ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

ÚSTAV MECHANIKY, BIOMECHANIKY A MECHATRONIKY

Odbor mechaniky a mechatroniky

Bakalářská práce

Výukový model autonomně parkujícího vozidla řízený z prostředí Matlab/Simulink

Praha, 2021 **Eva Valentová**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Valentová** Jméno: **Eva** Osobní číslo: **481769**

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:

Výukový model autonomně parkujícího vozidla řízený z prostředí Matlab/Simulink

Název bakalářské práce anglicky:

Pokyny pro vypracování:

1) Seznamte se s problematikou autonomního parkování vozidel.
2) Vytvořte simulační model autonomně parkujícího vozidla v prostředí Matlab/Simulink.
3) Navrhněte a sestavte model vozidla z komponent LEGO Mindstorms.
4) Propojte sestavený model s řízením z prostředí Matlab/Simulink.
5) Experimentálně ověřte funkčnost vytvořeného modelu.

Seznam doporučené literatury:

[1] Sekerka, M.: Systém pro plánování složitějších parkovacích manévrů, diplomová práce, FS ČVUT v Praze, 2018.
[2] Vrbský, L.: Experimentální model vozidla s asistentem pro couvání s přívěsem, bakalářská práce, FS ČVUT v Praze, 2016.
[3] Lv, Z., Zhao, L., Liu, Z.: A Path-Planning Algorithm for Automatic Parallel Parking, Proceedings of 2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control, IMCCC 2013, 2013, pp. 474-478.
[4] Wang, L., Guo, L., He, Y., Path Planning Algorithm for Automatic Parallel Parking from Arbitrary Initial Angle,10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, 2017, pp. 55-58.

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: **28.04.2021** Termín odevzdání bakalářské práce: **13.08.2021**

Platnost zadání bakalářské práce: _____

Ing. Petr Beneš, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Miroslav Španiel, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna 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 studentky _____

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu.

V Praze, dne …… ……

Podpis

Poděkování

Ráda bych poděkovala svému vedoucímu práce, panu Ing. Petru Benešovi, Ph.D., za jeho odborné vedení, vstřícný přístup a ochotu pomoci s jakýmkoliv problémem, který se v práci vyskytl.

Zároveň bych tímto chtěla poděkovat svým rodičům, kteří vždy byli mou největší podporou nejen ve studiu.

Anotační list

Jméno autora:	Eva Valentová
Název bakalářské práce:	Výukový model autonomně parkujícího vozidla řízený z prostředí Matlab/Simulink
Anglický název:	Educational model of an autonomously parking vehicle controlled from the Matlab / Simulink environment
Akademický rok:	2020/2021
Obor studia:	Teoretický základ strojního inženýrství
Ústav/odbor:	Ústav mechaniky, biomechaniky a mechatroniky Odbor Mechaniky a mechatroniky
Vedoucí bakalářské práce:	Ing. Petr Beneš, Ph.D.
Bibliografické údaje:	Počet stran: 53 Počet obrázků: 36 Počet příloh: 3

Klíčová slova:

Autonomní parkování, model, vozidlo, EV3 Brick, LEGO Mindstroms, Matlab, Simulink, parkovací manévr

Keywords:

Autonomous parking, model, vehicle, EV3, LEGO Mindstroms, Matlab, Simulink, parking maneuver

Abstrakt:

Tato bakalářská práce se zabývá tvorbou výukového modelu autonomně parkujícího vozidla poskládaného ze stavebnice LEGO Mindstorms EV3. Řízení vozidla je realizováno z prostředí Matlab a Simulink. V úvodních kapitolách je vytvořen kinematický model vozidla a jsou zde odvozeny rovnice potřebné pro výpočet geometrie vozidla a obecně pro vypočet trajektorie parkovacího manévru. V dalších kapitolách je ukázáno a vyhodnoceno sestavené vozidlo ze stavebnice LEGO Mindstorms. V dalších kapitolách je popsána tvorba řídicího programu v prostředí Matlab a Simulink. Dále propojení EV3 Brick s prostředím Matlab a Simulink a v poslední řadě je zde vyhodnocen finální experiment autonomně parkujícího vozidla.

Abstract:

This thesis deals with the creation of an educational model of an autonomously parking vehicle composed of a LEGO Mindstorms EV3 kit. The vehicle is controlled from Matlab and Simulink environment. In the introductory chapters, a kinematic model of the vehicle is created, and the equations needed to calculate the geometry of the vehicle and in general to calculate the trajectory of the parking maneuver are shown. In the following chapters, an assembled vehicle from the LEGO Mindstorms kit is shown and evaluated. The following chapters describe the creation of a control program in the Matlab and Simulink environment. Furthermore, the connection of the EV3 Brick with the Matlab and Simulink environment and, finally, the final experiment of an autonomously parked vehicle is evaluated here.

Obsah

Seznam obrázků

Úvod

Se stále se zvyšujícím počtem automobilů na silnicích po celém světě roste i problém s dostatečnou kapacitou parkovacích míst. Hlavní problém nastává zejména v hustě obydlených lokalitách. Parkovací místa mají tendenci se zmenšovat v důsledku maximálního využití parkovacích ploch, a tedy samotné parkování se pro některé řidiče může stávat neřešitelným úkolem [1].

Není překvapením, že samotné automobilky se snaží zvyšovat pohodlí a bezpečnost automobilů, a tedy přidávat nové a lepší funkce do automobilů za cílem zvýšení svých zisků. Jedním z funkcí, která se dnes nachází téměř ve všech nových automobilech, jsou různé modifikace parkovacích asistentů, popřípadě systémy pro samotné autonomní parkovaní automobilů. Není pochyb o tom, že tento komponent šetří mnohým řidičům nervy a zároveň snižuje riziko nehod při parkování [2] [3].

Myšlenka autonomního parkování vznikla již na začátku 21. století a během pár let vzniklo mnoho konceptů a výzkumů, jak dosáhnout pohodlí a bezpečnosti řidičů. Zároveň byl kladen důraz na vytvoření ideálního parkovacího asistenta, popřípadě autonomně parkujícího automobilu, který dokáže zaparkovat efektivně v adekvátním časovém rozmezí a zároveň i do těžko dostupných parkovacích míst. Teorií, které vznikly je velké množství a lze je dohledat v literatuře [4].

Cílem této práce je, mimo nastínění principu autonomně parkujících vozidel, zejména ukázat, jak pracovat v prostředí Malab a Simulink, jakožto silného výpočtového i simulačního nástroje, který v kombinaci s platformou LEGO Mindstorms EV3 může být vhodnou volbou pro tvorbu mechatronických modelů a výukových experimentů. Tato práce, díky detailnímu popisu, může posloužit jako návod, jak postupovat při tvorbě takových experimentů.

V první kapitole je vysvětleno, jak lze vytvořit kinematický model vozidla na základě kterého jsou tvořeny matematické vzorce, které popisují chování takového kinematického modelu. Dalším bodem této kapitoly je vytvoření trajektorie pohybu vozidla. Tato práce se pro větší přehlednost a stručnost zaměřuje pouze na podélné parkování realizované

couváním po dvou obloucích.

Pro názornost je v druhé kapitole vytvořen reálný model vozidla ze stavebnice LEGO Mindstorms EV3. Tato stavebnice dovoluje rychlé sestavení i složitějších modelů bez nutnosti vyrábění složitých dílů. Obsahuje široké spektrum dílů, motorů, senzorů a dalších komponentů a je tedy velice dobrým prostředkem pro simulování základních problémů. Dalším důvodem, proč byla použita tato stavebnice je, že hardware EV3 Brick dokáže komunikovat s prostředím Matlab a Simulink, a je tedy možné oživit libovolné modely a experimentální úlohy rychle a jednoduše jen za použití stavebnice LEGO Mindstorms EV3 a prostředí Matlab a Simulink.

V neposlední řadě je v práci naznačen postup vytvoření programu v prostředí Matlab a Simulink. Toto prostředí bylo vybránu z toho důvodu, že je kompatibilní i s dalšími hardwary, jako je například Arduino, a zároveň je dobrým prostředkem k vytváření i mnoha dalších simulací nejen mechanického charakteru. Zároveň je zde názorně popsáno, jak propojit stavebnici LEGO Mindstorms EV3 s prostředím Matlab a Simulink.

Závěrem této bakalářské práce je vyhodnocení funkčnosti vytvořeného modelu vozidla, jak z pohledu tvorby samotného programu v prostředí Matlab/Simulink, tak z pohledu konstrukce reálného modelu vozidla ze stavebnice LEGO Mindstorms EV3.

Cíl práce

Cílem této práce je vytvořit a popsat výukový model vozidla ze stavebnice Lego Mindstorms EV3, který je schopen autonomně zaparkovat do podélného parkovacího místa. Řízení vozidla je realizováno pomocí programu vytvořeného v prostředí Matlab a Simulink.

Částečné cíle k naplnění hlavního cíle:

1. Seznámení se s problematikou autonomně parkujících vozidel
2. Navržení a sestavení modelu vozidla z komponentů stavebnice Lego Mindstorms EV3
3. Vytvoření simulačního modelu autonomně parkujícího vozidla

4. Propojení sestaveného modelu s řízením z prostředí Matlab a Simulink
5. Experimentální ověření funkčnosti vytvořeného modelu

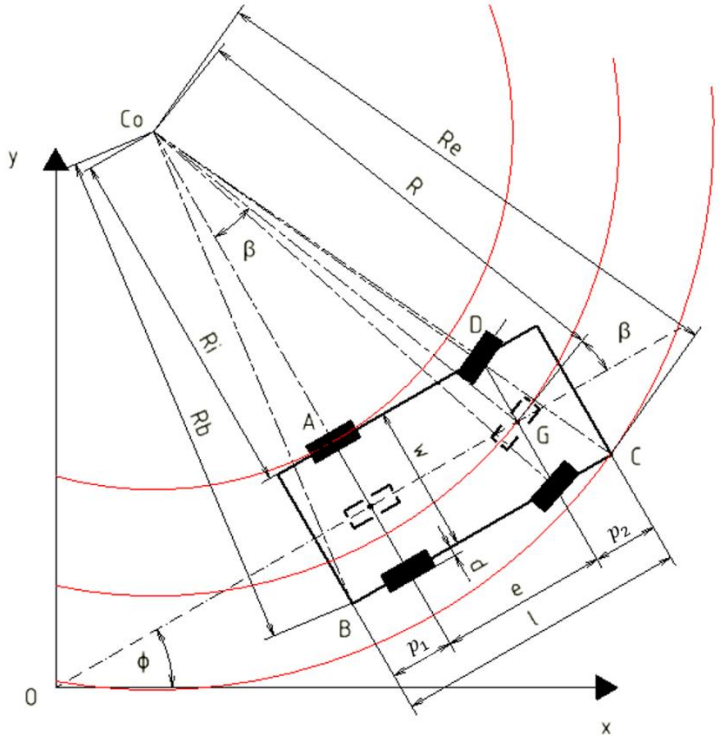
1 Problematika autonomně parkujících vozidel

1.1 Kinematický model vozidla

Prvním důležitým krokem je vytvoření kinematického modelu vozidla. Vzhledem k tomu, že předpokládáme nízkou rychlost vozidla při parkovacím manévru, lze říct, že bude nulový prokluz všech čtyř kol vozidla během manévru [5]. Za těchto předpokladů lze použít tzv. „Ackerman steering model“ (viz. Obr. 1), který předpokládá jiné natočení vnějšího a vnitřního předního kola. Pokud spustíme kolmici nejdříve k přednímu vnitřnímu kolu a poté k zadnímu vnějšímu kolu, dostaneme bod C_0, který je průsečíkem těchto kolmic. Bod C_0 je centrem rotace okolo kterého se každý jednotlivý bod kinematického modelu vozidla otáčí. Jak je zřejmé z Obr. 1, každý bod kinematického modelu vozidla se pohybuje po kružnici o poloměru vzdálenosti daného bodu od středu rotace C_0 [6].

Dále si představíme dvě fiktivní kola ležící na ose vozidla (viz. Obr. 1). První fiktivní kolo umístíme mezi kola zadní nápravy rovnoběžně se zadními koly. Druhé fiktivní kolo umístíme mezi kola přední nápravy pod příslušným úhlem β tak, aby kolmice spuštěná z tohoto fiktivního kola procházela bodem C_0. Struktura tohoto kinematického modelu vozidla se nazývá „jednostopý model“ a lze ji použít za předpokladů zmiňovaných výše za účelem zjednodušení dalších výpočtů [7] [8].

Pro náš případ parkovacího manévru jsou důležitá tři místa. Prvním z nich je bod A, který se při zatáčení vozidla pohybuje po nejmenší kružnici o poloměru R_i. Jako opak k bodu A lze vnímat bod C, který se pohybuje po největší kružnici o poloměru R_e. A třetí bod, který je třeba zmínit, je bod G, jakožto střed předního fiktivního kola pohybující se na kružnici o poloměru R [6].



Obr. 1: Kinematický model vozidla

Další rozměry, které jsou nutné uvést pro další výpočty je rozvor kol w, rozchod přední a zadní nápravy e, celková délka vozidla l, přesah zadní části vozidla p_1 a přední části vozidla p_2, a nakonec rozměr d, který použijeme na dopočítání šířky vozidla. Úhel ϕ určuje pozici vozidla v souřadnicovém systému xy [9].

Z geometrie kinematického modelu lze tedy získat tyto vzorce:

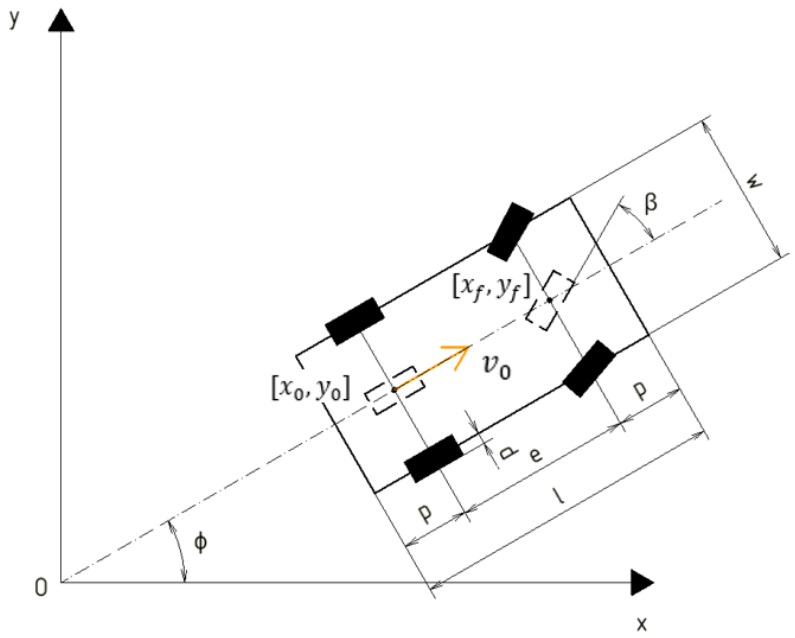
$$R = \frac{e}{sin\beta} \tag{1.1}$$

$$\tan\beta = \frac{e}{R_i + \frac{w}{2} + d} \tag{1.2}$$

$$\begin{aligned} R_i &= \sqrt{R^2 - e^2} - \frac{w}{2} - d = \sqrt{\frac{e^2}{\sin^2\beta} - e^2} - \frac{w}{2} - d = \sqrt{\frac{e^2 \cdot (1 - \sin^2\beta)}{\sin^2\beta}} - \frac{w}{2} - d \\ &= \sqrt{\frac{e^2 \cdot \cos^2\beta}{\sin^2\beta}} - \frac{w}{2} - d = \frac{e}{tg\beta} - \frac{w}{2} - d \end{aligned} \tag{1.3}$$

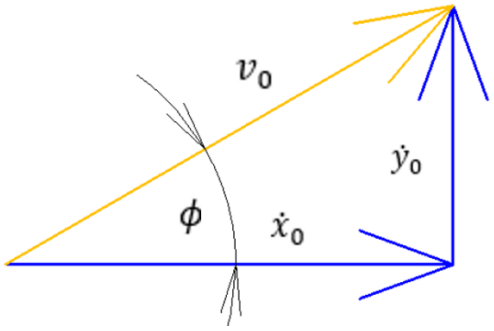
$$R_e = \sqrt{(R_i + w)^2 + (e + p_2)^2} = \sqrt{(\sqrt{R^2 - e^2} + \frac{w}{2} - d)^2 + (e + p_2)^2} \tag{1.4}$$

Z geometrie je zřetelně vidět, že čím větší bude úhel natočení kol β, tím menší budou poloměry kružnic, po kterých se pohybují jednotlivé body kinematického modelu [9].



Obr. 2: Kinematický model-pohybové rovnice

Nyní se přesuneme k určení pohybových rovnic vozidla v rovině xy. Rovnice jsou tvořeny na základě Obr. 2. Při odvozování následujících rovnic budeme vycházet z předpokladu, že střed přední i zadní nápravy se pohybují ve směru příslušného fiktivního kola. Střed zadní nápravy má souřadnice $[x_0, y_0]$ a střed přední nápravy má souřadnice $[x_f, y_f]$. Natočení vozidla v souřadném systému udává úhel ϕ vztažený ke kladnému směru osy x. Úhel β je úhel natočení předního fiktivního kola vztažený k úhlu ϕ. Uvažujeme u obou úhlů kladnou orientaci proti směru hodinových ručiček. Rychlost zadního kola v_o je nám známa [10].



Obr. 3: Rychlosti v souřadném systému xy

Vyjádření rychlosti v_0 z Obr. 3:

$$v_0 = \sqrt{\dot{x}_0{}^2 + \dot{y}_0{}^2} \tag{1.5}$$

Dále z Obr. 3 vyplývá vztah:

$$\tan\phi = \frac{\dot{y}_0}{\dot{x}_0} = \frac{\sin\phi}{\cos\phi} \tag{1.6}$$

Po úpravě vztahu (1.6) dostaneme:

$$\dot{x}_0 \sin\phi - \dot{y}_0 \cos\phi = 0 \tag{1.7}$$

Obdobně lze získat vztah pro přední nápravu:

$$\dot{x}_f \sin(\phi + \beta) - \dot{y}_f \cos(\phi + \beta) = 0 \tag{1.8}$$

Nyní určíme závislost polohy středu zadní a přední nápravy:

$$x_f = x_0 + e\cos\phi \tag{1.9}$$

$$y_f = y_0 + e\sin\phi \tag{1.10}$$

Provedeme časovou derivaci vztahů (1.9) a (1.10):

$$\dot{x}_f = \frac{dx_f}{dt} = \dot{x}_0 - e\dot{\phi}\sin\phi \tag{1.11}$$

$$\dot{y}_f = \frac{dy_f}{dt} = \dot{y}_0 + e\dot{\phi}\cos\phi \tag{1.12}$$

Dosadíme vztahy (1.11) a (1.12) do vztahu (1.8):

$$(\dot{x}_0 - e\dot{\phi}\sin\phi)\sin(\phi + \beta) - (\dot{y}_0 + e\dot{\phi}\cos\phi)\cos(\phi + \beta) = 0 \tag{1.13}$$

Vzorec roznásobíme a za pomoci goniometrických vzorců upravíme:

$$\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta \tag{1.14}$$

$$\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta \tag{1.15}$$

$$1 = \sin^2\alpha + \cos^2\alpha \tag{1.16}$$

$$\begin{aligned}\dot{x}_0 \sin(\phi+\beta) - e\dot{\phi}\sin\phi\sin(\phi+\beta) - \dot{y}_0\cos(\phi+\beta) - e\dot{\phi}\cos\phi\cos(\phi+\beta)\\ = \dot{x}_0\sin(\phi+\beta) - e\dot{\phi}\sin\phi\,(\sin\phi\cos\beta + \cos\phi\sin\beta)\\ - \dot{y}_0\cos(\phi+\beta) - e\dot{\phi}\,cos\,\phi(\cos\phi\cos\beta - \sin\phi\sin\beta)\\ = \dot{x}_0\sin(\phi+\beta) - \dot{y}_0\cos(\phi+\beta) - e\dot{\phi}\cos\beta = 0\end{aligned} \tag{1.17}$$

Z rovnice (1.17) vyjádříme $\dot{\phi}$ jako:

$$\dot{\phi} = \frac{\dot{x}_0\sin(\phi+\beta) - \dot{y}_0\cos(\phi+\beta)}{e\cos\beta} \tag{1.18}$$

Nyní dosadíme do rovnice (1.18) rovnici $\dot{x}_0 = v_0\cos\phi$ a rovnici $\dot{y}_0 = v_0\sin\phi$, které vycházejí z Obr. 3:

$$\dot{\phi} = \frac{v_0\cos\phi\sin(\phi+\beta) - v_0\sin\phi\cos(\phi+\beta)}{e\cos\beta} \tag{1.19}$$

Rovnici (1.19) opět roznásobíme, použijeme goniometrické vzorce a dostaneme:

$$\dot{\phi} = \frac{v_0\sin\beta}{e\cos\beta} = \frac{v_0}{e}\tan\beta \tag{1.20}$$

Nyní už máme všechny tři pohybové rovnice, které jsou potřebné k popisu pohybu vozidla v rovině:

$$\dot{\phi} = \omega_0 = \frac{v_0}{e}\tan\beta \tag{1.21}$$

$$\dot{x}_0 = v_0\cos\phi \tag{1.22}$$

$$\dot{y}_0 = v_0\sin\phi \tag{1.23}$$

Ze vztahu (1.21) lze získat také závislost úhlové rychlosti ω_0 na rychlosti a poloměru otáčení R_i, a to dosazením ze vztahu (1.2):

$$\dot{\phi} = \omega_0 = \frac{v_0}{e}\tan\beta = \frac{v_0}{e}\frac{e}{R_i + \frac{w}{2} + d} = \frac{v_0}{R_i + \frac{w}{2} + d} \tag{1.24}$$

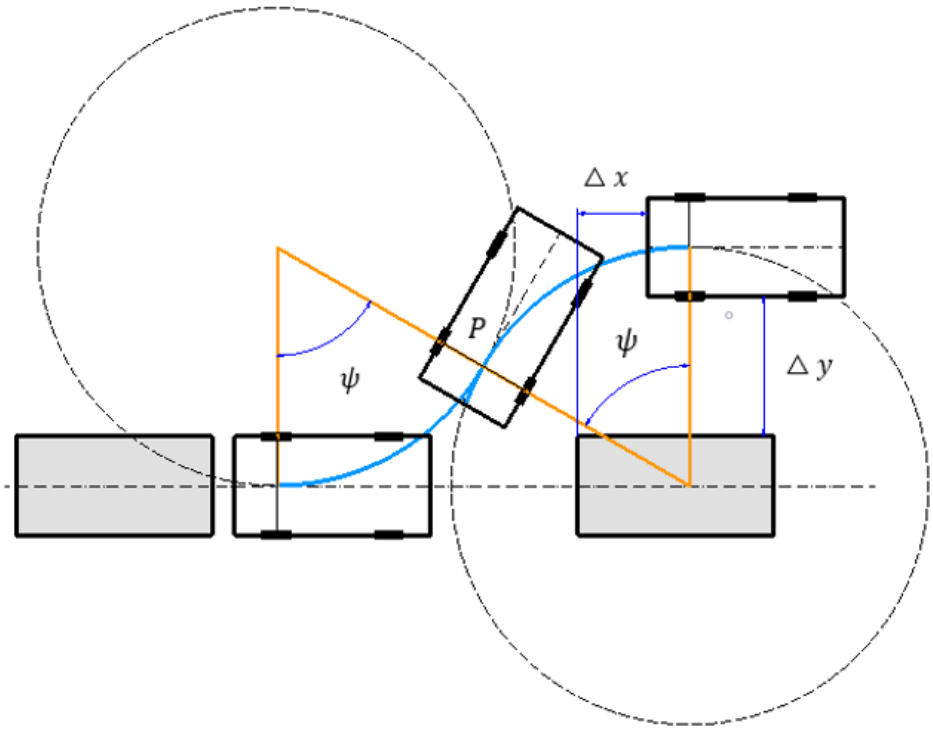
Tyto rovnice byly převzaty ze zdrojů [3] [5] [10] [11].

1.2 Trajektorie parkovacího manévru pro podélné parkování

Trajektorie parkovacího manévru pro podélné parkování může mít více variant, které lze dohledat v literatuře. Pro větší přehlednost je v této práci vybrána pouze jedna z variant.

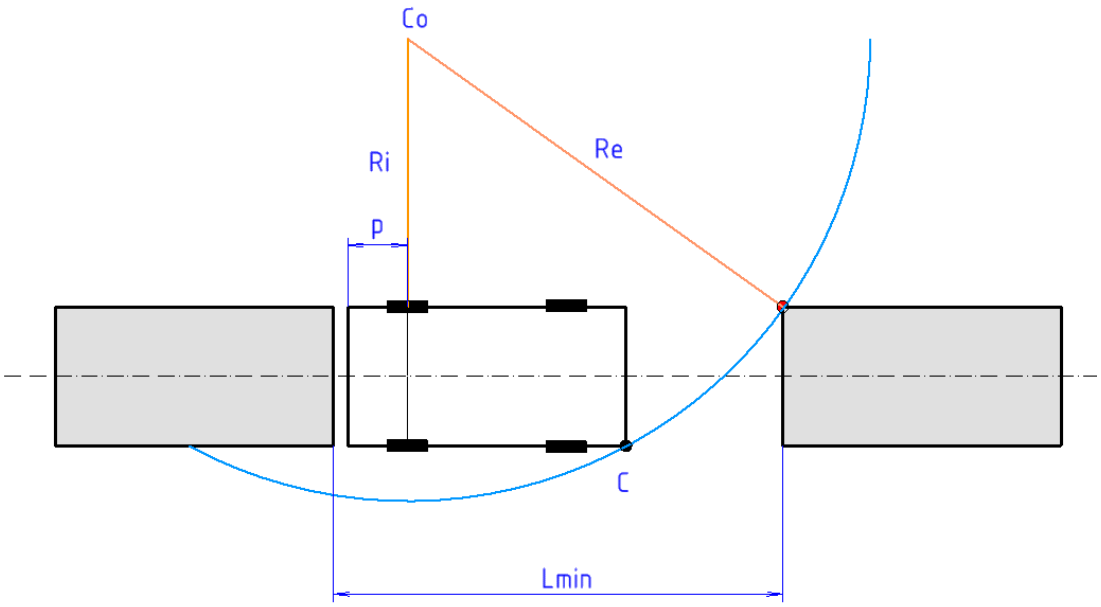
Parkovací manévr bude realizován couváním po dvou obloucích. Jak je vidět na Obr. 4 vozidlo nejprve natočí svá kola doleva do maximální polohy. Poté začne couvat po oblouku až do bodu *P*, poté otočí svá kola doprava opět do maximální polohy a zacouváním dokončí parkovací manévr [9].

Pro určení pozice vozidla v prostoru budou v praktické části využívány ultrazvukové senzory. Ultrazvukový senzor přirozeně nedokáže určit hloubku parkovacího místa, pokud zde není nějaká překážka, tudíž je cílem zaparkovat vozidlo do řady se sousedními vozidly [9].



Obr. 4: Trajektorie parkovacího manévru

Abychom předešli kolizi, je důležité vypočítat minimální velikost parkovacího místa, do kterého je vozidlo schopno zaparkovat. Pro výpočet minimální délky parkovacího místa budeme vycházet z Obr. 5 [9].



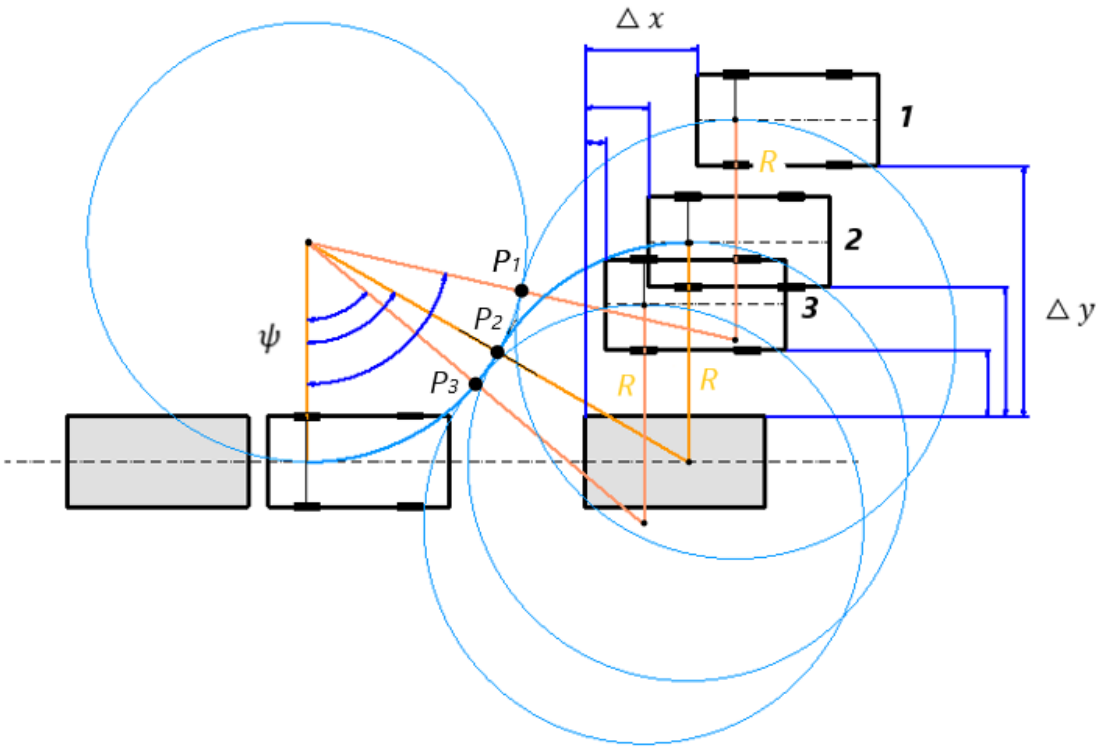
Obr. 5: Minimální délka parkovacího místa

Minimální velikost parkovacího místa se tedy vypočítá:

$$L_{min} = p + \sqrt{R_e^2 - R_i^2} \tag{1.25}$$

Jak je patrné z Obr. 5 a rovnice (1.25), L_{min} je hraniční délka parkovacího místa, kdy dojde již ke kolizi, proto je nutné délku minimálního parkovacího místa zvětšit [9]. O kolik je nutné tuto délku zvětšit, závisí na přesnosti jednotlivých komponentů vozidla, které hrají roli při vykonávání parkovacího manévru.

Z Obr. 4 je zřetelné, že oba oblouky, po kterých se vozidlo pohybuje, mají stejný poloměr R, a tento poloměr není závislý na vzdálenosti $\triangle y$ ani na vzdálenosti $\triangle x$ (viz. kapitola 1.1). Naopak úhel ψ, který je důležitý pro nalezení bodu P, kde vozidlo otáčí kola opačným směrem, na obou vzdálenostech $\triangle y$ a $\triangle x$ závisí. Tento fakt lze vidět na Obr. 6 [9].



Obr. 6: Trajektorie manévru v závislosti na poloze vozidla

Nyní sestavíme rovnice podle Obr. 7, které určí výchozí polohu vozidla $[x_i, y_i]$ před začátkem samotného parkovacího manévru, souřadnice bodu P $[x_p, y_p]$, a zároveň souřadnice středů oblouků $[x_{c1}, y_{c1}]$ a $[x_{c2}, y_{c2}]$, po kterých se vozidlo pohybuje [9].

Polohu středu $[x_{c1}, y_{c1}]$ získáme ve chvíli, kdy vozidlo detekuje parkovací místo a lze tedy použít rovnice:

$$x_{c2} = x + p \tag{1.26}$$

$$y_{c1} = R, \tag{1.27}$$

kde x je aktuální poloha vozidla.

Souřadnici y_{c2} získáme z rovnice:

$$y_{c2} = y_i - R \tag{1.28}$$

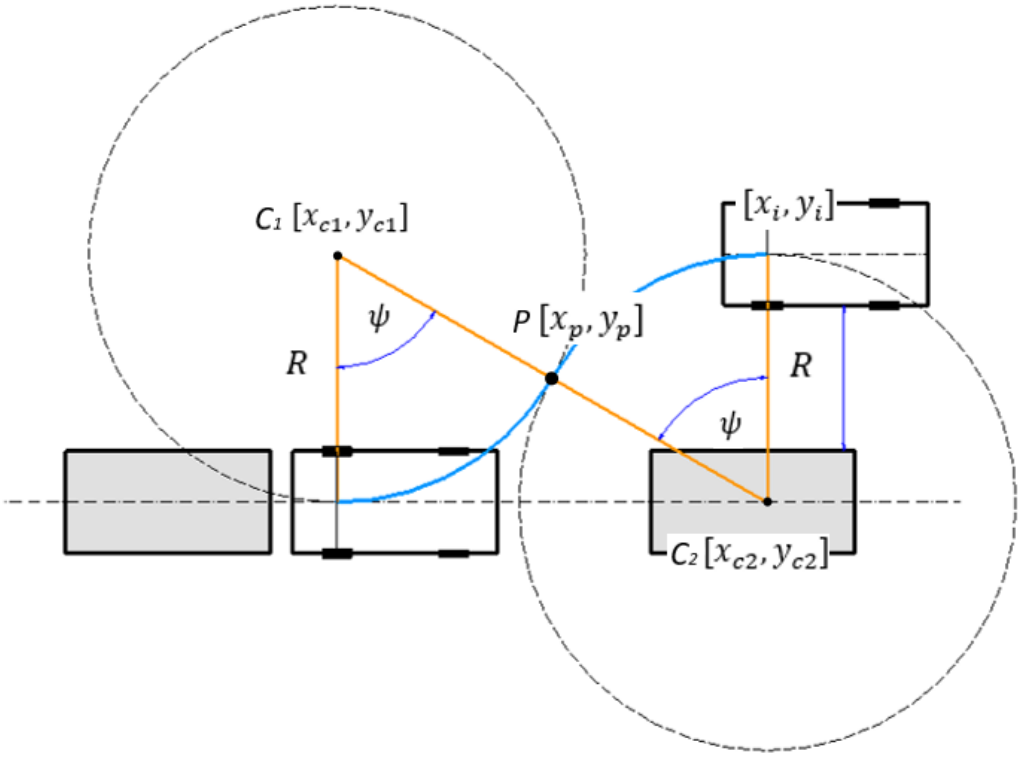
Nyní jsme schopni vypočítat souřadnice bodu P:

$$y_t = (y_{c1} + y_{c2})/2 \tag{1.29}$$

$$x_t = x_{c1} + \sqrt{R^2 - (y_t - y_{c1})^2} \tag{1.30}$$

A nakonec můžeme dopočítat souřadnici x_{c2} na základě symetrie:

$$x_{c2} = 2x_t - x_{c1} \tag{1.31}$$



Obr. 7: Trajektorie vozidla, určení globálních souřadnic

Nyní jsou určeny všechny rovnice, které jsou potřebné pro parkovací manévr vozidla.

2 Navržení a sestavení modelu vozidla z komponentů stavebnice Lego Mindstorms EV3

Pro sestavení modelu vozidla byla zvolena stavebnice Lego Mindstorms EV3. Výhodou této stavebnice je, že obsahuje širokou škálu komponentů, ať už se jedná o senzory, motory nebo jednotlivé plastové dílky [12]. Díky své variabilitě je příjemným nástrojem pro hledání ideálního modelu vozidla. Další výhodou je, že stavebnice je ke koupi na internetu i v mnoha kamenných obchodech, a je tedy lehce dostupná [13].

Stavebnice, mimo variace jednotlivých plastových dílů, obsahuje takzvanou „Brick“ neboli „Kostku“ (dále jen „Brick“). Brick v sobě obsahuje 300 MHz ARM9 procesor. Je zde 16 MB flash paměť a 64 MB RAM s možností vložení 32 GB SD karty pro zvětšení paměti. Dále je zde USB 2.0, které umožňuje připojení Wi-Fi přijímače/vysílače. Mimo jiné má v sobě Brick zabudované Bluetooth v2.1. Brick pracuje v operačním systému Linux [14]. Dokáže spolupracovat nejen s prostředím The EV3 Lab, které je přímo vyvinuto společností Lego pro tuto stavebnici [15], ale zároveň dokáže komunikovat i s prostředím Matlab/Simulink, které je využito v této bakalářské práci. Brick obsahuje čtyři vstupní porty určené pro senzory a čtyři výstupní porty určené pro motory [14]. Brick je napájena 6 tužkovými bateriemi nebo akumulátorem [12]. Další výhodou této stavebnice je variabilita propojení Brick s prostředím The EV3 Lab nebo Matlab/Simulink jak pomocí USB portu, tak přes Bluetooth nebo Wi-Fi (více k tomuto tématu v kapitole 4).

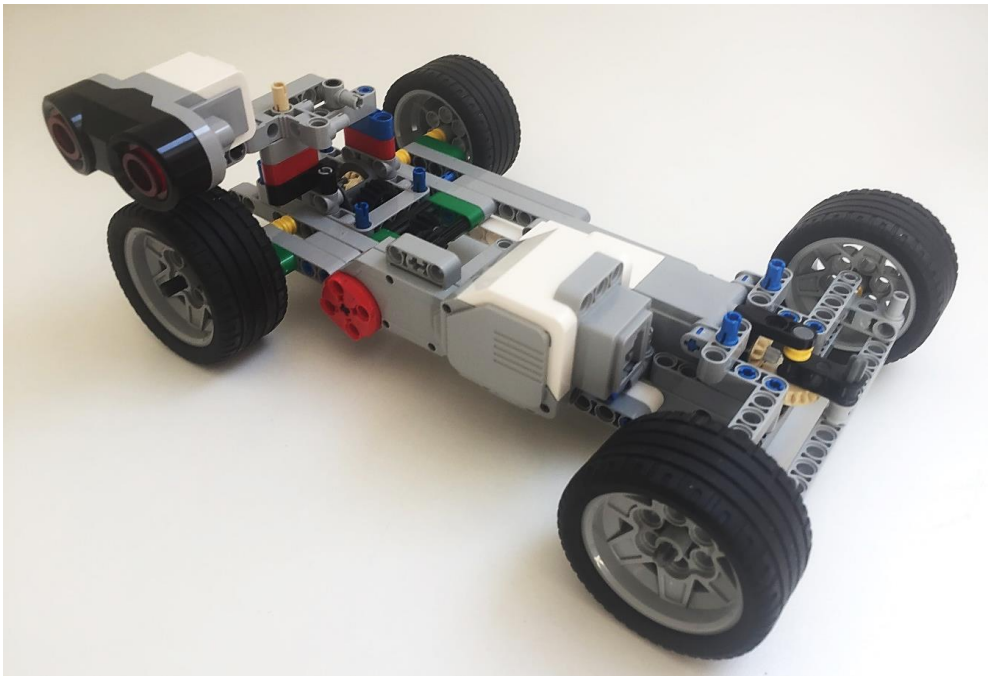
Další důležité komponenty, které stavebnice Lego Mindstorms obsahuje, jsou ultrazvukové senzory, dotykové senzory, infračervené senzory, gyroskop, dálkové ovládání, senzory barev [13], malé a velké servomotory se zabudovaným snímačem otáček s přesností jeden stupeň a další [27]. Pro náš model jsou stěžejní zejména servomotory a ultrazvukové senzory. Základní verze stavebnice Lego Mindstorms EV3 obsahuje jen určitou sestavu senzorů a motorů, nicméně další doplňky jsou volně prodejné na internetu [13].

2.1 Konstrukce modelu vozidla

Model vozidla by měl, co nejblíže demonstrovat reálné vozidlo. V konstrukci byl kladen zejména důraz na přední nápravu. Při konstrukci byla snaha dosáhnout co nejpřesnějšího natáčení předních kol, a tedy snížit nepřesnosti při samotném parkovacím manévru.

Další prvek, který se v průběhu pokusů stal důležitým, je diferenciál na zadní nápravě vozidla. Vozidlo je poháněno servomotorem na zadní nápravě, tudíž první model vozidla, který neměl diferenciál, nedosahoval takových přesností při parkování, jako právě model s diferenciálem. Důvodem bylo, že obě kola zadní nápravy se otáčela stejně, a tedy docházelo ke smýkání vždy jednoho z kol při zatáčení, a díky tomu se tvořily další nepřesnosti. Po přidání diferenciálu se tento problém vyřešil.

Finální model vozidla je na Obr. 8. Model je složen z velkého servomotoru, který pohání zadní kola vozidla, z malého servomotoru, který slouží k natáčení předních kol a ultrazvukového senzoru, který měří vzdálenost modelu vozidla od překážky, resp. od fiktivních automobilů.



Obr. 8: Model vozidla

2.1.1 Přední náprava

Parkovací asistent udává natočení kol přední nápravy při parkovacím manévru. Bylo tedy důležité najít ideální způsob natáčení předních kol.

V první konstrukční verzi modelu byl použit ozubený hřeben s malým pastorkem přímo na hřídeli malého servomotoru. Nicméně tento převod vykazoval vysoké nepřesnosti. Při spuštění servomotoru, který otáčel pastorkem, docházelo při větších rychlostech k přetáčení kol. Naopak při malých rychlostech se kola vůbec nenatáčela, protože servomotor nepřekonal tření kol o podložku. V důsledku těchto problémů byla realizována druhá varianta.

Druhou a finální variantou, která je na Obr. 9, byl převod za pomoci ozubeného kola a pastorku uloženého na hřídeli malého servomotoru. Tento převod se ukázal jako výhodný. Zejména proto, že i při vyšších rychlostech otáček servomotoru dosahoval větší přesnosti a byl tedy spolehlivější.

Maximální úhel natočení předních kol byl experimentálně zjištěn. Tedy nejprve bylo zjištěno, jak moc je možné přední kola natočit a pomocí úhloměru byl změřen maximální úhel $\beta = 36°$. Posléze pomocí převodu bylo vypočítáno, o kolik se musí otočit krokový motor.

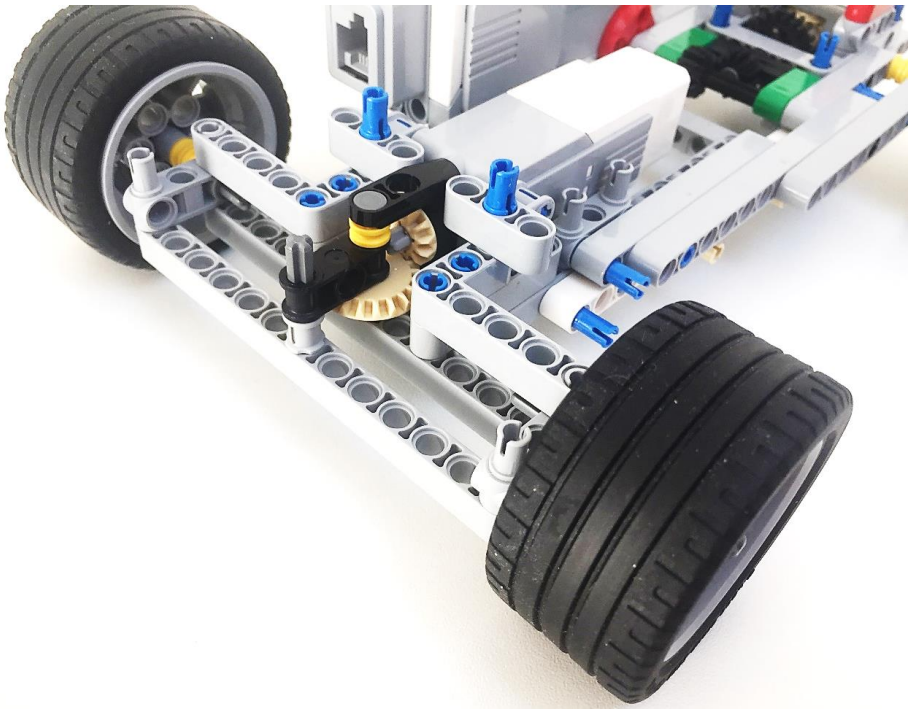
Převod na přední nápravě je

$$\mathrm{i}_p = \frac{\mathrm{z}_1}{z_2} = \frac{12}{20} = 0.6 \tag{2.1}$$

jinými slovy, pastorek uložený na hřídeli motoru má 12 zubů a ozubené kolo, které natáčí nápravu, má 20 zubů. Dále z geometrie vychází, že úhel, o který se otočí ozubené kolo na nápravě, je úhel natočení kol. Krokový motor se tedy musí otočit o úhel

$$\varphi = \frac{\beta}{i_p} = \frac{36}{0.6} = 60°, \tag{2.2}$$

aby bylo dosaženo natočení kol $\beta = 36°$.

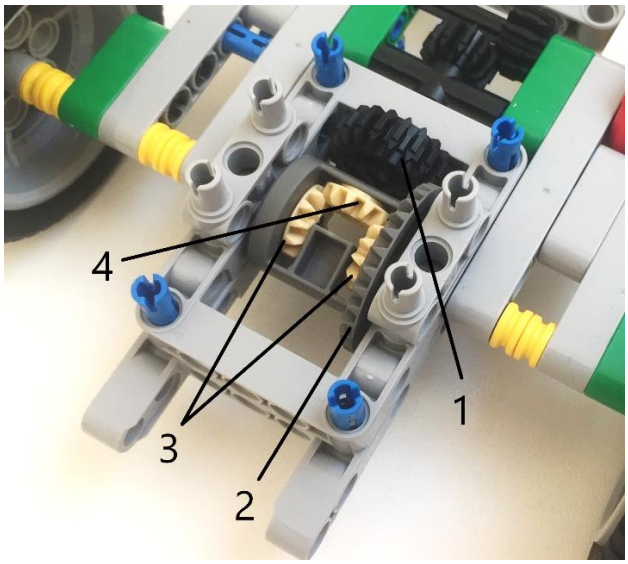


Obr. 9: Přední náprava modelu vozidla

2.1.2 Zadní náprava s diferenciálem

Zadní kola jsou poháněna velkým servomotorem. Jak je vidět na Obr. 11, je zde využito převodu 1: 1 k přenesení kroutícího momentu z motoru na diferenciál za pomoci ozubeného kola uloženého na hřídeli velkého servomotoru a druhého ozubeného kola uloženého na vstupní hřídeli diferenciálu.

Dále je zde využit diferenciál s jedním satelitem (viz. Obr. 10), který zajišťuje rozdělení poměru otáček na výstupních hřídelích. Jinými slovy se kola zadní nápravy mohou otáčet jinou rychlostí při zatáčení vozidla a nesmýkají se. Díly použité pro sestavení diferenciálu jsou součástí stavebnice. Diferenciál je poskládán z (viz. Obr. 10) 1-ozubeného kola, které přenáší kroutící moment ze servomotoru, 2-talířového kola, 3-planetových kol a 4-satelitního kola.

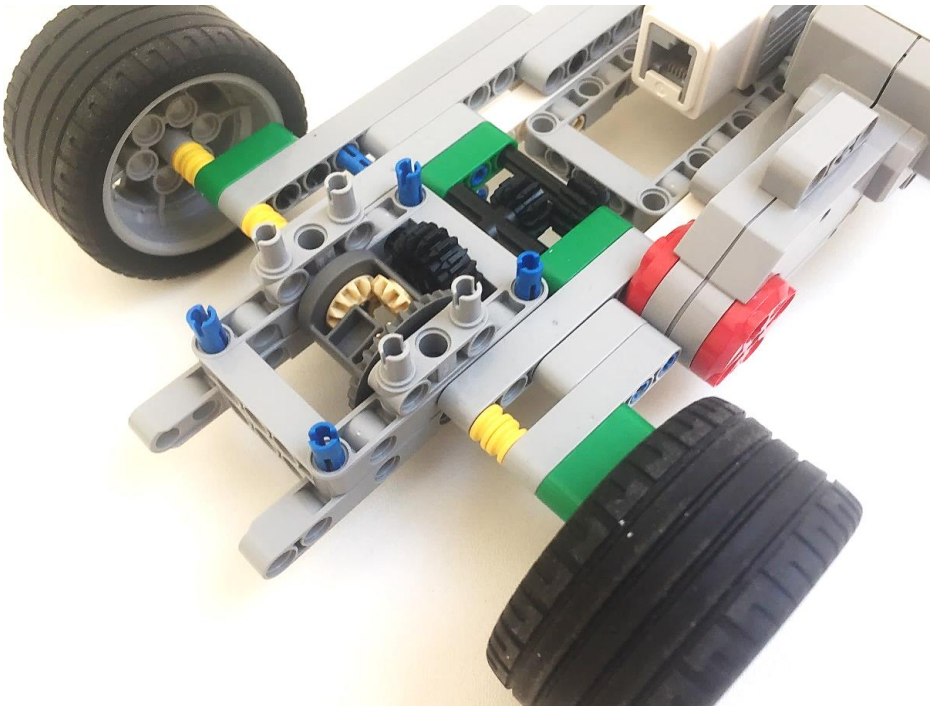


Obr. 10: Diferenciál

Převod, který je na zadní nápravě, je

$$i_z = \frac{z_1}{z_2} = \frac{20}{28} \cong 0.714, \tag{2.3}$$

tedy ozubené kolo, které přenáší kroutící moment ze servomotoru na diferenciál má 20 zubů a talířové kolo diferenciálu má 28 zubů.



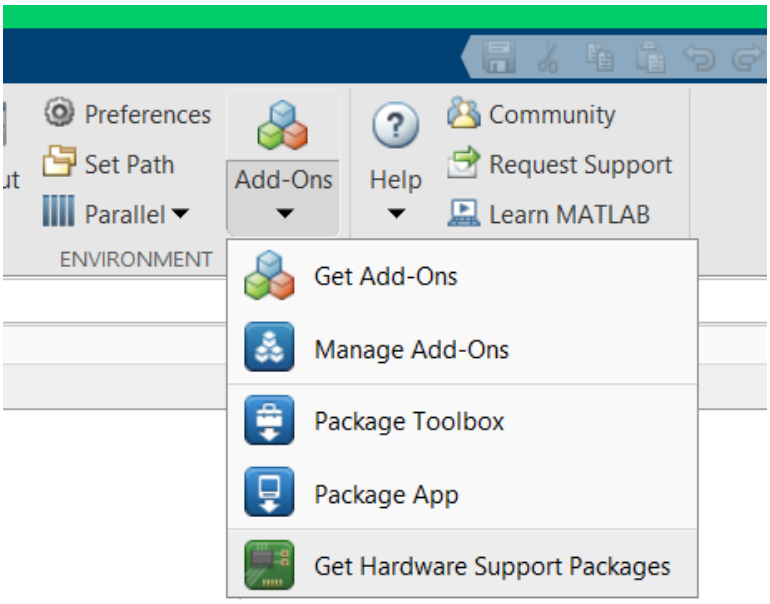
Obr. 11: Zadní náprava modelu vozidla

3 Vytvoření simulačního modelu autonomně parkujícího vozidla

Simulační model vozidla je tvořen v prostředí Matlab a Simulink, konkrétně v programu Matlab R2020b. Toto prostředí je silným výpočetním i simulačním nástrojem nejen pro úlohy a experimenty mechanického charakteru. Výhodou je uživatelsky příjemné pracovní prostředí, kdy je program tvořen za pomoci blokových schémat a díky tomu je tvorba programu přehledná. Uživatel vybírá jednotlivé bloky z knihovny a tvoří z nich logickou cestu. Knihovna bloků je velice komplexní. Zároveň je zde velké množství takzvaných „Support Packages" pro rozšíření knihovny Simulinku volně ke stažení.

3.1 Support Packages for LEGO MINDSTORMS EV3 Hardware

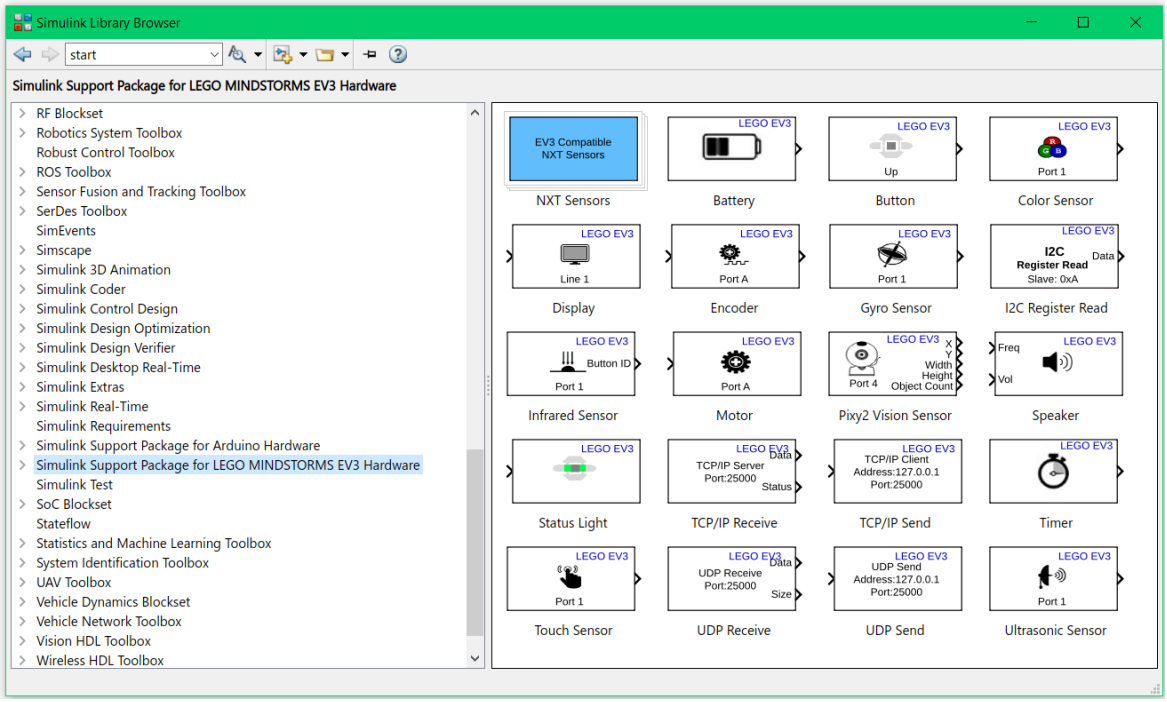
Simulink Support Package for LEGO MINDSTORMS EV3 Hardware a MATLAB Support Package for LEGO MINDSTORMS EV3 Hardware jsou nezbytnými doplňky pro to, aby prostředí Matlab a Simulink mohlo komunikovat s EV3 Brick. Tento doplněk spolu s dalšími je ke stažení v „Add-Ons" a dále „Get Add-Ons" na horní liště prostředí Matlab (viz. Obr. 12).



Obr. 12: Stažení Support Packages

Po rozkliknutí políčka „Get Add-Ons" se objeví okno, ve kterém si uživatel pouze vyhledá požadovaný „Support Packages" a klikne na tlačítko stáhnout a doplněk nainstaluje. Je důležité po stažení a nainstalování doplňku vypnout celý program a znovu ho otevřít pro správně fungování stažených doplňků.

Po nainstalování doplňku Simulink Support Package for LEGO MINDSTORMS EV3 Hardware a MATLAB Support Package for LEGO MINDSTORMS EV3 Hardware se po otevření Simulinku v knihovně bloků objeví nové bloky, které jsou určeny pro programování motorů, senzorů a dalších komponentů stavebnice LEGO Mindstorms EV3 (viz. Obr. 13).



Obr. 13: Bloky LEGO MINDSTORMS EV3 Hardware

3.2 Stručný popis jednotlivých bloků pro LEGO Mindstorms EV3

Vysvětlení, jak jednotlivé bloky v knihovně Simulinku fungují, získáme tak, že klikneme na blok pravým tlačítkem myši a zvolíme „Help for…“. Po kliknutí se rozbalí okno nápovědy s vysvětlením, jak daný blok knihovny funguje a zároveň jsou zde obvykle uvedeny jednoduché příklady použití daného bloku. Pokud na blok klikneme dvakrát levým tlačítkem, dostaneme nastavení bloku. Zde budou popsány pouze doplňkové bloky, které jsou využity v simulačním modelu této bakalářské práce.

- **Encoder** – tento blok udává rotaci motorů ve stupních. Pokud se motor točí dopředu, hodnoty jsou kladné. Pokud se motor točí dozadu, hodnoty jsou záporné. Encoder udává hodnoty portu, na který je nastaven, tzn. pokud je nastaven v enconderu port A, encoder vrací hodnoty motoru zapojeném ve EV3

Brick výstupu A. Je možné encoder nastavit do tří režimů. První režim je „No reset“, tedy měří celkové natočení motoru. Druhý režim je „Reset at each sample time“, tento režim měří jen po dobu času, který je nastaven jako „sample time“ v nastavení encoderu a pak se vyresetuje. Třetí režim je „Reset by external signal“, pokaždé když je do enconderu poslán signál, enconder se resetuje na 0.

- **Motor** – tento blok ovládá „sílu“ motoru v jednotkách prostředí LEGO EV3. Přijímá hodnoty $< -100,100 >$. Záporné hodnoty znamenají, že se motor točí dozadu, kladné hodnoty znamenají, že se motor točí dopředu. V nastavení motoru je důležité uvést správný výstup, do kterého je motor zapojen.
- **Ultrasonic Senzor** – výstup tohoto bloku je vzdálenost ultrazvukového senzoru od překážky v centimetrech. Je nutné uvést do nastavení bloku správný vstupní port, ve kterém je senzor zapojen.
- **Status Light –** na základě signálu, který je do bloku posílán, se rozsvěcí podsvícení tlačítek na EV3 Brick. Jaký signál je třeba poslat pro určitou barvu, je uvedeno v nastavení bloku.
- **Infrared Sensor** – lze použít pro tři různá nastavení. První je, že senzor měří vzdálenost od objektu. Druhé nastavení je měření vzdálenosti a směru tzv. „beacon“ neboli majáku. A třetí možnost je, že EV3 Brick detekuje, jaká tlačítka jsou mačkána na ovladači. Je důležité opět v nastavení zvolit správný port, kde je infračervený senzor zapojen, a zároveň, při používání ovladače, je důležité nastavit správný kanál v bloku v Simulinku a na samotném ovladači.

3.3 Princip realizace simulačního modelu autonomně parkujícího vozidla

První fází při tvorbě simulačního modelu bylo vytvoření scriptu v Matlabu. Tento skript obsahuje základní parametry vozidla jako je průměr kol d, šířka vozidla w, rozvor kol e, přesah přední a zadní části vozidla p, maximální úhel natočení kol β, převod natáčení předních kol p_p a převod na diferenciálu p_z. Na základě těchto hodnot se celý program

přepočítává tzn. pro libovolné vozidlo stačí přepsat jednotlivé hodnoty a program se přepočítá na dané vozidlo. V tomto skriptu jsou dále uvedeny rovnice z kapitoly 1.1, které přepočítávají geometrii vozidla a minimální délku parkovacího místa L_{min}.

Po vytvoření tohoto skriptu bylo vytvořeno samotné blokové schéma v Simulinku. Celý program je založený na posílání signálů do dvou motorů, které pohání vozidlo a natáčí kola přední nápravy.

Samotná logická cesta programu je rozdělena do tří částí. V první části vozidlo jede okolo zaparkovaných vozidel a detekuje parkovací místo. Vyhodnocuje existenci parkovacího místa na základě hodnot, které vychází z ultrazvukového senzoru. Jinými slovy, pokud ultrazvukový senzor měří větší vzdálenost, než je šířka vozidla, program detekuje parkovací prostor. Zároveň ve chvíli, kdy ultrazvukový senzor detekuje parkovací prostor, program začíná počítat o kolik stupňů se motor pootočí na základě hodnot z encoderu při stálé detekci místa ultrazvukovým senzorem. Stupně jsou přepočteny na reálnou vzdálenost parkovacího místa jednoduchým vzorcem:

$$\mathrm{L} = \mathrm{i_z} * \varphi * d, \tag{3.1}$$

kde L je délka parkovacího místa, i_z je převod na zadní nápravě, φ je úhel, o který se motor pootočil při detekci místa a d je průměr kol. Pokud je místo vyhodnoceno jako příliš krátké, vozidlo pokračuje dál v jízdě a hledá nové parkovací místo. Tento cyklus se opakuje, dokud vozidlo nenajde parkovací místo, které je dostatečně dlouhé a široké pro zaparkování vozidla. Ve chvíli, kdy program zaznamená dostatečně dlouhé a široké parkovací místo, přepočítá ideální pozici pro začátek parkovacího manévru opět pomocí vzorců z kapitoly 1.1 a dopraví se na ideální pozici.

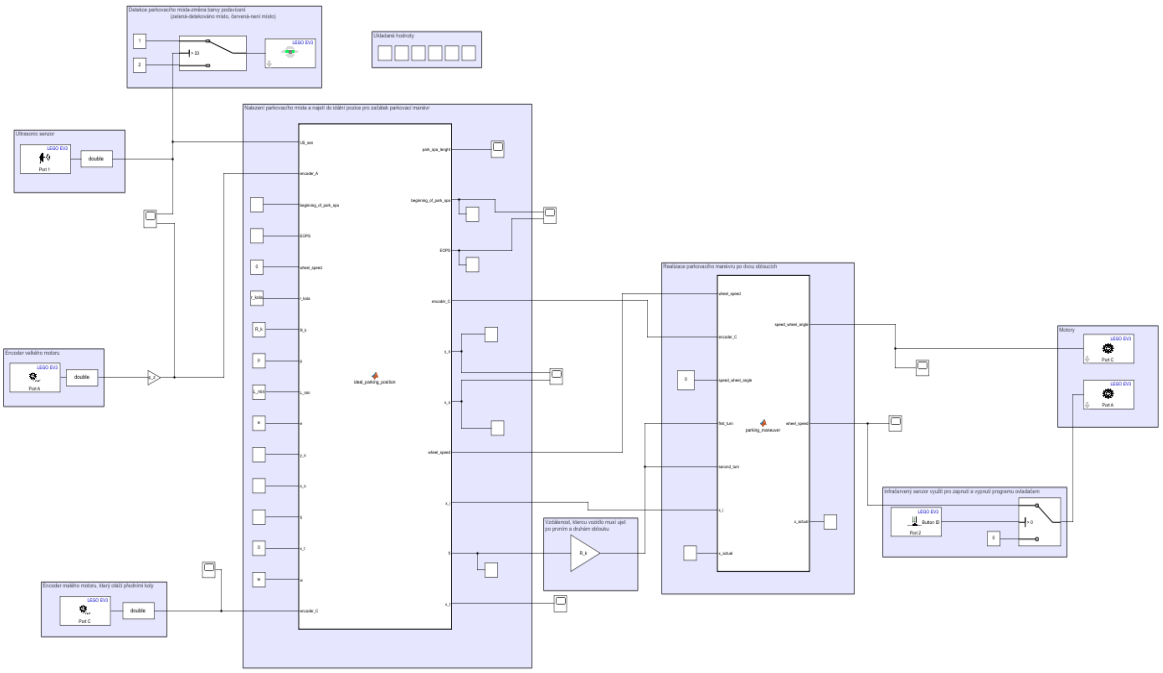
Ve chvíli, kdy se vozidlo dostalo do ideální pozice, je vyslán signál do motoru přední nápravy, která pootočí přední kola vozidla na požadovaný úhel. Posléze je vyslán signál do motoru, který vozidlo pohání. Vozidlo couvá po oblouku do nově vypočítané pozice ideální pro otočení předních kol na druhou stranu. Vozidlo se zastaví v nové pozici.

V poslední části je vyslán opět signál do motoru přední nápravy, který pootočí

předními koly na opačnou stranu. Poté je opět vyslán signál do motoru, který pohání vozidlo. Vozidlo začne couvat a dopraví se do konečné pozice, která je opět vypočítána programem. Zde je parkovací manévr ukončen.

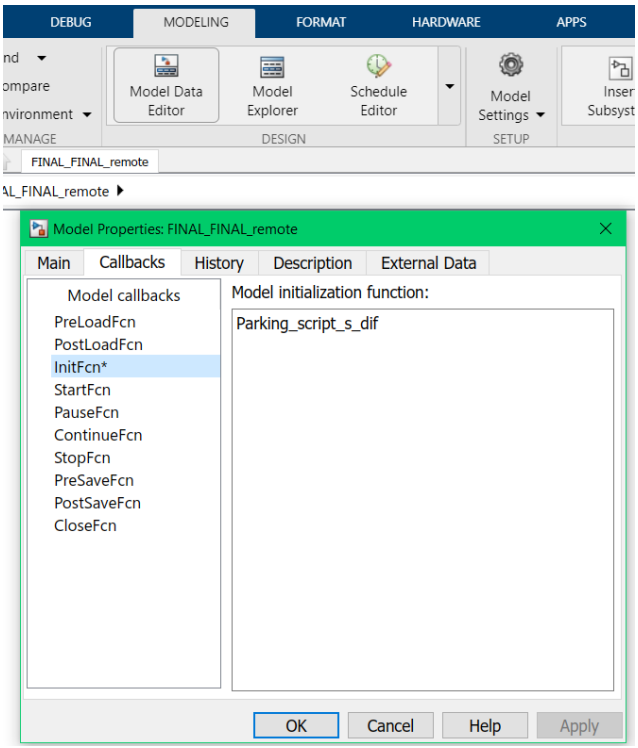
3.4 Blokové schéma simulačního modelu autonomně parkujícího vozidla

Řízení vozidla je realizováno sestavením blokového schématu v prostředí Simulink a zároveň jsou zde využity skripty vytvořené v prostředí Matlab. Jak může vypadat sestavené blokové schéma pro autonomně parkující vozidlo lze vidět na Obr. 14.



Obr. 14: Blokové schéma simulačního modelu

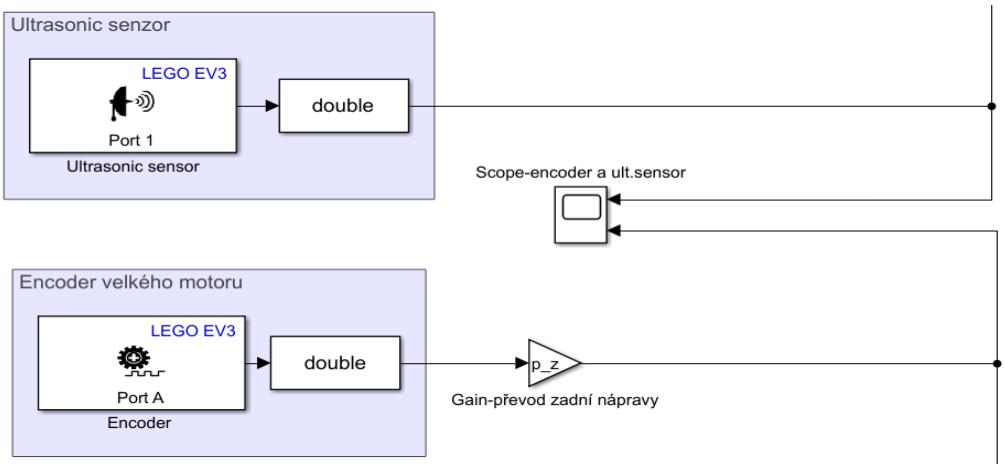
Proto, aby skript z Matlabu spolupracoval s programem tvořeném v Simulinku a posléze se nahrál do EV3 Brick, je nutné ho zadat do tzv. „Callbacks“. Na horní liště Simulinku zvolíme záložku „MODELING“>“Model Settings“>“Model Properties“>“Callbacks“ >“InitFcn“ a zde zadáme název skriptu, který má program v Simulinku zavolat při spuštění simulace (viz. Obr. 15).



Obr. 15: Callbacks v Simulinku

V simulačním modelu jsou využity, kromě bloků určených pro stavebnici LEGO Mindstorms EV3 z kapitoly 3.2, bloky ze základní knihovny Simulinku. Simulink disponuje velice bohatou základní knihovnou bloků. Zde budou popsány jen ty bloky, které byly využity pro tvorbu tohoto konkrétního simulačního modelu autonomně parkujícího vozidla. Informace k jednotlivým blokům jsou získány z „Help“ Matlabu a Simulinku.

Data Type Conversion převádí vstupní signál na zadaný datový typ. Tento blok je v schématu využit hned několikrát. Ultrazvukový senzor generuje signál v datovém typu uint8, nicméně encondery generují signály v datovém typu int32. Pokud posíláme signály různého datového typu do jednoho bloku, simulace nemusí fungovat. Zároveň některé bloky dokážou pracovat jen s určitým typem dat. Informace k tomu, jaký typ dat blok generuje, popřípadě jaký typ dat dokáže číst, jsou uvedeny v nápovědě bloku (viz. kapitola 3.2) [16].



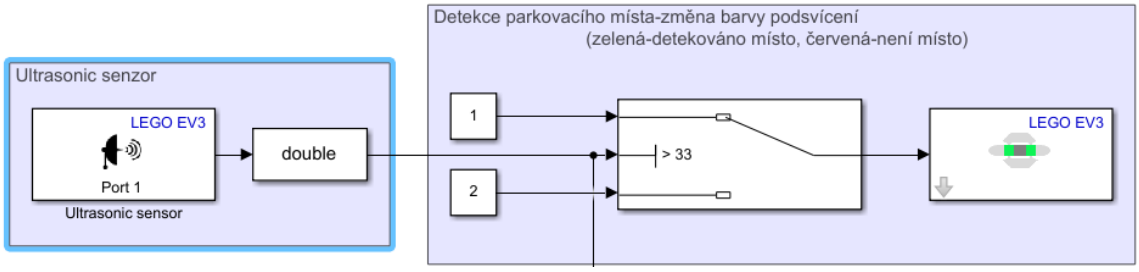
Obr. 16: Data Type Conversion, Scope, Gain

Jak je vidět na Obr. 16, signál z enconderu a ultrazvukového senzoru je převeden na datový typ „double“. Datových typů je velké množství a volí se v závislosti na velikosti čísla, které chceme uložit. Přirozeně čím větší číslo ukládáme, tím více zabíráme paměti. Datový typ „double“ je 64bitový s plovoucí desetinou čárkou a dvojitou přesností v přibližném rozsahu od -1.7^{308} do -4.9^{-324} pro záporná čísla a ve stejném rozsahu pro kladná čísla [17]. Pro takto jednoduchý simulační model je výběr datového typu založen hlavně na tom, jak moc přesné chceme výsledky simulace. Pro náš případ není třeba přemýšlet nad tím, jak velkou část paměti zabereme, neboť simulační model je jednoduchý. Nicméně u složitějších programů je třeba dbát i na tuto věc. Zároveň lze také využít takzvané *„unsigned“* datové typy, které se vyznačují tím, že začínají na *„u“*, jako je například uint8, uint16 atd. Tyto typy dat dokážou ukládat pouze kladné hodnoty ovšem ve dvojnásobném rozsahu do kladných čísel než jejich protějšky int8, int16 atd., které dokážou uložit čísla kladná i záporná, avšak v polovičním rozsahu než unsigned datové typy. Nazývají se „signed“ [18].

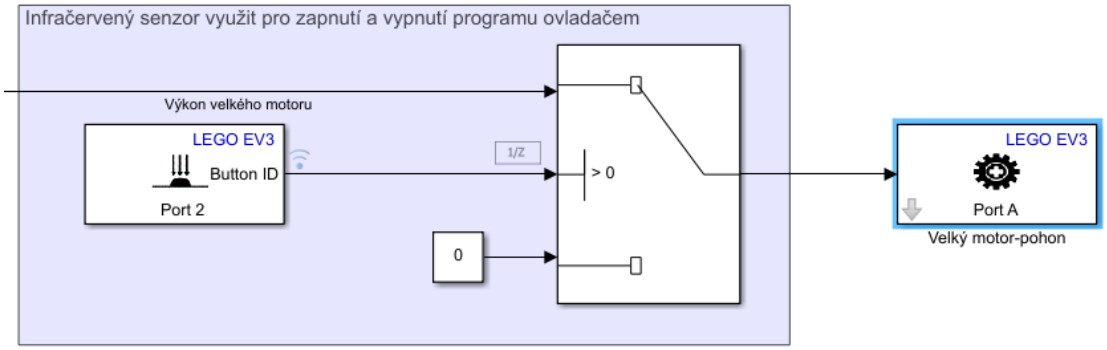
Scope zobrazí signál, který je generovaný během simulace. Pro případ, který je ukázán na Obr. 16, Scope zaznamená průběh hodnot z ultrazvukového senzoru a z enconderu velkého motoru. Scope zobrazí data v grafech a umožňuje v těchto grafech velice jednoduše hledat a dělat základní operace. Zároveň je možné tyto data převést ze Scope do WorkSpace v Matlabu [19].

Gain násobí vstupní signál konstantou. Tento prvek je použit na Obr. 16, kdy hodnota z enconderu velkého motoru je násobena převodem, který je umístěn na zadní nápravě [20].

Switch kombinuje více signálů do jednoho na základě zvoleného kritéria. Tento blok byl využit ve dvou případech. První z nich je změna podsvícení tlačítek na EV3 Brick při detekci parkovacího místa viz. Obr. 17. Pokud ultrazvukový senzor zaznamenává místo větší, než je šířka uvedená v bloku Switch, blok pošle dále signál s hodnotou 1 a podsvícení svítí zeleně. V opačném případě, kdy ultrazvukový senzor nezaznamenává místo větší, než je uvedeno, blok Switch pošle signál s hodnotou 2 a podsvícení je červené. Obdobně byl blok Switch využit pro spuštění a vypnutí simulace pomocí ovladače (viz. Obr. 18). Pokud se na ovladači stiskne tlačítko značící 0, infračervený senzor pošle signál s hodnotou 0, blok Switch opět vyšle signál s hodnotou 0, velký motor stojí a simulace neprobíhá. Ve chvíli, kdy se na ovladači stiskne tlačítko, které znamená číslo větší než 0, infračervený senzor pošle do bloku Switch toto číslo a do velkého motoru jde signál s požadovanou hodnotou [21].



Obr. 17: Switch-změna podsvícení při detekci parkovacího místa



Obr. 18: Switch-zapnutí a vypnutí simulace pomocí ovladače

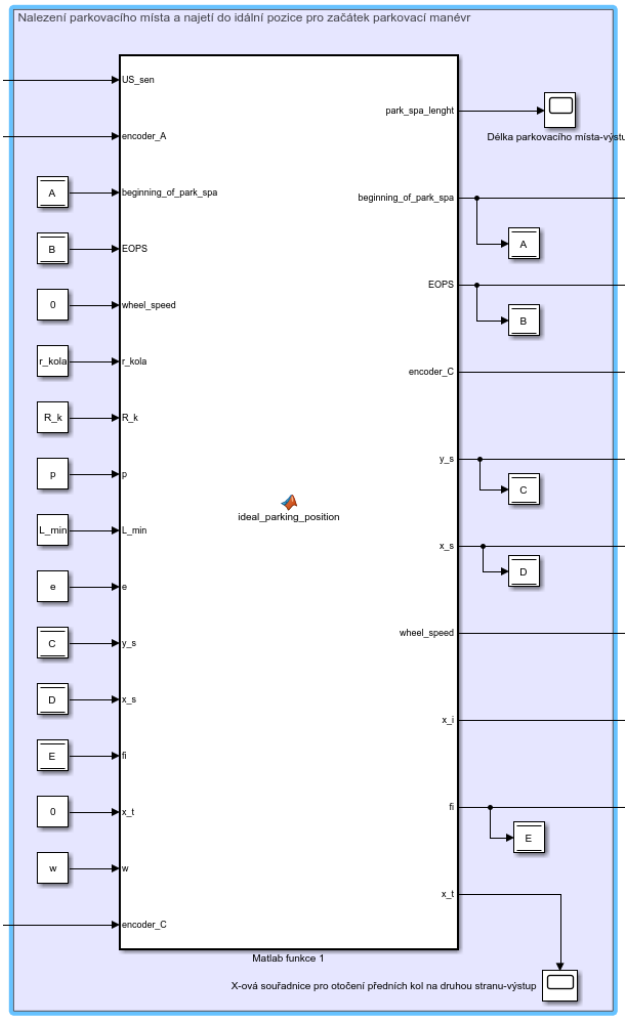
MATLAB Function dovoluje vložit do blokového schématu funkci, která se realizuje ve skriptu tvořeném v Matlabu. Tento blok je klíčovým v celém simulačním modelu. Blok funguje tak, že jsou do něj přiváděny vstupní nezávislé proměnné a opět z něj odcházejí výstupní proměnné. Pokud otevřeme tento blok, dostaneme se do skriptu Matlabu, kde je funkce zapsaná ve tvaru:

$$function[výstupní\ proměnné] = název\ funkce(vstupní\ proměné).$$

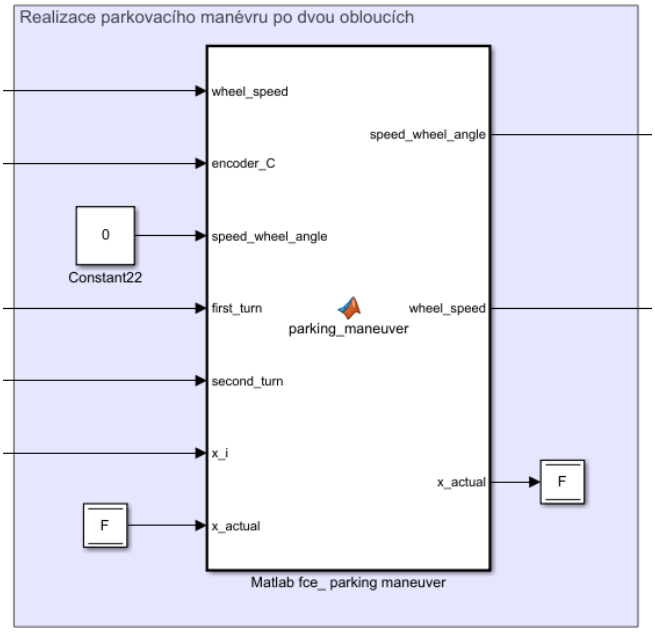
Posléze je dále do skriptu vypisována logická cesta pomocí příkazů nebo dalších funkcí [22]. V tomto simulačním modelu je využit pouze ten nejznámější a nejsnazší příkaz *„if, elseif, else"* [23]*:*

```
if expression
        statements
elseif expression
        statements
else
        statements
end
```

Blok MATLAB Function byl využit pro detekci parkovacího místa, nalezení ideální pozice pro začátek parkovacího manévru (viz. Obr. 19) a dále k realizaci parkovacího manévru (viz. Obr. 20) [22].

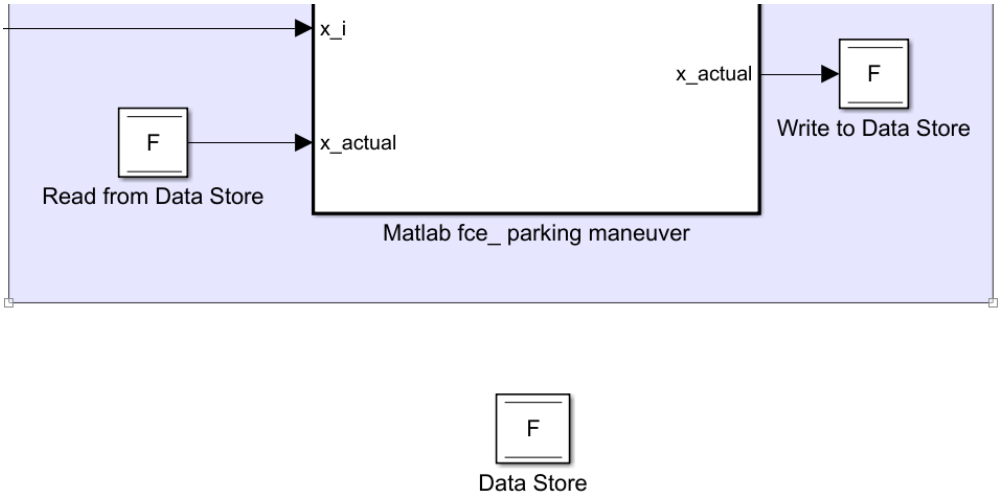


Obr. 19: MATLAB Function-detekce parkovacího místa a nalezení ideální pozice



Obr. 20: MATLAB Function-parkovací manévr

Data Store (viz. Obr. 21) zapisuje signál a ukládá ho do paměti, ze které je možné ho opět získat. Tento blok byl využit například pro uložení hodnoty enconderu, při první detekci parkovacího místa. Následně byla tato hodnota znovu použita k přepočítávání délky parkovacího místa [24].



Obr. 21: Data Store

Správným sestavením uvedených bloků a správným vytvořením logické cesty vznikne simulační model autonomně parkujícího vozidla, který je připraven k nahrání do EV3 Brick.

4 Propojení sestaveného modelu s řízením z prostředí Matlab a Simulink

Propojení sestaveného modelu s řízením z prostředí Matlab a Simulink je možné hned několika způsoby, a to pomocí USB, WiFi nebo Bluetooth. Zároveň, jak je popsáno v kapitole 3.1, pro řízení sestaveného modelu ze stavebnice LEGO Mindstorms EV3 z prostředí Matlab a Simulink, je třeba stáhnout a nainstalovat Simulink Support Package for LEGO MINDSTORMS EV3 Hardware a MATLAB Support Package for LEGO MINDSTORMS EV3 Hardware [25].

4.1 LEGO Mindstorms EV3 Firmware Version

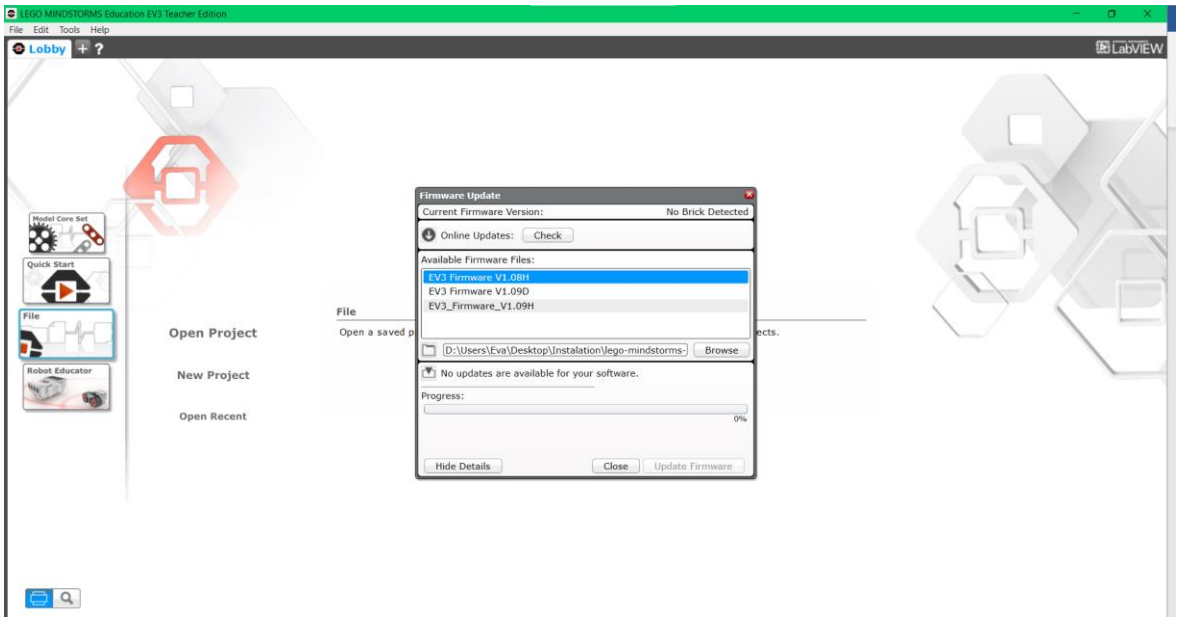
Aby bylo možné propojit EV3 Brick s prostředím Matlab a Simulink, je třeba mít nainstalovaný správný firmware na EV3 Brick. Jaký firmaware je nainstalován v EV3 Brick je uvedeno v „Setting“>„Brick Info“ na EV3 Brick (viz. Obr. 22) [25].

Obr. 22: EV3 Brick Firmware

Na stránkách Matlab a Simulink je uvedeno, že firmware by měl být V1.03E [25] nebo novější, ačkoliv po nainstalování nejnovějšího firmware V1.09E se ukázalo, že propojení nefunguje. Jako ideální volba se ukázal firmware V1.08H, který funguje bez problému.

Pro nainstalování firmware je ideální program The EV3 Lab, který je vyvinut společností LEGO přímo pro programování a komunikaci s LEGO Mindstorms EV3 Brick. Tento program je ke stažení na následujících stránkách https://education.lego.com/en-

us/downloads/retiredproducts/mindstorms-ev3-lab/software. Po instalaci a otevření tohoto programu se objeví prostředí The EV3 Lab. Po rozkliknutí „Tools“ na horní liště se objeví nabídka. V této nabídce je možnost „Firmware Update“. Po kliknutí na tuto možnost se objeví okno z Obr. 23. Zde vyberete verzi firmware, kterou chcete nahrát do své EV3 Brick. The EV3 Lab vždy nabízí pouze nejnovější verzi firmware, takže pokud je třeba nahrát jiná verze, je nutné ji stáhnout do počítače a posléze vyhledat v „Firmware Update“ okně. Pro správném nainstalování firmware je třeba zapnout EV3 Brick a propojit ji s počítačem pomocí USB, které je součástí stavebnice.



Obr. 23: The EV3 Lab

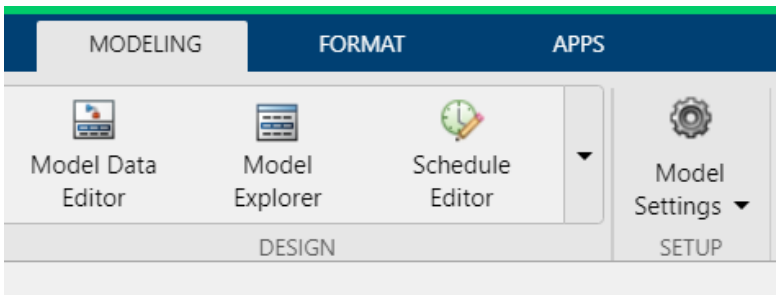
4.2 Propojení EV3 Brick pomocí USB

USB kabel je součástí stavebnice. Mini-USB port se zapojí do EV3 Brick a klasický USB port se zapojí do počítače. Propojení EV3 Brick s počítačem je možné zkontrolovat v Matlabu v Command Window zadáním příkazu *„myev3 = legoev3('USB')“*. Po zadání tohoto příkazu se v Command Window vypíšou informace o EV3 Brick jako je verze firmware, stav baterie, zapojené senzory a další (viz. Obr. 24). Proto, aby tento příkaz fungoval, je třeba vymazat příkazem *„clear all“* Workspace Matlabu. Zároveň v pravém horním rohu EV3 Brick je napsáno „USB“ [25].

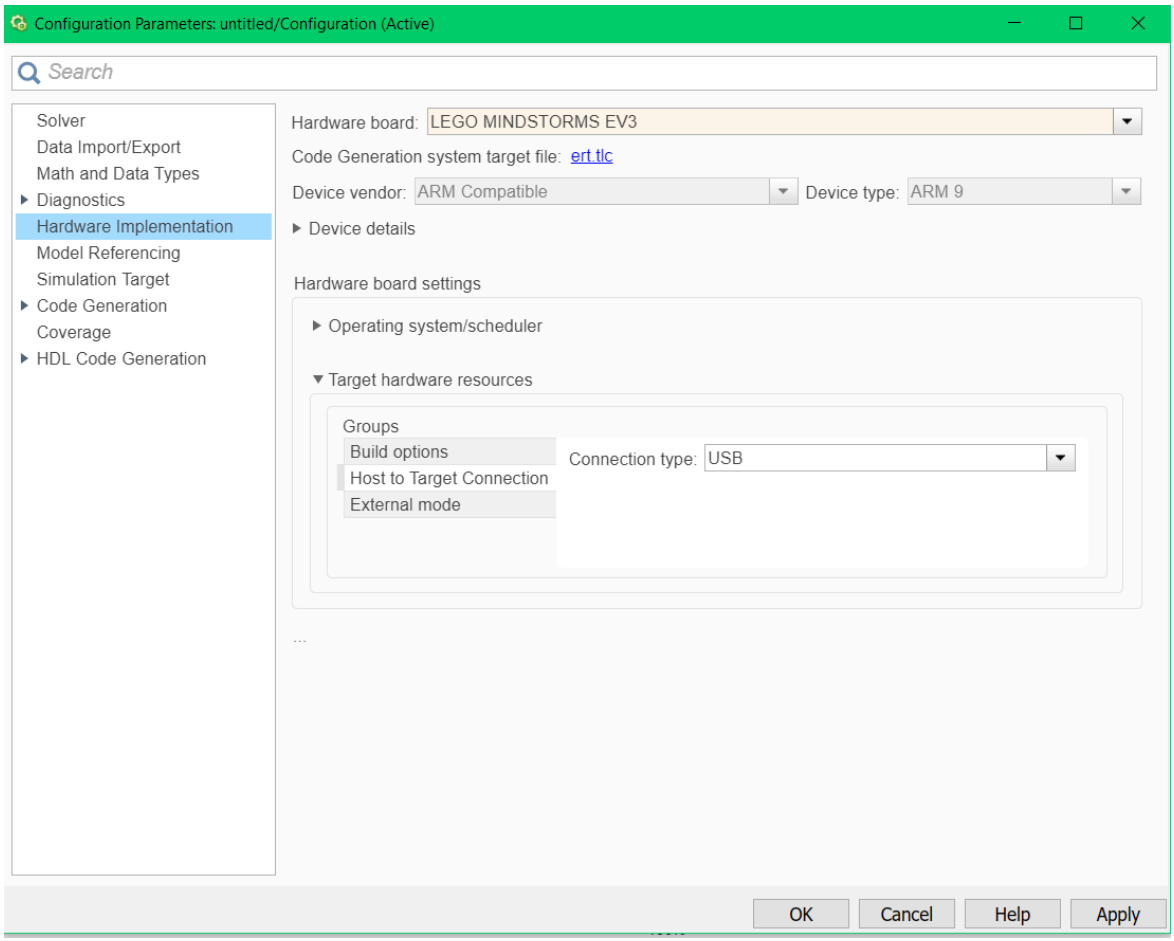
```
Command Window
>> myev3 = legoev3('USB')

myev3 =

  legoev3 with properties:

      FirmwareVersion: 'V1.08H'
           HardwareID: []
            IPAddress: []
    CommunicationType: 'USB'
         BatteryLevel: 100
     ConnectedSensors: {'sonic'  'infrared'  ''  ''}
```

Obr. 24: Command Window-propojení pomocí USB

Pokud chceme nahrát program do EV3 Brick, je nutné nastavit správný „Hardware board“. Na horní liště Simulinku zvolíme záložku „MODELING“>“Model Settings“>“Hardware Implementation“ (viz. Obr. 25 a Obr. 26). Vybereme správný Hardware board LEGO MINDSTORMS EV3 (tento Hardware board se ukáže v nabídce až po nainstalování Simulink Support Package for LEGO MINDSTORMS EV3 Hardware z kapitoly 3.1). Rozklikneme záložku „Host to Target Connection“ a v kolonce „Connection type“ zvolíme „USB“. Nakonec zvolíme „Apply“ (viz. Obr. 26).

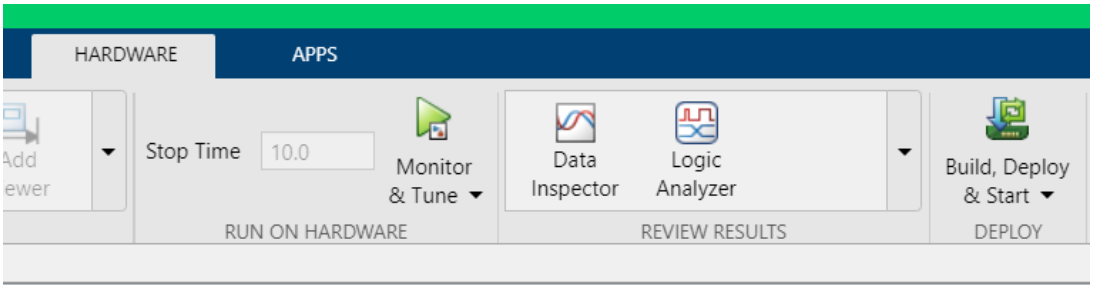


Obr. 25: Model Settings



Obr. 26: Hardware Implementation

Poté, co je správně nastavený Hardware board se změní horní lišta Simulinku. V kartě „HARDWARE“ se objeví tlačítko „Build, Deploy & Start“ popř. „Build“. Zmáčknutím tohoto tlačítka se program nahraje do EV3 Brick a lze ho spustit z EV3 Brick.



Obr. 27: Build, Deploy & Start

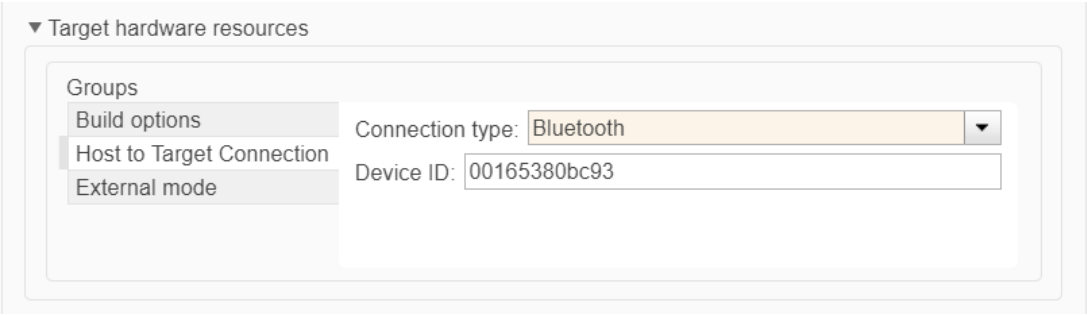
Při propojení EV3 Brick s prostředím Matlab a Simulink pomocí USB není možné používat „Monitor & Tune“ režim.

4.3 Propojení EV3 Brick pomocí Bluetooth

Propojení EV3 Brick s Matlab a Simulink pomocí Bluetooth je výhodnější než propojení pomocí USB kabelu jen díky tomu, že zde nepřekáží kabel a lze tedy realizovat nahrání programu i na delší vzdálenost. Ani v případě propojení EV3 Brick s prostředím Matlab a Simulink pomocí Bluetooth není možné použít „Monitor & Tune“ režim [25].

EV3 Brick má v sobě zabudované Bluetooth, které se aktivuje v „Setting“>“Bluetooth“ a zaškrtnutím políčka „Bluetooth“ a posléze „Visibility“. V levém horním rohu EV3 Brick se objeví symbol Bluetooth. Poté se EV3 Brick spáruje s počítačem nebo laptopem jako jiná zařízení využívající Bluetooth. Propojení EV3 Brick s počítačem je možné zkontrolovat v Matlabu v Command Window zadáním příkazu *„myev3 = legoev3('Bluetooth','comport')“.* Po zadání tohoto příkazu se v Command Window vypíšou informace o EV3 Brick jako v předešlém případě. Nicméně je zde důležité najít správný comport. Jaký comport je využíván, lze obvykle nalézt v „Device Manager“>“Ports“. Proto, aby tento příkaz fungoval, je třeba opět vymazat příkazem *„clear all“* Workspace Matlabu.

Nastavení Simulinku probíhá obdobně jako u propojení pomocí USB. Jediná změna nastane při volbě „Connection type“. Zde se zvolí možnost „Bluteooth“ a poté je nutné zadat „Device ID“ (viz. Obr. 28). „Device ID“ je napsáno opět v nastavení EV3 Brick v „Brick Info“ (viz. Obr. 22).



Obr. 28: Propojení EV3 Brick se Simulinkem pomocí Bluetooth

4.4 Propojení EV3 Brick pomocí Wi-Fi

Propojení EV3 Brick s Matlab a Simulink pomocí Wi-Fi je bezpochyby nejlepší volbou. Tento typ propojení je jediným, který dokáže používat tzv. „Monitor & Tune“ režim popsaný v kapitole 4.4.1 a zároveň je možné nahrát program do EV3 Brick na nejdelší vzdálenost.

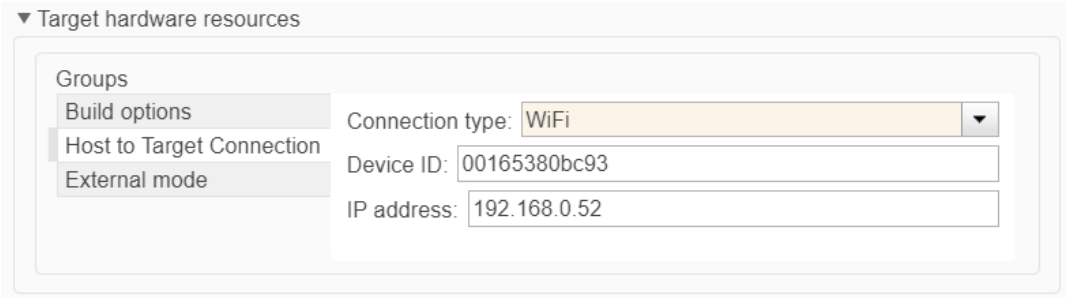
Nevýhodou tohoto propojení je, že spotřebovává nejvíce baterie oproti předchozím dvěma typům propojení. Nicméně rozdíl není tak markantní. Další nevýhodou nebo komplikací může být, že EV3 Brick v sobě nemá zabudovaný Wi-Fi přijímač/vysílač, tudíž je nutné ho dokoupit. Na „Help“ Matlabu a Simulinku je uveden jako kompatibilní Wi-Fi přijímač/vysílač NETGEAR WNA 1100 nebo Onkyo UWF-1. První z uvedených se již téměř neprodává a zároveň je poměrně veliký, nicméně je to jediný doporučený Wi-Fi přijímač/vysílač na oficiálních stránkách Lega. Onkyo UWF-1 lze sehnat v online obchodech, ačkoliv jeho cena je poměrně vysoká. Další možností, která je cenově nejvíce přijatelná a zároveň je Wi-Fi přijímač/vysílač nejmenší z uvedených, je Edimax EW-7811Un. Ačkoliv tento Wi-Fi přijímač/vysílač není doporučený k EV3 Brick, funguje bez jakýchkoliv problémů. Informace byly získány ze zdroje [26] [25].

Po zapojení Wi-Fi přijímače/vysílače do EV3 Brick je nutné v nastavení EV3 Brick kliknout na „WiFi“>“Connections“, poté vybrat Wi-Fi, ke které je připojen laptop nebo počítač, se kterým má EV3 Brick komunikovat, a zvolit „Connect“. Pokud je Wi-Fi zaheslovaná, je nutné zadat heslo na EV3 Brick. Nelze zadávat v EV3 Brick diakritická znaménka nad písmeny.

Propojení EV3 Brick s počítačem je možné zkontrolovat v Matlabu v Command Window zadáním příkazu *„myev3 = legoev3(‘wifi‘,‘ipaddress‘,‘id‘)“.* Po zadání tohoto příkazu se v Command Window vypíšou informace o EV3 Brick jako v předešlém případě. Nicméně je zde důležité najít správnou IP adresu, která je uvedena v nastavení EV3 Brick pod „Brick Info“ společně s Brick ID. Proto, aby tento příkaz fungoval, je třeba opět vymazat příkazem *„clear all“* Workspace Matlabu.

Nastavení Simulinku je obdobné jako v předešlých případech. Jediným rozdílem je,

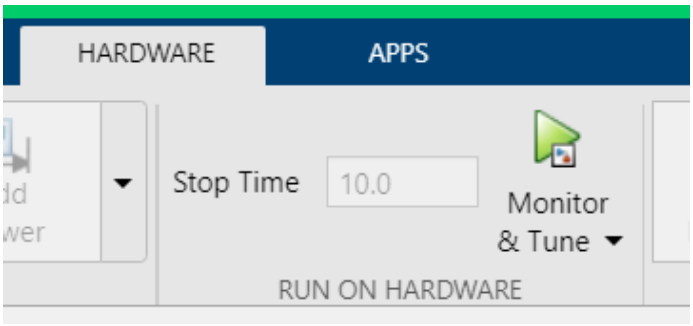
že v kolonce „Connection type“ je nutné zvolit typ „WiFi“ (viz Obr. 29). Posléze se objeví další dvě kolonky a to „Device ID“ a „IP address“, které je nutné správně vyplnit a na závěr potvrdit nastavení tlačítkem „Apply“.



Obr. 29: Propojení EV3 Brick se Simulinkem pomocí Wi-Fi

4.4.1 Monitor & Tune režim

Jak bylo uvedeno v kapitole 4.4, Monitor & Tune režim neboli „External mode“, lze používat pouze při propojení Simulinku s EV3 Brick pomocí Wi-Fi. Na rozdíl od klasického nahrání programu do EV3 Brick, tento režim dovoluje naladit hodnoty parametrů v průběhu simulace a monitorovat signály z algoritmu běžícího na hardwaru LEGO Mindstorms EV3. Díky tomu je možné získat výstupy z motorů a sensorů a nalézt optimální parametry simulace. Tlačítko pro spuštění režimu Monitor & Tune se nachází v kartě „HARDWARE“ na horní liště (viz. Obr. 30).



Obr. 30: Monitor & Tune režim

5 Experimentální ověření funkčnosti vytvořeného modelu

Po sestavení reálného modelu vozidla, sestavení programu v Simulinku a Matlabu a propojení EV3 Brick s řídicím prostředím zůstává jen spustit výsledný model v testovacím prostředí. Testovací prostředí bylo vytvořeno z krabic od bot, které představují zaparkované automobily.

5.1 Parametry modelu vozidla a výpočet minimální délky parkovacího místa

Ve výchozím skriptu Matlabu jsou zapsány základní parametry vozidla. Tyto jednotlivé hodnoty byly změřeny posuvným měřítkem nebo pravítkem po sestavení modelu vozidla.

Průměr kol d	0.0562	m
Rozvor kol e	0.176	m
Šířka vozidla w	0.165	m
Přesah přední a zadní části vozidla p	0.07	m
Úhel natočení předních kol β	36	°
Délka vozidla l	0.3160	m

Tab. 1: Parametry modelu vozidla

Tento skript dále vypočítává poloměr střední kružnice R, poloměr nejmenší kružnice R_i a poloměr největší kružnice R_e, které vozidlo při pohybu opisuje (viz. kapitola 1.1), a z těchto hodnot dopočítává minimální délku parkovacího místa L_{min}.

Poloměr střední kružnice R	0.2952	m
Poloměr nejmenší kružnice R_i	0.1545	m
Poloměr největší kružnice R_e	0.4032	m
Min. délka parkovacího místa L_{min}	0.4425	m

Tab. 2: Dopočítaná geometrie parkovacího manévru

5.2 Poznatky v průběhu testování modelu

V průběhu testování se ukázalo, že správná funkčnost servomotorů je závislá na tom, jaký je stav baterie. Je tedy důležité pro finální experimenty zkontrolovat, že baterie EV3 Brick je plně nabitá.

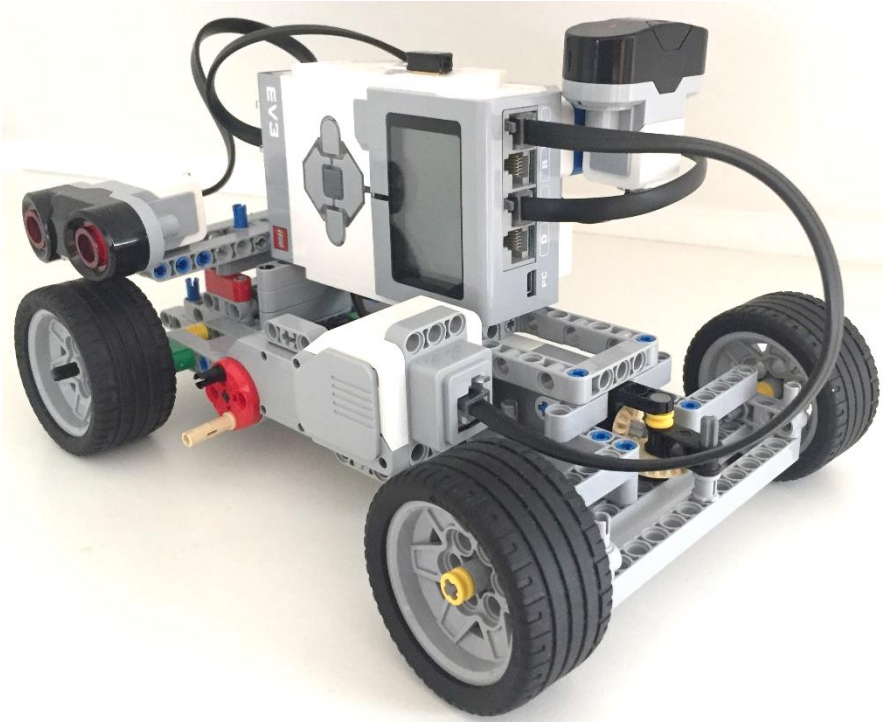
Zároveň bylo v průběhu testování zjištěno, že model vykazuje nepřesnosti na koberci, proto je pro finální experiment ideální, aby vozidlo jelo po hladké podlaze. Důvod, proč není koberec ideálním je, že hodnoty, které jsou do motorů posílány, jsou cíleně malé, aby vozidlo dosahovalo, co největších přesností, nicméně pasivní odpory při jízdě po koberci jsou příliš velké na to, aby vozidlo jelo plynule.

Hodnoty, které jsou posílány do motorů modelu vozidla byly postupně upravovány v „Monitor & Tune“ režimu popsaném v kapitole 4.4.1. Senzory a motory stavebnice LEGO Mindstorms EV3 dokážou posílat výstupní signál maximálně každou 0.01 s, tzv. *„sample time“.* Je tedy důležité, aby se model vozidla pohyboval pomalu a program vyhodnocoval výstupní signály včas. Jinými slovy, aby například vozidlo neprojelo okolo začátku parkovacího místa a nevyhodnotilo jeho detekci se zpožděním. Poté by poloha parkovacího místa byla nepřesná. V jiném případě by model mohl přejet jednotlivé polohy při parkovacím manévru a tím tvořit další nepřesnosti. Na druhou stranu, výkony motorů musí být dostatečně velké pro pohánění vozidla a natáčení předních kol. Ideální hodnota pro velký servomotor, který pohání vozidlo, byla okolo 20 až 30. Pro malý servomotor, který natáčí přední kola, byla nalezena ideální hodnota v rozmezí 5 až 10.

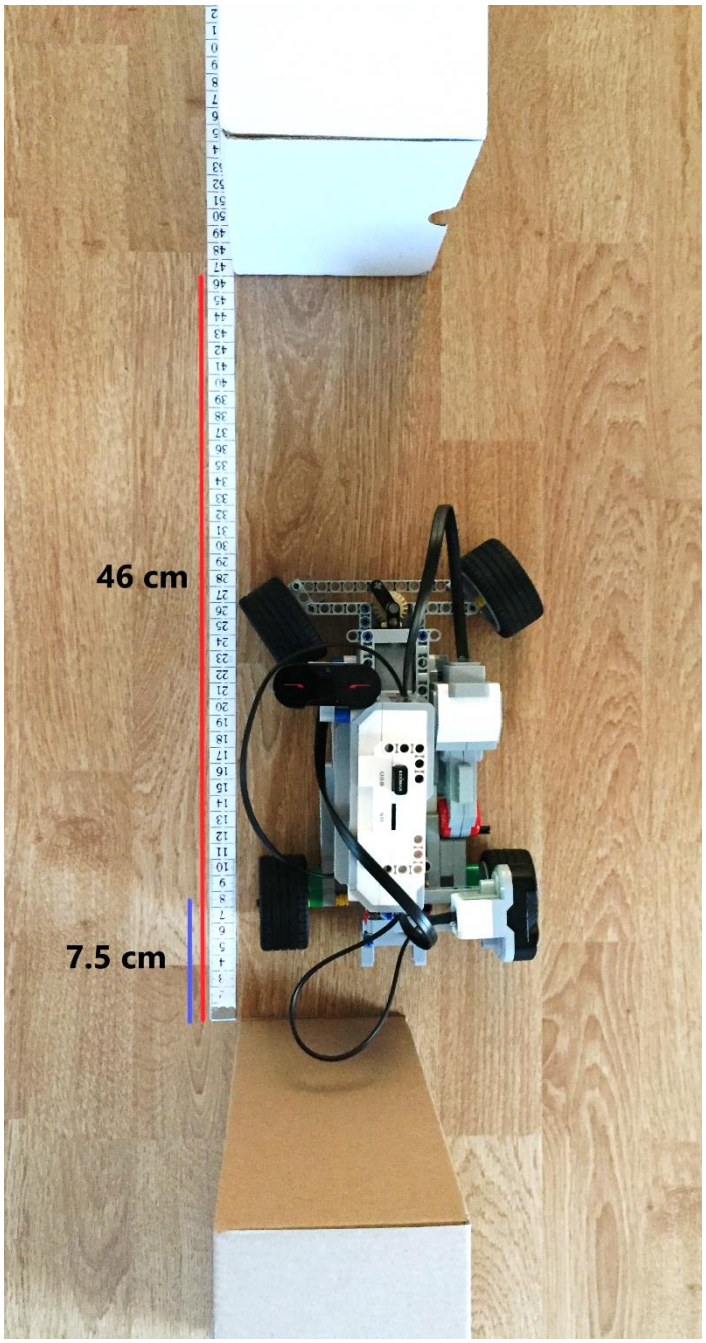
5.3 Finální experiment výukového modelu autonomně parkujícího vozidla

Finální model vozidla je na Obr. 31. Po nahrání finálního programu do EV3 Brick bylo vozidlo desetkrát spuštěno v testovacím prostředí. Výsledky byly pozitivní. Model vozidla opravdu dokázal relativně přesně detekovat parkovací místo a realizovat parkovací manévr. Parkovací místo bylo naměřeno na 46 cm (viz. Obr. 32), tedy o necelé 2 cm delší, než je minimální délka parkovacího místa (viz. Tab. 1). Z deseti pokusů byl pouze jeden, kdy vozidlo

nedetekovalo správně parkovací místo a jelo dál. Důvod mohl být, že vozidlo nejelo rovnoběžně s fiktivními vozidly, ale více šikmo a ultrazvukový senzor tedy detekoval i boky krabic a tím nenašel dostatečný prostor. Další nepřesnosti byly patrné například v tom, že vozidlo se lehce přetáčelo, nicméně to nebylo pravidlem. Ve výsledku všechny pokusy až na jeden zmíněný byly úspěšné a vozidlo našlo správné parkovací místo a zrealizovalo parkovací manévr bez kolize. Model vozidla ve všech devíti případech zaparkoval na stejném místě s odchylkou doprava/doleva nebo dopředu/dozadu okolo 3 *cm*. V příloze na CD jsou přiložená videa, jak některé pokusy probíhaly.



Obr. 31: Finální model vozidla

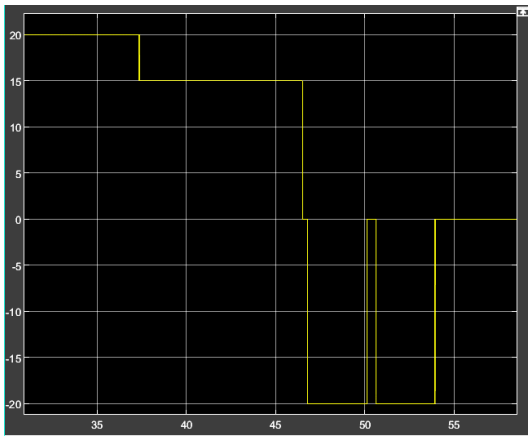


Obr. 32: Výsledek experimentu

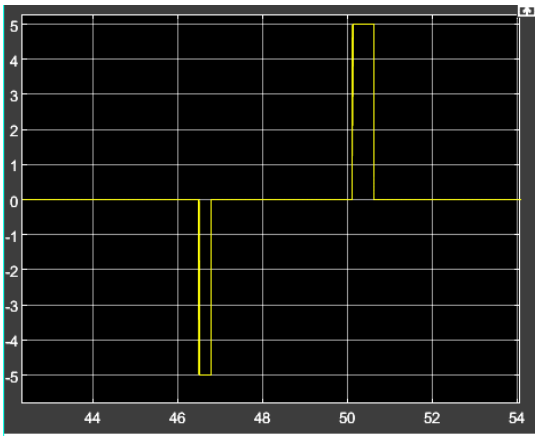
5.4 Výstupy z „Monitor & Tune“ režim

Zde jsou některé grafy získané ze „Scopes“ Simulinku, které popisují a ukazují jaké signály jsou do motorů posílány a zase naopak, jaké signály posílají motory nazpět. Všechny grafy jsou závislosti na čase ve vteřinách.

Na Obr. 33 a Obr. 34 je patrné, jaké hodnoty, v jaký čas, byly vyslány do velkého a malého motoru.

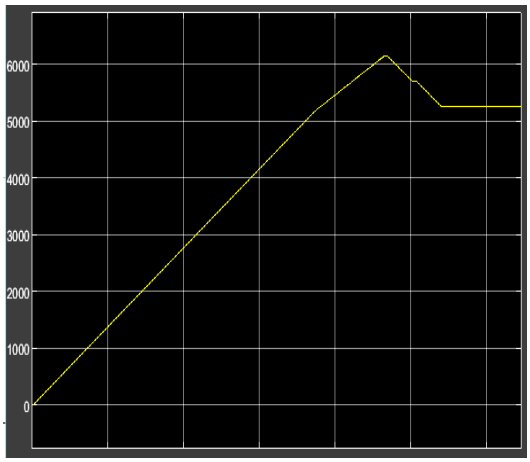


Obr. 33: Velký motor-vstupní signál

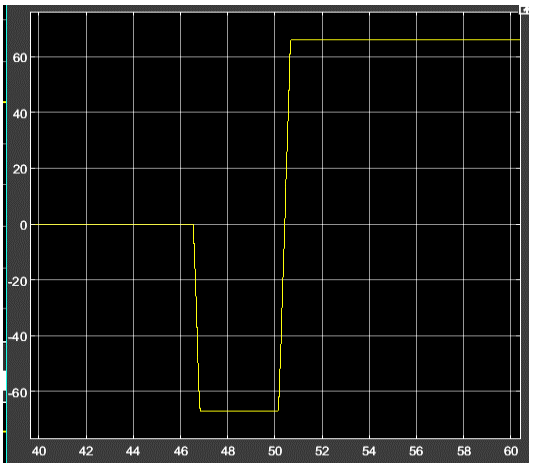


Obr. 34: Malý motor-vstupní signál

Další dva grafy popisují úhel pootočení servomotorů ve stupních v čase. První graf Obr. 35 ukazuje, jak se vozidlo pohybuje dopředu a hledá parkovací místo, tedy natočení motoru stále roste, poté začne couvat a hodnota natočení motoru klesá. Druhý graf Obr. 36 mapuje, jak se natáčejí přední kola nejprve na jednu stranu a poté na druhou.

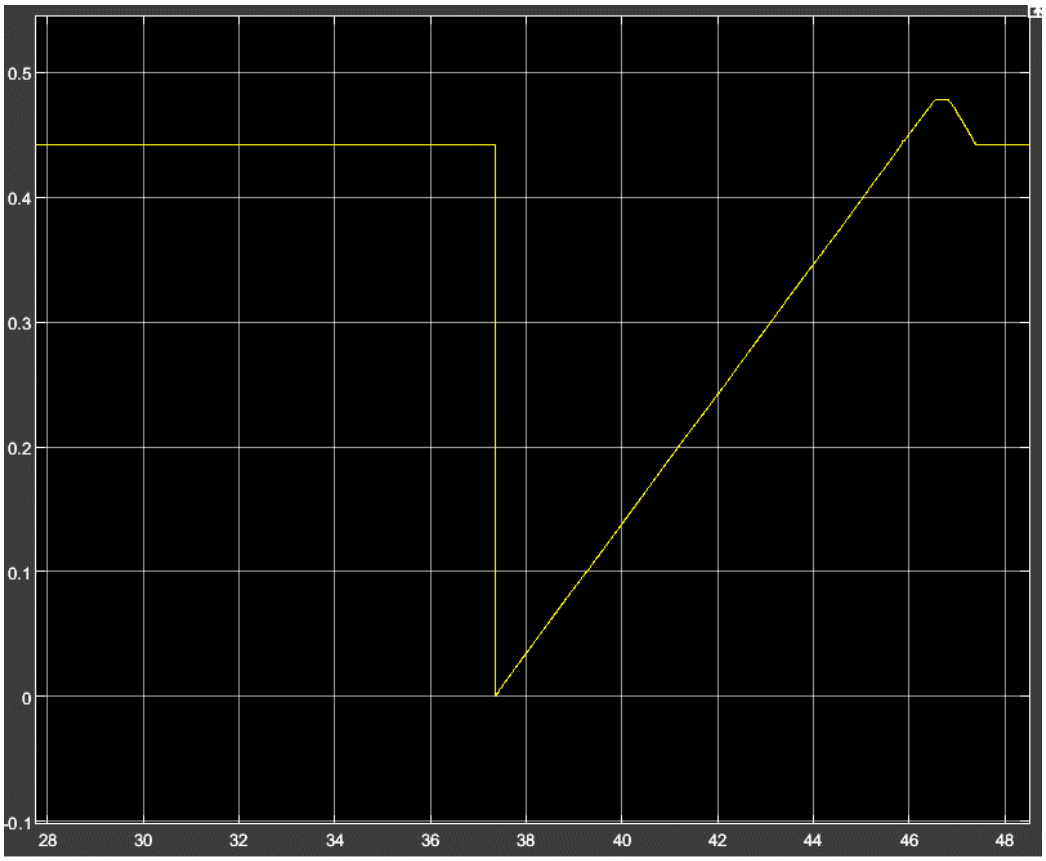


Obr. 35: Velký motor-výstupní signál



Obr. 36: Malý motor-výstupní signál

Poslední graf Obr. 37 mapuje to, jak vozidlo detekuje a měří parkovací místo. Graf je závislost délky parkovacího místa v metrech na čase. Ve chvíli, kdy vozidlo detekuje parkovací místo, začne se měřit délka místa od nuly. Ve chvíli, kdy je místo dostatečně dlouhé, pro tento konkrétní případ zhruba v 46 *s*, začne výpočet pro samotný parkovací manévr.



Obr. 37: Detekce parkovacího místa

Závěr

Cílem této práce bylo sestavit model vozidla, který nastíní problematiku autonomně parkujících vozidel, a zároveň může být nástrojem pro výuku. Model vozidla byl poskládán ze stavebnice LEGO Mindstorms EV3 a řízení bylo vytvořeno v prostředí Matlab a Simulink.

První kapitola popisuje kinematický model vozidla na základě, kterého jsou tvořeny rovnice a matematické vzorce důležité pro chování takového modelu. Jedním z bodů této kapitoly je geometrie samotného vozidla a vytvoření ideální trajektorie parkovacího manévru pro podélné parkování. Samotný parkovací manévr je realizován couváním po dvou opačných obloucích.

V druhé kapitole je ukázka sestaveného modelu vozidla ze stavebnice LEGO Mindstorms EV3. Tato stavebnice se ukázala jako velmi komplexní díky jejímu velkému množství dílků, senzorů a motorů. Je tedy možné z této stavebnice rychle, bez náročného vytváření jednotlivých dílů, poskládat i složitější mechanické modely. Zároveň je jednoduché tyto modely libovolně upravovat, a tedy rychle získat ideální model pro konkrétní experiment. V průběhu testování byl model vozidla upravován. Byl zde přidán diferenciál na zadní nápravu a byla upravena i přední náprava z hřebenového převodu na převod s ozubenými kolečky. Obě tyto úpravy významně přispěly k funkčnosti modelu, a to zejména proto, že diferenciál na zadní nápravě odstranil smýkání zadních kol při zatáčení a převod na přední nápravě zajistil přesnější natáčení předních kol.

V další části je stručně vysvětlena práce s Matlabem a Simulinkem, jakožto silných výpočetních a simulačních programů. Jsou zde popsány bloky Simulinku, které jsou přímo určeny pro programování hardwaru EV3 Brick. Tyto bloky je nutné stáhnout do knihovny Simulinku a jsou prakticky stěžejní při tvorbě programu. Zároveň v dalších bodech této kapitoly je lehce vysvětleno, jak funguje sestavování bloků v Simulinku, a jak propojit skript v Matlabu s programem v Simulinku.

Dalším bodem práce je propojení EV3 Brick s prostředím Matlab a Simulink. Je zde velice názorně ukázáno, jak postupovat. V této kapitole je detailně popsáno propojení EV3

Brick s prostředím Matlab a Simulink pomocí Wi-Fi. Propojení pomocí Wi-Fi je výhodným, zejména proto, že lze používat tzv. „External mode“ neboli „Monitor & Tune“ režim, který zajišťuje to, že Simulink spustí program na EV3 Brick a získává výstupy z motorů a senzorů. Zároveň je možné v průběhu simulace pozměňovat jednotlivé parametry. Výhoda tohoto propojení je tedy jasná, nicméně samotná EV3 Brick nedisponuje Wi-Fi přijímačem/vysílačem, proto je ho nutné dokoupit.

V poslední kapitole práce je ukázka, jak dopadl finální experiment autonomně parkujícího vozidla sestaveného ze stavebnice LEGO Mindstorms EV3 řízeného z prostředí Matlab a Simulink. Výsledky experimentu byly pozitivní. Z deseti pokusů bylo devět úspěšných. Vozidlo podélně zaparkovalo mezi krabice, které představovaly zaparkovaná vozidla, bez kolize. Při chybném pokusu vozidlo nedetekovalo parkovací místo a jelo dále. Důvodem mohlo být, že vozidlo nejelo rovnoběžně s krabicemi, a tedy ultrazvukový senzor detekoval i boky krabic, a proto zde nevyhodnotil dostatečně velký prostor. Zároveň, v druhé části této kapitoly, jsou popsány výstupní grafy ze Simulinku. Na grafech jsou vidět hodnoty, které byly posílány do motorů, a úhly natočení jednotlivých motorů v průběhu parkovacího manévru.

Obecně by tato práce měla ukázat možné využití stavebnice LEGO Mindstorms EV3 v kombinaci s řízením z prostředí Matlab a Simulink při tvorbě některých mechatronických modelů a výukových experimentů. Zároveň, díky svému detailnímu popisu, může tato práce sloužit, jako návod při tvorbě právě takových experimentů. Dá se říct, že velice hravou formou, lze sestavit mechanické modely a naprogramovat je v relativně krátkém čase, a přitom silně rozšířit svoje znalosti o dané problematice.

Reference

[1] ZIPS, Patrik. A Fast Motion Planning Algorithm for Car Parking Based on Static, *in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.* 2013, pp. 2392-2397.

[2] ZHOU, Wenyi. An Improved Approach for Automatic Parallel Parking in Narrow, *Electronic Theses and Dissertations.* 2015.

[3] CHENG, Kunpeng. Planning and Control for a Fully-automatic Parallel Parking Assist, *Intelligent Vehicles Symposium.* 2013, pp. 1440-1445.

[4] ZHENJI, Lv. A path-planning algorithm for parallel automatic parking, *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control.* 2013, pp. 474-478.

[5] ZIPS, Patrik. Optimisation based path planning for car parking in, *Robotics and Autonomous Systems.* 2016, sv. 79, pp. 1-11.

[6] SEKERKA, Michal. *Systém pro plánování složitějších parkovacích manévrů,* Praha, 2018, Diplomová práce. České vysoké učení technické.

[7] WANG, Lixue. Path Planning Algorithm for Automatic Parallel Parking from Arbitrary Initial Angle*, 2017 10th International Symposium on Computational Intelligence and Design (ISCID).* 2017, doi: 10.1109/ISCID.2017.115, pp. 55-58.

[8] VOROBIEVA, Hélene. Geometric Path Planning for Automatic Parallel Parking in Tiny Spots, *IFAC Proceedings Volumes.* 2012, sv. 24, č. 45, pp. 36-42.

[9] CHOI, Sungwoo. Easy Path Planning and Robust Control for, *IFAC World Congress.* 2011, pp. 656-661.

[10] KOCIÁN, Vladislav. *Asistent pro nalezení optimální trajektorie,* Praha, 2018. Diplomová práce. České vysoké učení technické.

[11] SCHEUER, A. Continuous-Curvature Path Planning for Car-Like Vehicles, v *Conf. on Intelligent Robots and Systems* , Grenoble, 1997.

[12] VRBSKÝ, Ladislav. *Experimentální model vozidla s asistentem pro couvání s přívěsem,* Praha, 2016. Bakalářská práce. České vysoké učení technické.

[13] LEGO® MINDSTORMS® Education EV3 Homeschool Combo Pack. *LEGO* [online]. 2021. Available: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-homeschool-combo-pack/5003480. [Přístup získán 4. 7. 2021].

[14] Inteligentní kostka EV3. *LEGO* [online]. 2021. Available: https://www.lego.com/cs-cz/product/ev3-intelligent-brick-45500. [Přístup získán 4. 7. 2021].

[15] Download the EV3 Lab Software v. 1.4.5, *LEGO* [online]. 2021. Available: https://education.lego.com/en-us/downloads/retiredproducts/mindstorms-ev3-lab/software. [Přístup získán 4. 7. 2021].

[16] Data Type Conversion. *MahWorks* [Online]. Available: https://uk.mathworks.com/help/simulink/slref/datatypeconversion.html?searchHighlight=Data%20Type%20Conversion%20&s_tid=srchtitle. [Přístup získán 14. 7. 2021].

[17] Double. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/matlab/ref/double.html. [Přístup získán 14. 7. 2021].

[18] Numeric Types. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/matlab/numeric-types.html. [Přístup získán 14. 7. 2021].

[19] Scope. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/simulink/slref/scope.html?searchHighlight=Scope&s_tid=srchtitle. [Přístup získán 14. 7. 2021].

[20] Gain. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/simulink/slref/gain.html?searchHighlight=Gain&s_tid=srchtitle. [Přístup získán 15. 7. 2021].

[21] Switch. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/simulink/slref/switch.html?searchHighlight=Switch&s_tid=srchtitle. [Přístup získán 15. 7. 2021].

[22] „MATLAB Function," MathWorks, [Online]. Available: https://uk.mathworks.com/help/simulink/slref/matlabfunction.html?searchHighlight=MATLAB%20Function%20&s_tid=srchtitle. [Přístup získán 15. 7. 2021].

[23] If, elseif, else. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/matlab/ref/if.html?searchHighlight=if%2C%20elseif&s_tid=srchtitle. [Přístup získán 15. 7. 2021].

[24] Data Store Memory. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/simulink/slref/datastorememory.html?searchHighlight=Data%20Store%20&s_tid=srchtitle. [Přístup získán 10. 7. 2021].

[25] Set Up the EV3 Hardware. *MathWorks* [Online]. Available: https://uk.mathworks.com/help/supportpkg/legomindstormsev3/ug/set-up-networking-for-ev3-hardware.html?searchHighlight=Set%20Up%20the%20EV3%20Hardware&s_tid=srchtitle. [Přístup získán 15. 6. 2021].

[26] Connecting your LEGO® MINDSTORMS® EV3 to Wi-Fi. *LEGO* [Online]. Available: https://www.lego.com/en-gb/service/help/MINDSTORMS_EV3/connecting-your-lego-mindstorms-ev3-to-wi-fi-kA009000001dcjmCAA. [Přístup získán 10. 6. 2021].

[27] LEGO® MINDSTORMS® Education EV3 Large Servo Motor. *LEGO* [Online]. Available: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-large-servo-motor/45502. [Přístup získán 10. 8. 2021].

Přílohy

- bakalářská práce v elektronické formě (.pdf)
- simulační model v prostředí Matlab/Simulink
- video realizace parkovacího manévru