vhodné nastavení mrtvé pásma je

$$\text{mrtve\_pasmo} = 0.040.$$

Při použití pouze P složky docházelo k tomu, že zatáčení přejelo nulovou polohu. Po přidání složky I hodnotou $r_I = 1\,1/\mathrm{s}$ dochází k překmitu a lepšímu dosažení požadované hodnoty. Zjištěná vhodná hodnota pro šířku tolerančního pásma, při jehož dosažení je regulace ukončena, je

$$\mathrm{toleranci\_pasmo} = 0.01$$

a pro hodnotu minimální doby regulace je

$$\mathrm{doba\_regulace} = 100 \text{ ms}.$$

4.2 Pohon

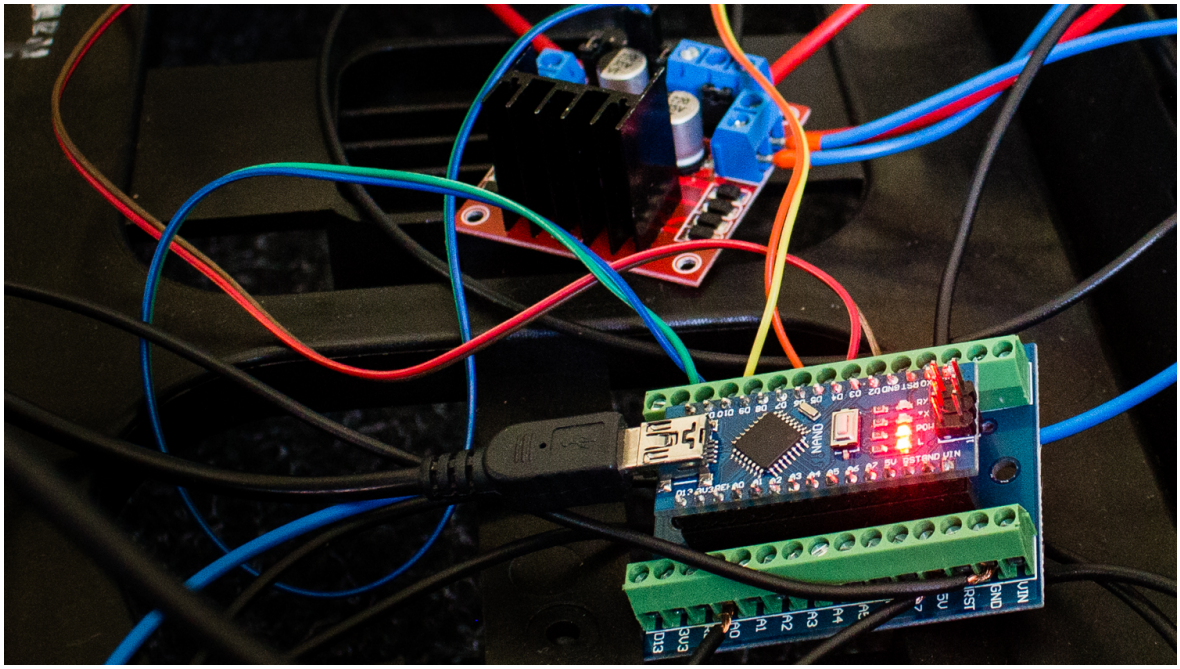
Pro ověření funkčnosti pohonu jsem použil model *UDPrizeni* a postupně jsem zvyšoval hodnotu rychlosti, dokud se motory neroztočily. Pro překonání mechanického odporu a roztočení je potřeba PWM signál 60 %, tedy napětí 7.2 V. Jako vhodná sekvence pro rozjezd se jeví nejprve nastavit rychlost na 100 % na dobu asi jedné sekundy a následně snížit na 60 %. S touto sekvencí se podvozek rozjede a jede nízkou rychlostí. Motory jsou připojeny k baterii přes H-můstky s čipem L298N, který má nominální proud 2 A. Tato hodnota je pro tyto motory nedostačující a při provozu se můstky rychle zahřívají. Pro budoucí použití bude nutné je nahradit výkonnějším hardwarem.

4.3 Celková funkčnost

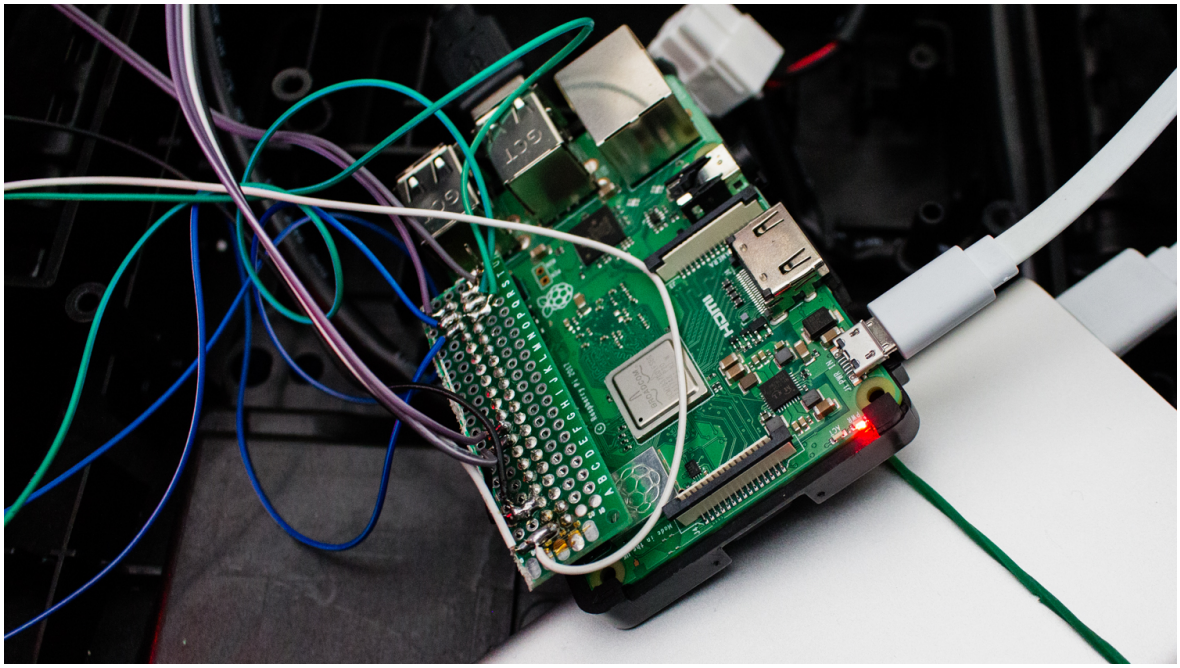
Pro ověření celkové funkčnosti jsem použil model *UDPtrajektorie*, který postupně posílá příkazy podle zadané trajektorie. S tímto modelem se mi povedlo realizovat jízdu s křivočarou trajektorií a ověřit funkčnost zastavení při detekci překážky. Vhodné nastavení vzdálenosti pro zastavení je

$$\mathrm{vzdalenost\_pro\_zastaveni} = 700.$$

Na obrázcích 4.2 a 4.3 je vyfoceno Raspberry Pi a Arduino. Další fotografie podvozku s řídícím hardwarem jsou v příloze C.



Obrázek 4.2: Arduino a H-můstek



Obrázek 4.3: Raspberry Pi napájené z powerbanky

Závěr

V této práci jsem prozkoumal hardware vhodný pro ovládání elektrického vozítka a použitelné softwarové vybavení. Ze zjištěných možností jsem vybral jednu kombinaci a následně blíže rozvedl dostupné softwarové možnosti.

Pro vybranou variantu jsem vytvořil řešení, používající jako základní prvek Raspberry Pi a programovací jazyk Python. Raspberry Pi přijímá příkazy z PC, kde jsou generovány z prostředí Simulink, buď interaktivně použitím posuvníků nebo jako předdefinovaná sekvence. K Raspberry Pi je připojeno Arduino, které přijímá příkazy z Raspberry Pi a zajišťuje ovládání motorů pohonu a zatáčení. Motory jsou k baterii připojené přes relé, které ovládá Raspberry Pi. Relé jsou sepnuta při spuštění programu a rozepnuta při ukončení, které může být způsobeno například výskytem nějaké chyby.

Raspberri Pi dále disponuje dálkoměrem, a pokud před podvozkem detekuje překážku, zastaví. Dále detekuje stisk tlačítka *STOP*, což způsobí rozepnutí relé a ukončení programu.

Pro zlepšení stávajícího řešení by bylo možné přidat ke kolům odměřování, umožňující implementovat řízení pohonu jako regulační proces, což by umožnilo pohyb zadanou rychlostí nebo ujetí zadané vzdálenosti, a tedy přesné sledování trajektorie. K tomu by bylo vhodné zavést interaktivní komunikační protokol mezi Raspberry Pi a Arduinem, který by umožnil Raspberry Pi zadat příkazy ve smyslu „*natoč kola na úhel φ a po dobu t jeď rychlostí v*“ nebo „*ujeď oblouk o zakřivení ρ a délce x*“, a Arduinu umožnil ohlásit zpět splnění posledního příkazu. Dále by bylo vhodné místo UDP použít pro komunikaci mezi Raspberry Pi a PC sofistikovanější protokol, který by umožnil lepší vzájemnou komunikaci.

Řídící model pro Raspberry Pi by bylo možné vytvořit v Simulinku, například s použitím toolboxu *Stateflow* pro definování logiky a toolboxu *Simulink Coder* pro vytvoření spustitelných souborů běžících na Raspberry Pi. Nevýhodou tohoto řešení je menší rozsah balíku podpory pro Simulink, a tak pro použití některého hardware, jako například ultrazvukového dálkoměru, by bylo nutné vytvořit obsluhující ovladač. To je možné udělat použitím *S-Function* a vyžaduje složité programování v jazyce C.

Jako další směr rozvoje této experimentální podvozkové platformy navrhuji připojit k Raspberry Pi alespoň dvě kamery a použit svobodnou knihovnu OpenCV [42] pro zpracování stereoskopického obrazu, a umožnit tak lepší rozpoznávání překážek a orientaci v prostoru. Pro rozpoznávání překážek v 360stupňovém rozsahu kolem podvozku by bylo možné také použít LiDAR, jako například v práci [43].

Zdroje

[1] Amazon.com: Henes Broon F830 with Tablet PC 12V Kids Ride On Car Electric Powered Wheels Remote Control RC Black. [online], [cit. 2018-06-14]. Dostupné z: https://www.amazon.com/Tablet-Electric-Powered-Wheels-Control/dp/B00P51BVNE?tag=t9r-20

[2] Henes Broon F830 - bílé. [online], [cit. 2018-06-25]. Dostupné z: https://www.sparkys.cz/auto-henes-broon-f830-bile

[3] *Robotics*. Wikibooks, 2018. Dostupné z: https://en.wikibooks.org/wiki/Robotics

[4] What are the best single-board computers? [online], [cit. 2018-06-14]. Dostupné z: https://www.slant.co/topics/1629/~single-board-computers

[5] Parent-Charette, S.: The Best Microcontrollers 2017. [online], 2017, [cit. 2018-06-14]. Dostupné z: https://www.robotshop.com/blog/en/best-microcontrollers-2017-21178

[6] Arduino Uno Rev3. [online], [cit. 2018-06-14]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[7] *ATmega328/P*. 2016. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[8] Bruce, J.: Meet the Arduino Killer: ESP8266. [online], 2018, [cit. 2018-06-22]. Dostupné z: https://www.makeuseof.com/tag/meet-arduino-killer-esp8266/

[9] NodeMCU a jeho verzie. [online], 2017, [cit. 2018-06-22]. Dostupné z: https://www.root.cz/clanky/nodemcu-a-jeho-verzie-doska-s-wi-fi-cipom-esp8266/

[10] Dudka, M.: STM32. [online], 2017, [cit. 2018-06-22]. Dostupné z: https://www.root.cz/clanky/stm32-mikrokontroler-vstricny-k-amaterum/

[11] Holoubek, T.: Raspberry Pi: Programování v prostředí MATLAB/SIMULINK. 2017.

[12] *Raspberry Pi 3 Model B+*. Dostupné z: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[13] Raspberry Pi. [online], [cit. 2018-06-21]. Dostupné z: https://www.raspberrypi.org/documentation/

[14] Raspberry Pi Documentation. [online], [cit. 2018-06-15]. Dostupné z: https://www.raspberrypi.org/documentation/

[15] MTaylor; Jimb0: Raspberry gPIo. [online], [cit. 2018-06-15]. Dostupné z: https://learn.sparkfun.com/tutorials/raspberry-gpio

[16] MATLAB Documentation. [online], 2018, [cit. 2018-06-15]. Dostupné z: https://www.mathworks.com/help/

[17] Adams, J.: Introducing Raspberry Pi HATs. [online], 2014, [cit. 2018-06-17]. Dostupné z: https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/

[18] Jimb0: Arduino Shields. [online], [cit. 2018-06-17]. Dostupné z: https://learn.sparkfun.com/tutorials/arduino-shields

[19] Castle, A.: Everything You Need to Know about The Beaglebone Black. [online], [cit. 2018-06-21]. Dostupné z: http://www.tested.com/art/makers/459278-everything-you-need-know-about-beaglebone-black/

[20] Beagle Board. [online], [cit. 2018-06-21]. Dostupné z: https://beagleboard.org/

[21] Visual Programming Guide. [online], [cit. 2018-06-26]. Dostupné z: https://www.postscapes.com/iot-visual-programming-tools/

[22] Hawkins, M.: How To Format Pi SD Cards Using SD Formatter. [online], 2018, [cit. 2018-06-21]. Dostupné z: https://www.raspberrypi-spy.co.uk/2015/03/how-to-format-pi-sd-cards-using-sd-formatter/

[23] Long, S.: A security update for Raspbian PIXEL. [online], 2016, [cit. 2018-06-18]. Dostupné z: https://www.raspberrypi.org/blog/a-security-update-for-raspbian-pixel/

[24] Team, M. M. H.: MATLAB Support Package for Raspberry Pi Hardware. [online], 2018, [cit. 2018-06-19]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/45145-matlab-support-package-for-raspberry-pi-hardware

[25] Team, M. S.: Simulink Support Package for Raspberry Pi Hardware. [online], 2018, [cit. 2018-06-19]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/40313-simulink-support-package-for-raspberry-pi-hardware

[26] Peters, T.: PEP 20 – The Zen of Python. [online], [cit. 2018-07-17]. Dostupné z: https://www.python.org/dev/peps/pep-0020/

[27] Using an LCD Display with Inputs & Interrupts on Raspberry Pi. [online], 2014, [cit. 2018-06-19]. Dostupné z: https://www.rototron.info/using-an-lcd-display-with-inputs-interrupts-on-raspberry-pi/

[28] Eames, A.: RPi.GPIO basics 3 – How to Exit GPIO programs cleanly, avoid warnings and protect your Pi. [online], [cit. 2018-06-19]. Dostupné z: http://raspi.tv/2013/rpi-gpio-basics-3-how-to-exit-gpio-programs-cleanly-avoid-warnings-and-protect-your-pi

[29] Python 3.5.5 documentation. [online], 2018, [cit. 2018-06-19]. Dostupné z: https://docs.python.org/3.5/index.html

[30] Raspberry-gpio-python. [online], 2013, [cit. 2018-06-19]. Dostupné z: https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/

[31] HC-SR04 User Guide. [online], [cit. 2018-07-03]. Dostupné z: https://www.mpja.com/download/hc-sr04_ultrasonic_module_user_guidejohn.pdf

[32] Nuttall, B.: GPIO Zero: a friendly Python API for physical computing. [online], 2018, [cit. 2018-06-20]. Dostupné z: https://www.raspberrypi.org/blog/gpio-zero-a-friendly-python-api-for-physical-computing/

[33] Nuttall, B.: Gpiozero. [online], [cit. 2018-06-20]. Dostupné z: https://gpiozero.readthedocs.io/en/stable/index.html

[34] Physical Computing with Python. [online], [cit. 2018-06-20]. Dostupné z: https://projects.raspberrypi.org/en/projects/physical-computing/

[35] H bridge. [online], 2001-, [cit. 2018-06-20]. Dostupné z: https://en.wikipedia.org/wiki/H_bridge

[36] *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*. August 2011.

[37] Hofreiter, M.: *Základy automatického řízení*. V Praze: České vysoké učení technické, druhé vydání, 2018, ISBN 978-80-01-06380-4.

[38] Beauregard, B.: PIDLibrary. [online], 2018, [cit. 2018-06-26]. Dostupné z: http://playground.arduino.cc/Code/PIDLibrary

[39] Robin2: Serial Input Basics. [online], 2018, [cit. 2018-06-26]. Dostupné z: http://forum.arduino.cc/index.php?topic=288234.0

[40] Nelli, F.: Controlling Arduino by Raspberry Pi. [online], 2017, [cit. 2018-06-25]. Dostupné z: https://www.meccanismocomplesso.org/en/controlling-arduino-raspberry-pi/

[41] Arduino. [online], 2018, [cit. 2018-06-25]. Dostupné z: https://www.arduino.cc

[42] OpenCV library. [online], 2018, [cit. 2018-07-23]. Dostupné z: https://opencv.org/

[43] Řehořka, J.: Graphic Processing of LiDAR Data. 2017. Dostupné z: https://is.muni.cz/th/q0y8y/

Licence

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

Příloha A

Seznam použitých zkratek

ČVUT České vysoké učení technické v Praze.

ADC Analog-to-digital converter.

BBB Beaglebone Black.

DAC Digital-to-analog converter.

DVD Digital Versatile Disc.

EEPROM Electrically Erasable Programmable Read-Only Memory.

FPGA Field-Programmable Gate Array.

GND Ground.

GNU GNU's Not Unix!.

GPIO General-purpose Input/Output.

HDMI High-Definition Multimedia Interface.

I/O Input/Output.

I^{2}C Inter-Integrated Circuit.

LAN Local-Area Network.

LCD Liquid-Crystal-Display.

LED Light-Emitting Diode.

PC osobní počítač.

PLC Programovatelný logický automat.

PRU Programmable Real-Time Unit.

PWM Pulse-width modulation.

RAM Random-Access Memory.

RPi Raspberry Pi.

SD Secure Digital.

SoC System on a Chip.

SPI Serial Peripheral Interface.

SSH Secure Shell.

URO Uzavřený regulační obvod.

USB Universal Serial Bus.

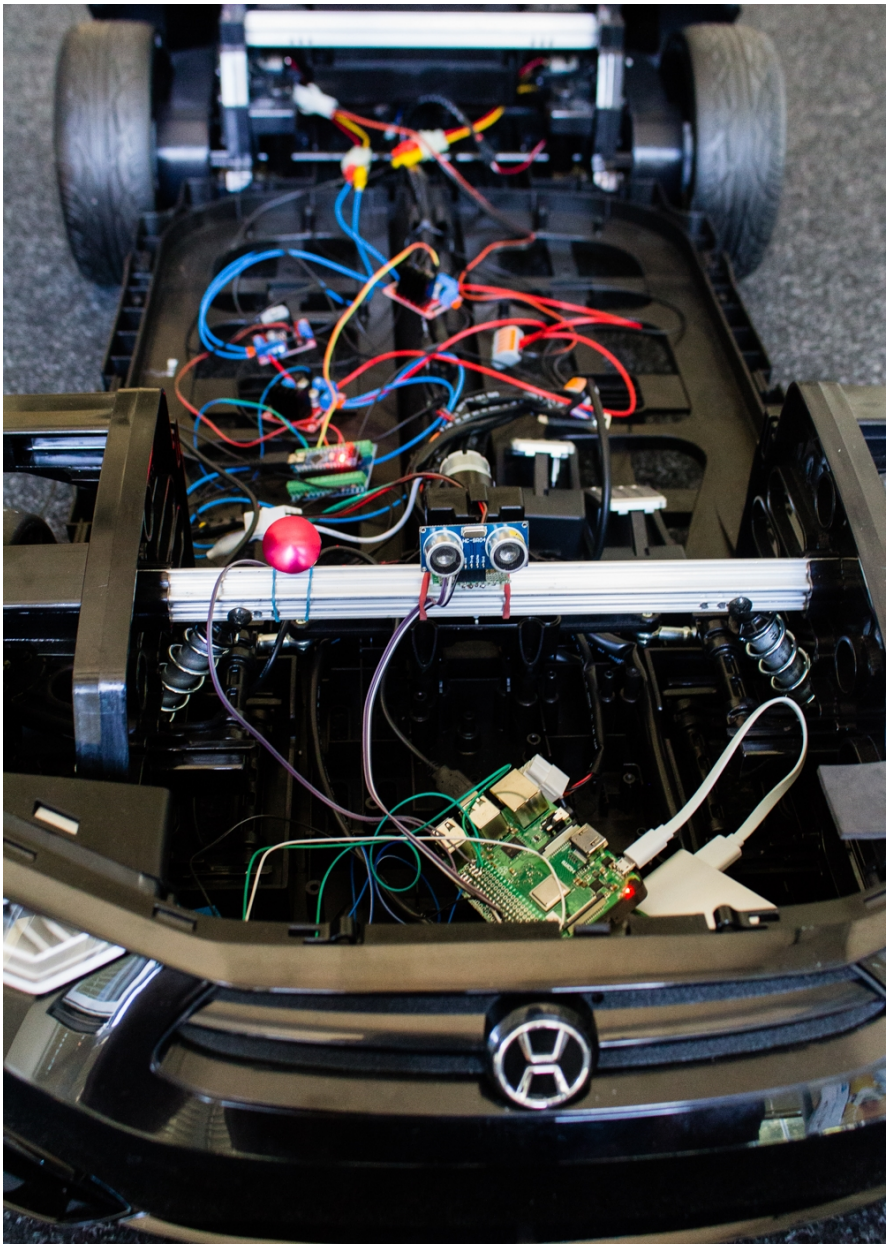
Příloha **B**

Obsah přiloženého media

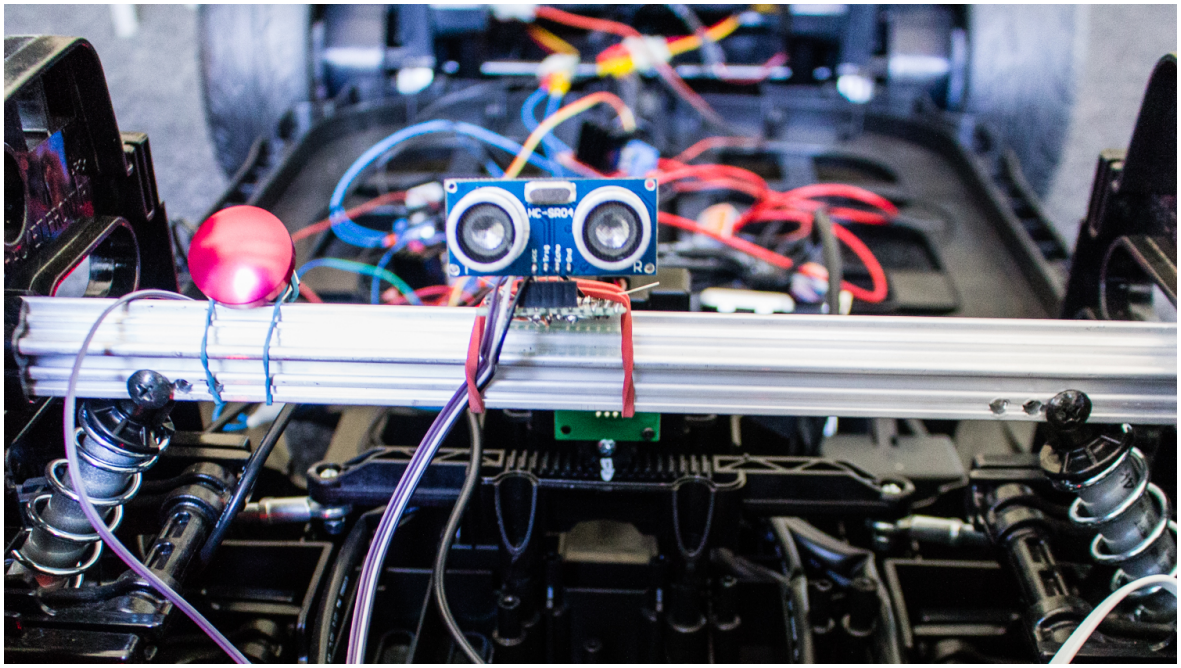
- `readme.txt` ..... stručný popis obsahu CD
- `arduino` ..... programy běžící na Arduinu
  - `arduino_ovladani`
  - `mereni_natoceni`
  - `ovladani_zataceni`
  - `serial_test`
- `raspberrypi` ..... programy běžící na Raspberry Pi
  - `boot`
  - `test`
  - `motor_rizeni.py`
  - `motor_rizeni_funkce.py`
  - `raspberry_rizeni.py`
  - `raspberry_rizeni_funkce.py`
- `PC` ..... programy běžící na PC
  - `simulink`

Příloha C

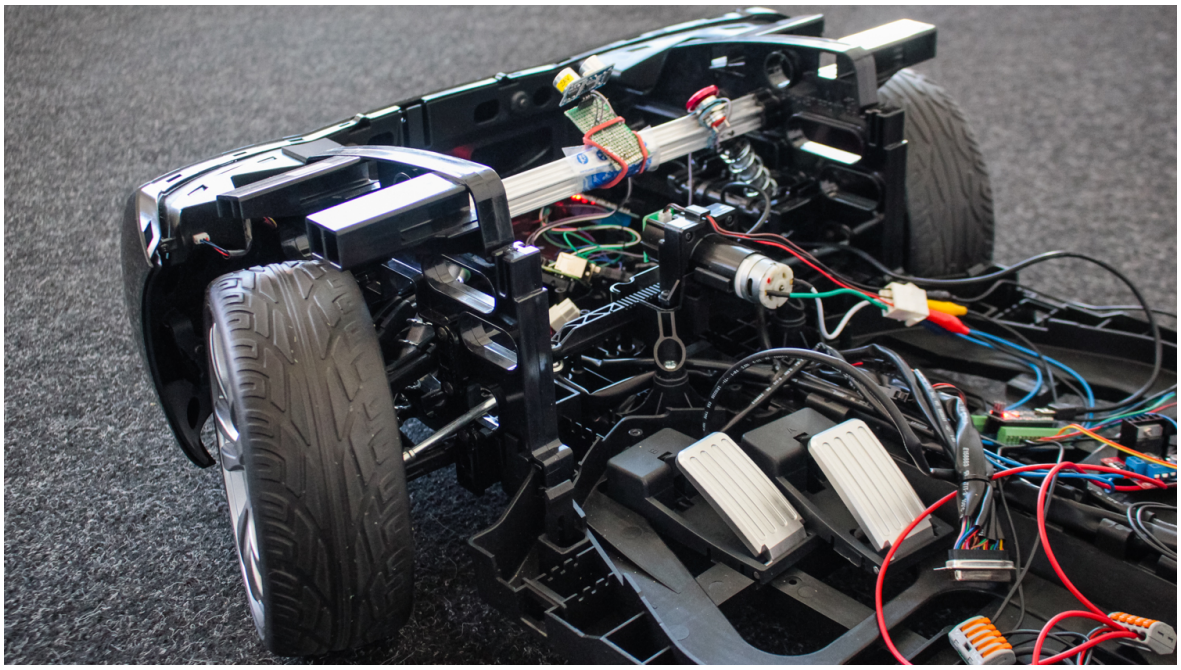
Fotografie výsledného hardware



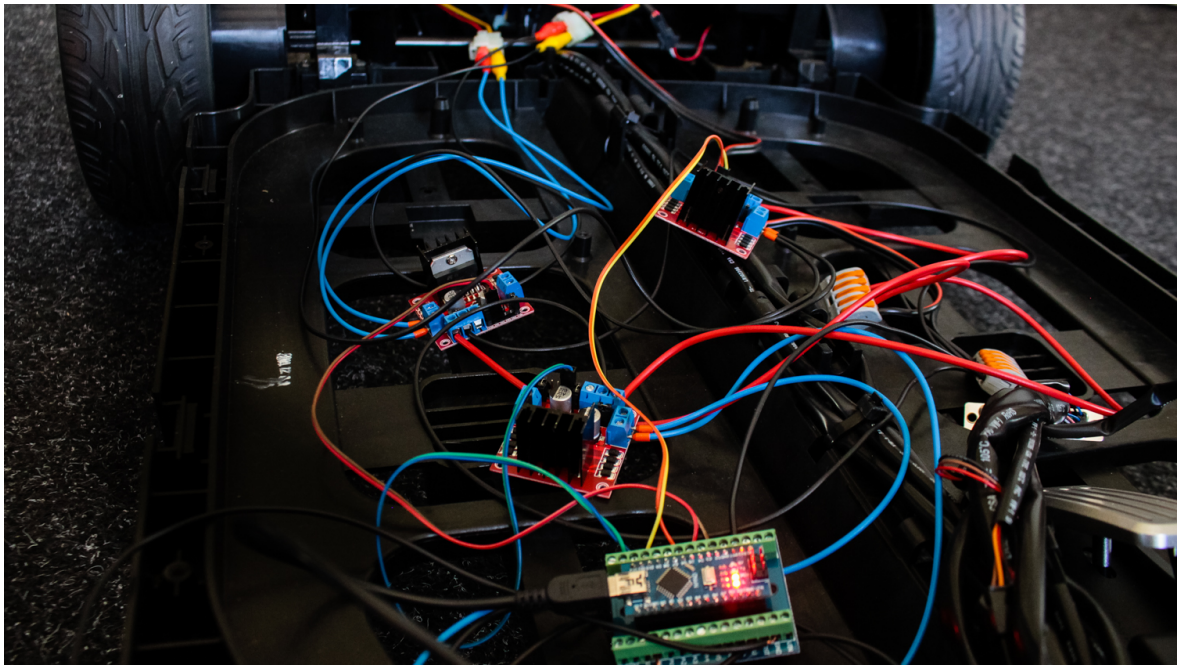
Obrázek C.1: Pohled zepředu. V popředí lze vidět dálkoměr a Raspberry Pi.



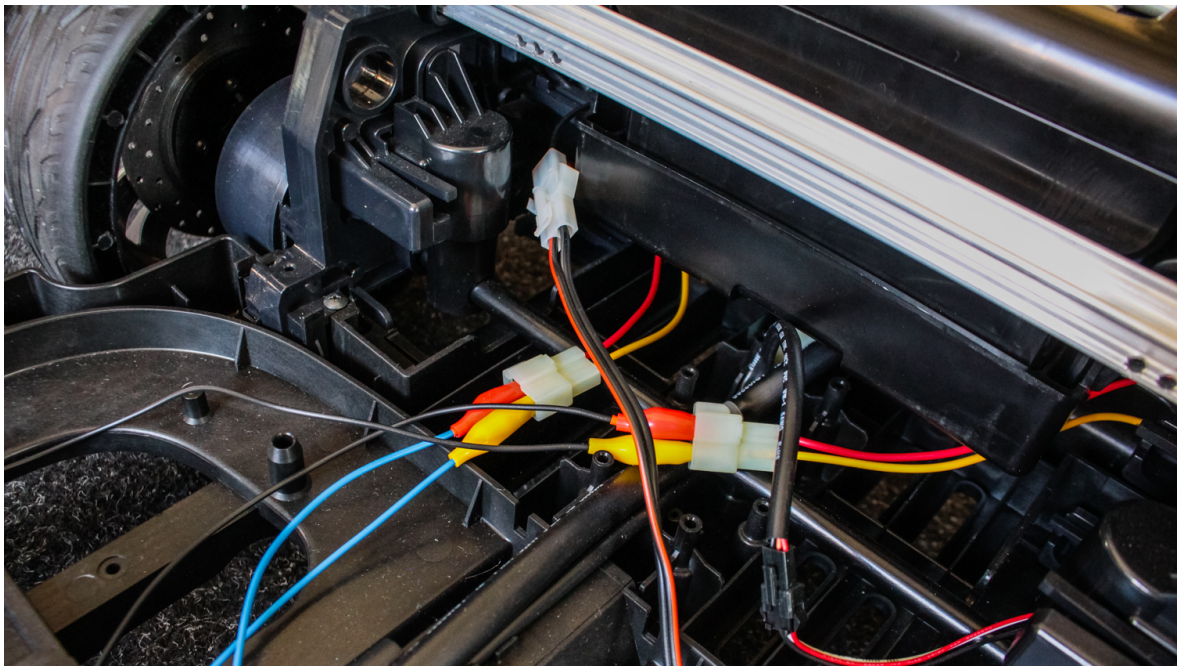
Obrázek C.2: Dálkoměr a tlačítko STOP



Obrázek C.3: Pohled na motor a hřeben zatáčení



Obrázek C.4: Pohled na Arduino a H-můstky. Napájení z baterie je rozvedeno pomocí WAGO svorkovnic.



Obrázek C.5: Detail vývodu napájení z baterie s konektorem a vodičů s krokosvorkami k motorům

Příloha D

Užitečné příkazy pro použití Raspberry Pi

```
sudo apt-get update # aktualizace instalovanych programu a knihoven

# vypnuti Raspberry Pi
sudo halt
# restartovani
sudo reboot

# nastaveni Raspberry Pi
sudo raspi-config

# zobrazeni bezicich procesu a vyuziti prostredku
top

# vypsani nastaveni wifi
iwconfig
# vypsani SSID dostupnych Wi-Fi siti
iwlist wlan0 scan | grep ESSID

# vypsani konfigurace GPIO portu
gpio -v
gpio readall
```

Nastavení pevné IP adresy

Pro nastavení pevné IP adresy je nutno editovat konfigurační soubor `sudo nano /etc/dhcpd.conf` a přidat do něj nastavení pro příslušnou Wi-Fi síť, například:

```
ssid raspberry_wifi
static ip_address=192.168.137.2/24
static routers=192.168.137.1
```