VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY
INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

HIL SIMULACE MANIPULÁTORŮ NEBO STROJE
HIL SIMULATION OF MANIPULATORS OR MACHINES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Dočekal**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Daniel Zuth, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**Bc. Martin Dočekal**
Studijní program:	Strojní inženýrství
Studijní obor:	Výrobní stroje, systémy a roboty
Vedoucí práce:	**Ing. Daniel Zuth, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

HIL simulace manipulátorů nebo stroje

Stručná charakteristika problematiky úkolu:

Práce se bude zabývat HIL simulací (Hardware–in–the–loop simulation), kdy bude ve virtuálním SW vytvořen manipulator/stroj, který bude řízen reálným HW. Propojení reálných vstupů a výstupů s virtuálním strojem bude řešeno pomocí měřicí karty. Důraz bude kladen na vytvoření propojení, které bude v budoucnu možné dále modifikovat. Navržené rozhraní bude sloužit k výukovým účelům.

Cíle diplomové práce:

Rešerše v oblasti simulačního SW.
Vybrat a popsat vhodný SW.
Vytvoření virtuálního modelu.
Popis propojení simulačního SW a DAQ SW.
Popis navrženého řešení včetně dokumentace.
Vytvoření vzorové úlohy.
Vlastní závěr a doporučení pro praxi.

Seznam doporučené literatury:

NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. Praha: BEN - technická literatura, 2005. ISBN 8073001411.

ROS.org | Powering the world's robots. ROS.org | Powering the world's robots [online]. Dostupné z: https://www.ros.org/

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Diplomová práce se zabývá simulací HIL (z anglického - hardware in the loop). Práce obsahuje manipulátor vytvořený ve virtuálním softwarovém prostředí V-REP. Propojení reálných vstupů a virtuálních výstupů stroje je realizováno pomocí mikrokontroleru Arduino UNO. První úloha se zabývá ovládání manipulátoru pomocí joysticku PS2. U druhého úkolu se jedná o samostatné řízení robota pomocí mikrokotroleru Arduino UNO. Výsledné propojení je možné v budoucnu dále modifikovat a navržené rozhraní dále upravovat. Práce bude využita k výukovým účelům.

ABSTRACT

The diploma thesis deals with HIL simulation (hardware in the loop). The thesis contains a manipulator created in the virtual software V-REP. The connection of real inputs and virtual outputs of the machine is realized by the microcontroller Arduino UNO. The first task deals with the control of the manipulator using the joystick PS2. The second task is a separate control of the robot using an microcontroller Arduino UNO. The resulting connection can be modified in the furher and the interface modified. The work will be used for educational purposes.

KLÍČOVÁ SLOVA

HIL, Hardware ve smyčce, V-REP, Arduino, Matlab, simulační software.

KEYWORDS

HIL, Hardware in the loop, V-REP, Arduino, Matlab, simulation software.

BIBLIOGRAFICKÁ CITACE

DOČEKAL, M. *HIL simulace manipulátorů nebo stroje,* Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2021, 57 s., Vedoucí diplomové práce Ing. Daniel Zuth, Ph.D.

PODĚKOVÁNÍ

Chtěl bych poděkovat svému vedoucímu Ing. Danielovi Zuthovi, Ph,D. za odborné vedení práce, čas strávený konzultacemi a cenné rady, které mi pomohl při zpracování této práce a zároveň bych chtěl poděkovat rodině za podporu.

ČESTNÉ PROHLÁŠ ENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Daniela Zutha,Ph,D. a s použitím literatury uvedené v seznamu.

V Brně dne 21. 5. 2021 ……

Bc. Dočekal Martin

OBSAH

1 ÚVOD 15

2 VYMEZENÍ CÍLŮ A LIMITŮ PRÁCE 16

3 HISTORIE SIMULACE 17

4 SIMULAČNÍ SOFTWARE 21
4.1 V-REP 21
4.1.1 Popis scény V-REP 22
4.2 Webots 24
4.3 Gazebo 25
4.4 Breve 26
4.5 Volba softwaru 27

5 SIMULACE VE SMYČCE 29
5.1 MIL - Model in the loop 29
5.2 SIL - Software in the loop 29
5.3 PIL - Processor in the loop 30
5.4 HIL - Hardware in the loop 31

6 HARDWARE IN THE LOOP 33
6.1 Současný stav metod HIL 33
6.2 Přehled metod 34
6.2.1 HIL simulátory základních jednoduchých a pomalých systémů 34
6.2.2 HIL simulátory středně náročných a rychlých systémů 34
6.2.3 HIL simulátory náročných a extrémně rychlých systémů 34
6.2.4 HIL simulátory komplexních systémů 35

7 PRAKTICKÁ ČÁST 37
7.1 Arduino 37
7.1.1 Arduino UNO 37
7.1.2 Joystick PS2 39
7.1.3 Nepájivé kontaktní pole 830 pinů 39
7.1.4 Ostatní komponenty 39
7.2 Matlab 40
7.3 V-REP 40
7.4 Komunikace 41
7.4.1 Komunikace přes sériovou komunikaci 41
7.4.2 Komunikace Matlabu s V-REP 42
7.5 Vytvoření komunikace 44
7.6 ÚLOHA 1 - Ovládání robota pomocí joysticku PS2 47
7.7 ÚLOHA 2 – Hardware in the loop 49

8 ZÁVĚR 51

9 ZDROJE 53

10 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK 55
10.1 Seznam tabulek 56
10.2 Seznam obrázků 56

1 ÚVOD

V současnosti se výroba a výrobní procesy nachází ve čtvrté generaci průmyslové revoluce neboli v Průmyslu 4.0 (Industry 4.0), která souvisí s rostoucím využíváním automatizace, digitalizace a umělé inteligence. Umožňuje flexibilitu výroby, zároveň produkovat unikátní díly v malých sériích a vychází vstříc specifickým požadavkům koncových zákazníků. Při implementaci metod a nástrojů Průmyslu 4.0 ve firmách by mělo dojít k úsporám času a nákladů. V budoucnosti budou robotické systémy nahrazovat lidskou sílu u výrobní linky. Tím budou vznikat nová pracovní místa, která budou vyžadovat vyšší kvalifikaci zaměstnanců.

Hlavním cílem diplomové práce je HIL simulace (Hardware in the loop simulation), kde bude ve virtuálním softwaru vytvořen manipulátor/stroj, který je řízen reálným hardwarem. Jedná se o propojení reálných vstupů a výstupů s virtuálním strojem pomocí měřící karty. Pro propojení byla zvolena měřící karta typu mikrokontroler Arduino UNO. Výsledné propojení je možné v budoucnu dále upravovat a implementovat na reálné případy. Dále navržené prostředí a propojení bude v budoucnu využito pro výukové účely.

Tato diplomová práce se zabývá jednou z nich, a to testování pomocí simulace HIL (z anglického - hardware in the loop). Práce obsahuje manipulátor vytvořený ve virtuálním softwarovém prostředí, který pracuje a je řízen reálným hardwarem. Propojení reálných vstupů a virtuálních výstupů stroje je realizováno pomocí měřící karty, kterou zastupuje mikrokontroler Arduino UNO. Výsledné propojení je možné v budoucnu dále upravovat a navržené rozhraní bude využito k výukovým účelům.

Teoretická část se zabývá rešerší v oblasti simulačního softwaru, kde je nejprve popsána historie simulace a dále simulační softwary. Další podkapitola nabízí shrnutí softwarů a výběr toho nejvhodnějšího pro moji diplomovou práci. Teoretická část dále obsahuje popis simulací ve smyčce i jejich rozdíly.

Praktická část zahrnuje popis vybraného hardwaru a všech použitých programů při simulaci. V další části je popsána komunikace mezi jednotlivými programy. Na závěr jsou popsány dvě úlohy. První se zabývá ovládáním manipulátoru pomocí joysticku ps2. Druhá úloha představuje HIL simulaci. Hardware UNO je v nekonečné smyčce s virtuální prostředím V-REP.

2 VYMEZENÍ CÍLŮ A LIMITŮ PRÁCE

VYMEZENÍ CÍLŮ PRÁCE

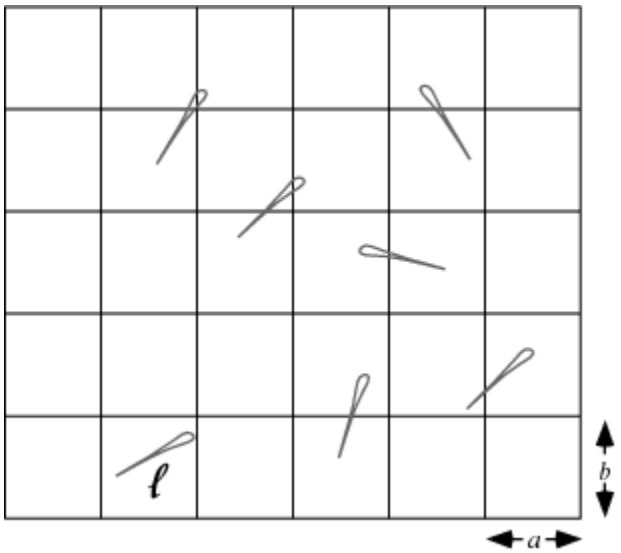
- Rešerše simulačního softwaru.
- Výběr a popis vhodného softwaru.
- Popis propojení simulačního SW a DAQ SW.
- Popis navrženého řešení včetně dokumentace.
- Vytvoření virtuálního modelu a vzorové úlohy na ovládání a řízení manipulátoru.

VYMEZENÍ LIMITŮ PRÁCE

- Rešerše bude obsahovat programy, které jsou licenčně dostupné.
- Pro vytvoření virtuálního modelu lze využít knihovny programu.
- Reálný hardware bude zastupovat mikrokontroler Arduino UNO.

3 HISTORIE SIMULACE

Historie a vývoj simulace se dá rozdělit do několika odvětví jako např. analýza, školení atd. Simulace vznikla ještě dávno před sestrojením prvního počítače. Jednalo se o metodu Monte Carlo, která vznikla Buffonovým needle experiment ("jehlovým experimentem") už v roce 1977, jenž vymyslel francouzský matematik Georges Louis Leclerc de Buffon. Samotná úloha zní takto: *„Na podlaze je velký list papíru, který je rozdělený rovnoběžnými linkami. Vzdálenost mezi všemi linkami je stejná. Na tento papír se libovolným způsobem hází jehla, jejíž délka je rovna vzdálenosti mezi linkami. Jaká je pravděpodobnost, že jehla po dopadu bude ležet tak, že protne některou z linek."* [1]



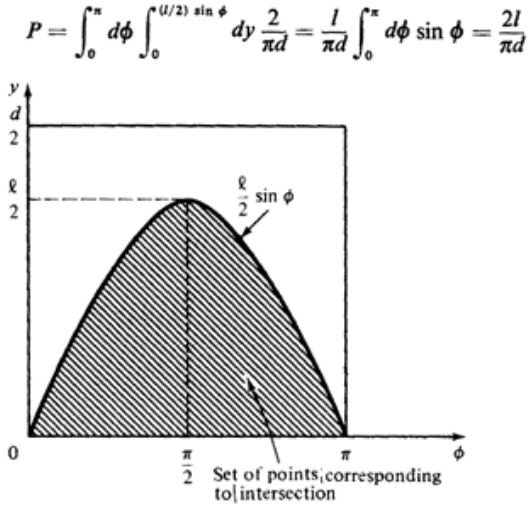
Obr. 1) Jehlový experiment [2]

„Hodnota této pravděpodobnosti je 2/pi. Pomocí takového experimentu je tedy možné zjistit přibližnou hodnotu pi: hod jehlou se bude mnohokrát opakovat a bude se zapisovat, v jakém poměru z celkového počtu hodů jehla protne linku." [1]

Metoda Monte Carlo

Jedná se o třídu algoritmů sloužící pro simulaci systémů. Mezi jeho typické využití patří výpočet integrálů. Především pak vícerozměrných, a to z toho důvodu, že běžné metody nejsou až tak efektivní. Dále lze pomocí této metody simulovat různé experimenty, počítat určité integrály nebo řešit diferenciální rovnice. [3] [4]

Základní myšlenka spočívá v tom, že chceme určit střední hodnotu náhodného děje po dostatečném množství opakování. Na základě simulace určíme průměr a směrodatnou odchylku. [3] [4]



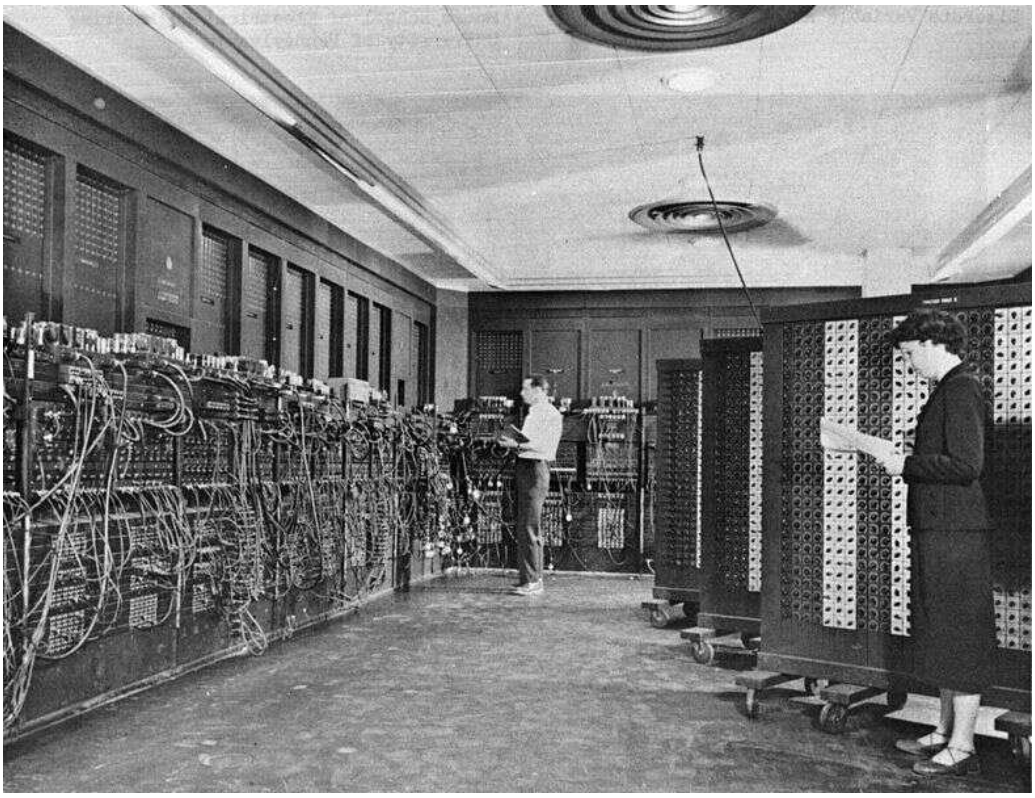
Obr. 2) Jehlový experiment [1]

Buffonovo řešení obsahovalo chybu, kterou v roce 1812 opravil francouzský matematik a fyzik Piere Simon de Laplace, proto se od tohoto roku používá název Buffon--Laplace needle problem (jehlový problém). [5]

Asi o sto let později sehrál velice důležitou a překvapivou roli v oblasti simulace vystudovaný matematik a chemik Williom Sealy Gosset, jenž se stal sládkem ve věku 23 let společně s Arthurem Guinessem. V roce 1988 Guiness umožnil Gossetovi publikovat významné statistické výsledky, a to za předpokladu, že nebude mít žádná vlastnická práva. Proto byla v roce 1908 data publikována pod pseudonymem „Student" a formulovala tzv. Studentovu t-distribuci. [5]

Protože měl Gosset neúplné analytické výsledky, použil surovou formu manuální simulace k ověření své domněnky o přesné formě funkce hustoty pravděpodobnosti Studentova t-rozdělení. Tato inaugurační (zahajovací) aplikace simulace řízení procesu v oblasti průmyslu je pozoruhodným příkladem synergie experimentů založených na simulaci a analytických technik v systému. Jedná se o objev přesného řešení, což je typický problém pro oblast průmyslového inženýrství. [5]

V polovině 40. let 20. století připravily půdu pro rychlý růst oblasti simulace dva hlavní vývojové trendy. Konstrukce prvních univerzálních elektronických počítačů, jako je ENIAC a Goldsman, Nance a Wilson. [5] [6]



Obr. 3) ENIAC PC [6]

Práce Stanislava Ulama, Johna von Neumana a dalších, kteří vyžívali metodu Monte Carlo na elektronických počítačích k řešení určitých problémů při difuzi neutronů, které vyvstaly při konstrukci vodíkové bomby, které byly a stále jsou analyticky neřešitelné (Cooper 1988).

Ulam měl velice rád karetní hry, a proto se snažil najít jednoduchý způsob, jak odhadnout pravděpodobnost událostí při hře. To ho vedlo k myšlence, že přístup Monte Carlo může být efektivní i u problémů matematické fyziky. [5] [6]

Zvyšující se dostupnost univerzálních elektronických počítačů v 50. letech minulého století připravila půdu pro rychlé rozšíření simulačních technik a aplikací do jiných oborů. Keith Douglas Toucher byl profesor operačního výzkumu na univerzitě v Southamptonu, jenž byl známý svými příspěvky k počítačové simulaci. Vyvinul (GSP - General Simulation Program), první univerzální simulátor jako nástroj pro systematické budování simulace průmyslového závodu, který zohledňuje stav každého stroje, jako je např. zaneprázdněn, nečinný, nedostupný, chyba stroje. [5]

Stav a časy dalších operací stroje společně definují stav závodu. Mezi nejvýznamnější příspěvky Tochera k simulaci patří třífázová metoda provedení časování, první učebnice v oblasti simulace The Art of Simulation (1963), koláčový graf nebo diagram cyklů a aktivit.

ACD se stala jedním ze základních kamenů simulační výuky ve Velké Británii a jádrem pro výzkumné programy generátorů v 70. letech 20. století. Je velice pozoruhodné, že Tocher vymyslel a implementoval přístup ke kombinované simulaci (diskrétní události a kontinuální provádění modelu, dlouho před jeho objevením v simulačním jazyce US. [5]

V tomto období vstoupil do divize Advanced System Development společnosti IBM Geoffrey Gordon jako manažer vývoje simulace a v roce 1960-1961 představil General Purpose System Simulator, jenž byl později přejmenován na General Purpose Simulation System (GPSS). GPSS byl navržen proto, aby umožňoval rychlé simulační modelování komplexních systémů teleprocesingu, které zahrnuje např. řízení městského provozu, odposlech a přepojování telefonních hovorů, zpracování rezervací leteckých společností a provoz průmyslových společností. Gordon se obával, že programování by mohlo být pro inženýry příliš složité, a proto se zaměřil na používání blokových diagramů tak, jak je známe dnes. Převážně kvůli snadnému užívání se stal v brzké době nejpopulárnější výukový simulační jazyk v USA. Počáteční verze SIMSCRIPT byla v roce 1963 určena pro odborníky, jež využívali sadu šablon pro definici modelů a generování zpráv. Druhá generace SIMSCRIPT 1.5 změnila překlad jazyku a sloužila jako generátor konceptů i nápadů pro SIMSCRIPT 2, což byl v té době nejambicióznější jazyk pro podnikový záměr. SIMCRIPT 2 byl publikován jako "vrstvený" jazyk definovaný na pěti úrovních. [5]

Kristen Nygaard a Ole Johan Dahl se společně se svými týmy podíleli v roce 1961 na vytvoření programovacího jazyku SIMULA známého také jako ALGOL60. Následný vývoj vytvořil pravděpodobně nejvlivnější programovací jazyk v historii výpočetní techniky. Tento programovací jazyk našel uplatnění převážně pro obecné účely, který byl v následujících letech rozšířen o příspěvek týkající se jakéhokoli simulačního jazyka nebo jakéhokoli aspektu simulačních aplikací. V dnešní době nese název WSC a jedná se o přední mezinárodní fórum pro šíření nových pokroků v oblasti simulace systémů. V dobách minulých do nich přispívali autoři jako například. R. W. Conway, W.L. Maxwell. Tito průlomoví autoři provedli rozdělení simulací do dvou obsáhlých kategorií: [5]

- konstrukce simulace
- použití simulace

Problémy konstrukce simulačního modelu obsahují:

- modulární návrh simulačních programů pro snadnou revizi
- správa paměti počítače
- řízení chyb vyplývajících z diskretizace všech spojitých veličin, které jsou vlastní digitální simulací
- návrh a implementace účinného mechanismu časového posunu
- správa souborů obsahující entity simulace

I přesto, že bylo mnoho z výše uvedených problémů do značné míry vyřešeno, návrh a implementace efektivního postupu s časovým předstihem pro zvládání určitých typů událostí je i v dnešní době stále aktivní oblast v oblasti výzkumu a vývoje. [5]

Mezi hlavní problémy pro simulaci patří strategický problém návrhu simulačního experimentu a následující taktické problémy týkající se provádění simulace uvedených v experimentálním návrhu: [5]

- problém spuštění - jedná se o stav, kdy je simulace v rovnováze (ustálený stav)
- odhad přesnosti (rozptylu) odhadů ustáleného stavu založených na simulaci
- provádění přesných srovnání alternativních simulací systému

4 SIMULAČNÍ SOFTWARE

V této kapitole je popsána rešerše v oblasti simulačního softwaru. Rešerše popisuje známé Open Source robotické simulátory jako V-REP, Webots, Gazebo. Simulační prostředí byla vybrána pro jejich široké využití. Poskytují velice realistické simulace, díky pokročilým fyzikálním enginům. Díky široké základně nabízejí mnoho databází a modelů. Dále je v kapitole popsán jednodušší robotický simulátor Breve. Simulátor umožňuje pouze omezené možnosti simulace vlivem menšího počtu modelů a funkcí.

4.1 V-REP

V-REP (Virtual Robot Experimentation Platform) je simulační software od firmy Coppelia Robotics, sloužící primárně pro výukové účely. Simulátor obsahuje širokou škálu možností a knihoven. Nabízí velké množství programování, a to jak samotného simulačního prostředí, tak jednotlivých objektů, které obsahuje scéna. Pro HIL simulaci je to jeden z nepřijatelnějších softwaru, jelikož je navržen tak, aby jeho zapojení do Hil simulace bylo co nejjednodušší. Coppelia Robotics nabízí velké množství návodů a příkladů dostupných na webových stránkách nebo fórech. [7]

Pro vytvoření prostředí a programování robotů, ale i pro samotnou simulaci, poskytuje V-REP interní API[1]. To je možné zavolat ze skriptů běžících přímo v programu, ale vzdálené API je možné volat i z jiného PC, které není přímo připojeno, ale má vzdáleného klienta. [7] [8]

V-REP je vhodné použít, pokud požadujeme: [8]

- rychlý vývoj algoritmů
- výrobní automatizační simulace
- rychlé vytváření prototypů a jejich následovné otestování
- robotické vzdělávání
- vzdálené monitorování a připojení

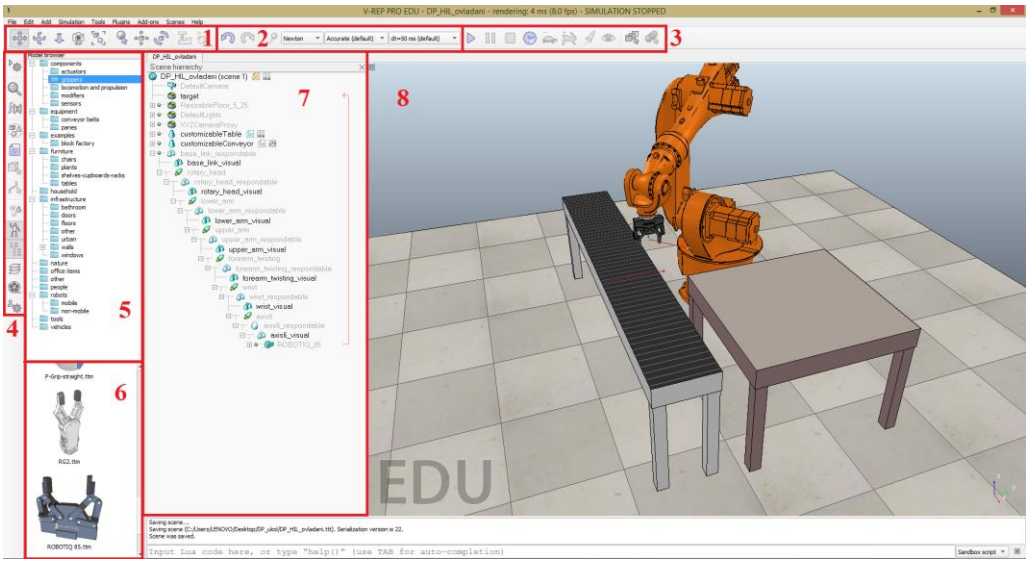
Simulační software se skládá ze tří fyzikálních enginů: [9]

- Bullet physics library
- Open Dynamics Engine
- Vortex Dynamics engine

Enginy zajišťují reálnost simulace z hlediska dynamické stránky. Vzhledem k jejich počtu může být v simulaci dosaženo vyšší přesnosti, rychlosti a podpory mnoha dalších funkcí. Zároveň umožnují simulovat vzájemné interakce mezi objekty, které se chovají jako interakce v reálném světě např. objekty se mohou srážet, odrážet nebo padat. Zároveň může robotická ruka uchopit předmět nebo dopravní pásy přemístit objekt atd. [10]

[1]API - Rozhraní pro programování aplikací (angl. Application Programming Interface)

4.1.1 Popis scény V-REP

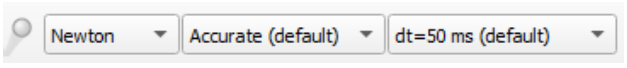


Obr. 4) Simulační prostředí V-REP

Číslo jedna představuje panel pro ovládání scény. Obsahuje ovládání kamery, manipulaci s objekty a ovládání kroku vpřed nebo zpět.

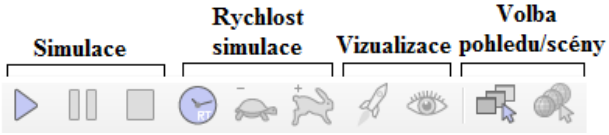
Obr. 5) Ovládací panel č. 1

Pod číslem dvě se nachází panel pro výběr a nastavení fyzikálního enginu a délky kroku simulace. Na výběr z enginů je: Bullet, ODE, Vortex a Newton. Třída přesnosti se volí z nabídky: Velmi přesné, přesné, rychlé, super rychlé a vlastní. Rychlost simulace je primárně nastavená na 50 ms. Lze ji měnit na hodnoty: 10, 25, 100 a 200 ms.



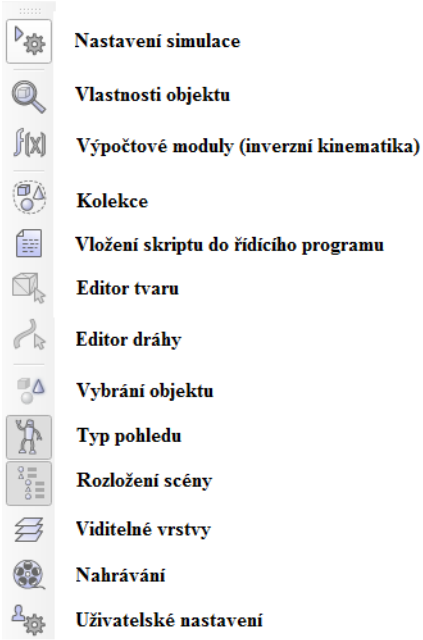
Obr. 6) Ovládací panel č. 2

Další okno obsahuje panel pro ovládání simulace. Lze simulaci pozastavit, upravit scénu a mimo jiné simulaci zrychlit nebo zpomalit.



Obr. 7) Ovládací panel č. 3

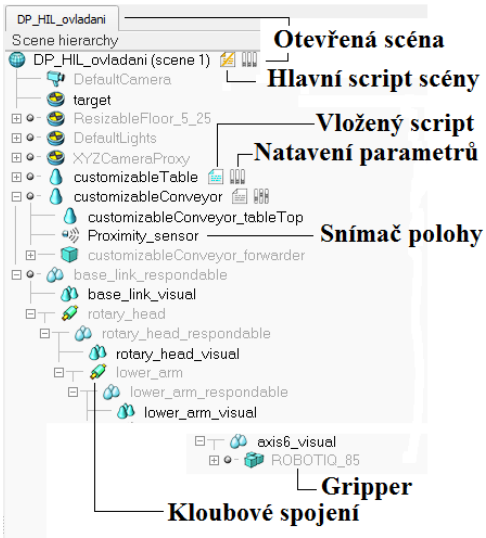
Svislý panel sloužící pro nastavení simulace. Nastavení mohou být, tak propracovaná, že simulace téměř napodobuje realitu. Během simulace je možné spustit nahrávání.



Obr. 8) Ovládací panel č. 4

Vedlejší panel pro výběr objektů z knihovny V-REP. Libovolně můžeme přidávat části nábytku, senzory, roboty atd. Následující panel slouží pro vizualizace vybraného objektu z knihovny

Pod číslem se ukrývá panel zobrazující rozložení scény. Zde může probíhat propojování objektů nebo jejich nastavení.



Obr. 9) Hierarchie scény

Největší plochu softwaru zabírá simulační scéna. Zde můžeme vkládat libovolné množství objektů z knihoven V-REP, nebo jakýkoliv objekt ve formátu .ttt.

4.2 Webots

Webots je open source multiplatformní simulační aplikace využívaná primárně pro simulaci robotů. Poskytuje kompletní vývojové prostředí pro modelování, programování i samotnou simulaci. Simulační software je primárně využíván pro výzkumné a vzdělávací projekty zahrnující mobilní roboty. [7] [11]

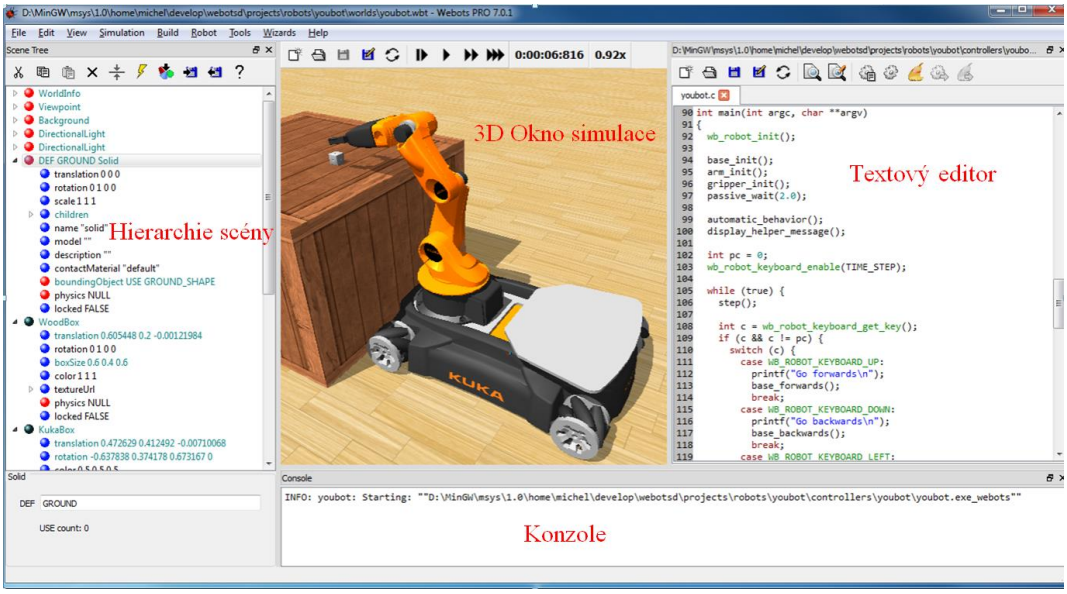
Webots umožňuje navrhnout robotizované sestavy s jedním nebo více roboty ve sdíleném prostředí. Řídící programy mohou být naprogramovány pomocí vývojového prostředí (v programovacích jazycích C, C++, Java, Python a další), nebo mohou být vytvořeny přes rozhraní ROS nebo pomocí Matlabu s Arduinem. [12]

Webots obsahuje širokou škálu knihoven, aktiv (robotů, senzorů, akčních členů, objektů a materiálů) a simulací včetně jedno stopových stolních robotů, průmyslových ramen, modulárních robotů, létajících robotů, pásových robotů atd. [11] [12]

Webots GUI se skládá ze čtyř hlavních oken, které jsou 3D okna (zobrazuje a umožňuje interakci s 3D simulací), hierarchie scény (zobrazování objektů ve vytvořeném světě), textový editor (úprava zdrojových kódů) a konzole (zobrazuje kompilační a řídící vstupy). [7]

Oblasti využití Webots: [11]

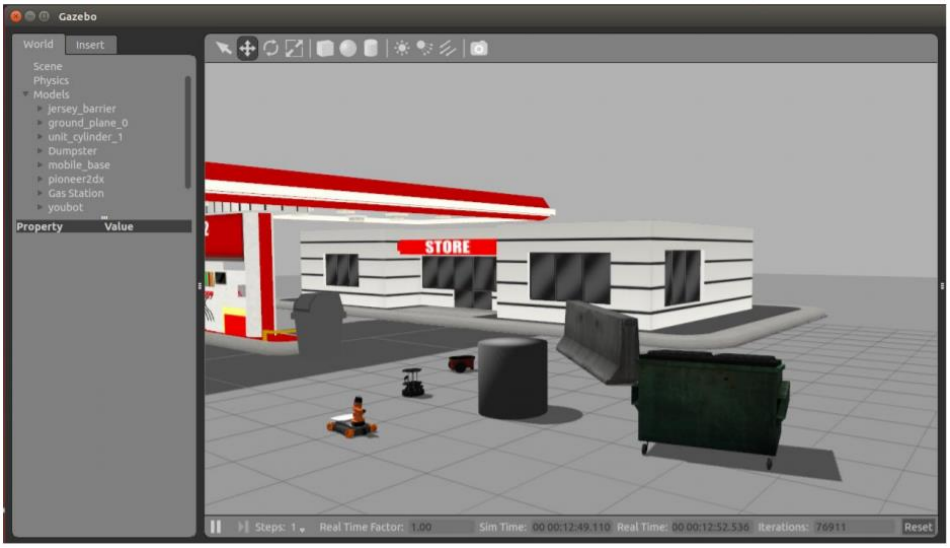
- vývoj prototypů v oblasti mobilních robotů (automobilový průmysl, letectví atd.)
- výzkum pohybu robotů
- výzkum adaptivního chování
- výuka robotiky
- soutěže v oblasti robotiky



Obr. 10) Simulační prostředí Webots [7]

4.3 Gazebo

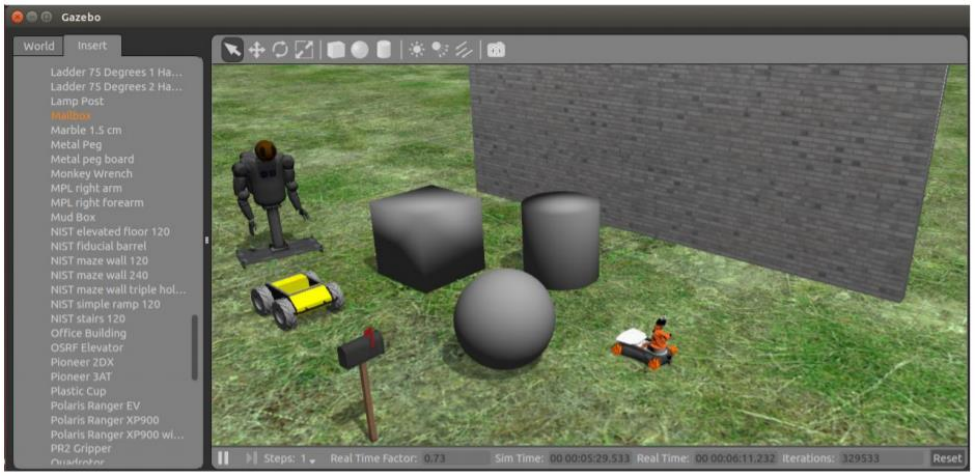
Jedná se o simulační nástroj pro simulaci robota a prostředí. Umožňuje rychlé testování algoritmů, vytvářet nové designy robotů, provádět regresivní testování pomocí realistických scénářů a simulace populace robotů ve složitých vnitřních i venkovních prostředích. [7]



Obr. 11) Simulační prostředí Gazebo [7]

Veškeré modely a objekty jsou popsány pomocí skriptů ve formátu SDF (Simulation Description Format). SDF, nebo-li XML formát popisuje objekty a prostředí robotických simulátorů. Tento formát je schopný popsat všechny aspekty robotů a statických i dynamických objektů, je zde popsáno také osvětlení, terén a fyzikální zákony. [7] [13]

Využívá distribuovanou architekturu s oddělenými knihovnami pro simulaci fyziky, vykreslování uživatelského rozhraní, komunikaci a generování senzorů. Obsahuje dva programy pro běh simulace, které jsou server gzserver (pro simulaci fyzikálních zákonů, vykreslování a senzory) a klient gzclient (poskytuje grafické rozhraní pro vizualizaci a interakci se simulací). [7] [13]



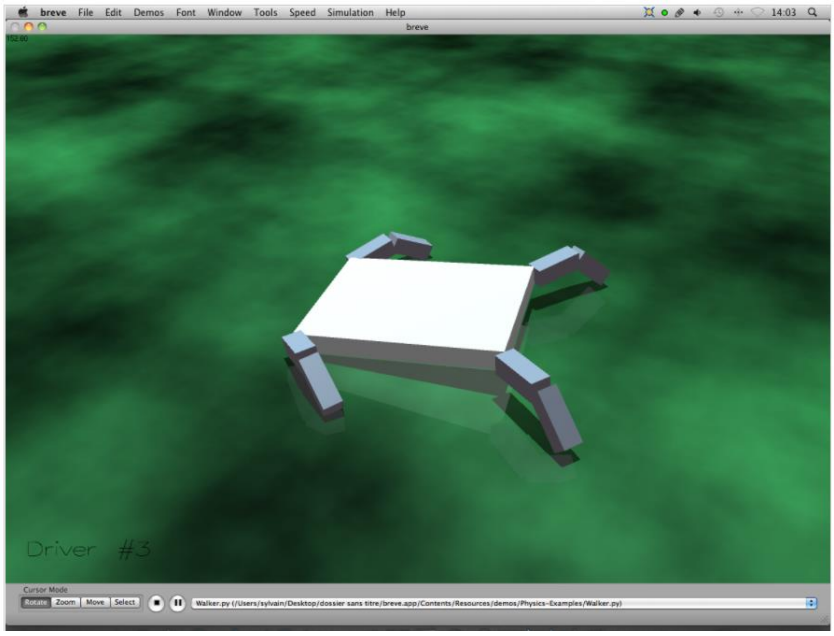
Obr. 12) Gazebo - knihovna objektů/modelů

Hlavní podporou je operační systém Linux, lze ho ovšem spustit i v operačních systémech Windows a Mac OS. Při propojení s ROSem dokáže efektivně komunikovat pomocí zpráv a podávat zpětné vazby jako reálný robot. [7] [13]

4.4 Breve

Simulátor Breve je popsán pomocí jazyka Python nebo také pomocí jazyka Steve, který je objektově orientovaný a využívá mnoho vlastností jazyků jako např. jazyk C. [14]

V Breve kromě simulace lze spustit simulační dema od nejjednodušších po složitější. Po spuštění dema se nám zobrazí jeho vlastní kód, který je možné použít do vlastní simulace. Lze přidávat různé simulace prostředí, např. přidat mlhu, nastavit poloviční nebo úplně vypnout gravitaci. Zároveň je možné ovládat rychlost simulace. [14] [15]



Obr. 13) Simulační prostředí Breve [16]

Ovládání robotického simulátoru je velmi jednoduché a hodí se pro jednoduché 3D simulace. Vlastní simulace se píše pomocí skriptovacího jazyka Python v samostatném okně. Pomocí programovacího jazyka lze do simulace přidávat různé objekty, které se mohou poté rozpohybovat. [14] [15]

4.5 Volba softwaru

Z nabízených simulačních softwarů jsem si vybral prostředí V-REP (Virtual Robot Experimentation Platform) od firmy Coppelia Robotics z licenčních důvodů. Jelikož je program dostupný pro kohokoliv. Vytvořenou úlohu lze přenášet mezi zařízeními a dále modifikovat.

Tab 1) Porovnání softwarů

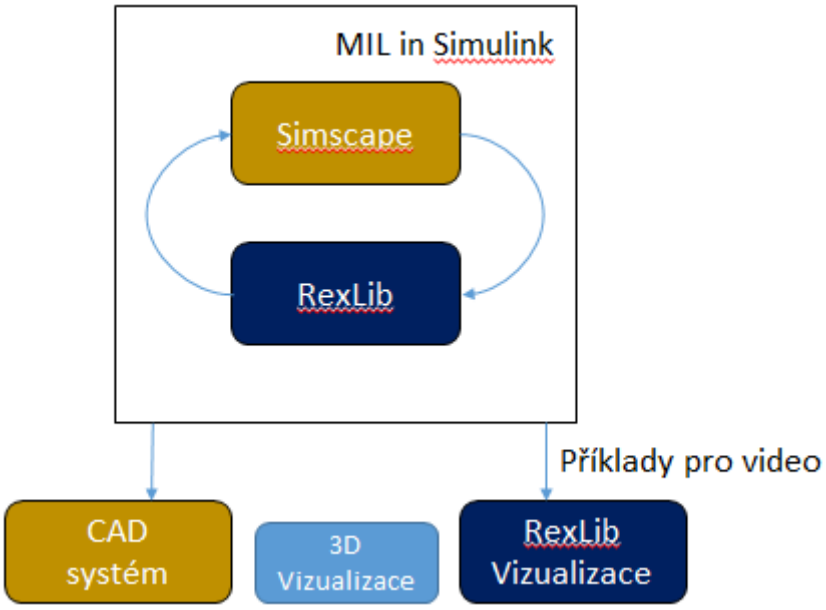
Software	Licence	Simulace	Funkce
V-REP	komerční studentská open source	mobilní statické roboty	rychlý vývoj algoritmů rychlé prototypování reálné prostředí
Webots	komerční studentská freeware NAO	mobilní statické roboty	návrh komplexního robotického nastavení sdílené prostředí
Gazebo	komerční studentská open source	mobilní statické roboty	rychlý vývoj algoritmů rychlé prototypování
Breve	open source	mobilní roboty	simulace multiagentních systémů a umělého života

Hlavním důvodem pro volbu prostředí byla jeho všestrannost. Jedná se o jeden z nejpřijatelnějších softwaru pro HIL simulaci. Nabízí velké možnosti programování nejen simulačního prostředí, ale i jednotlivých objektů. Mezi další výhodu patří velká spousta knihoven a návodů dostupných, jak na webových stránkách https://www.coppeliarobotics.com, tak na fóru https://forum.coppeliarobotics.com. Z těchto zdrojů jsem čerpal i při vypracování vzorové úlohy.

5 SIMULACE VE SMYČCE

5.1 MIL - Model in the loop

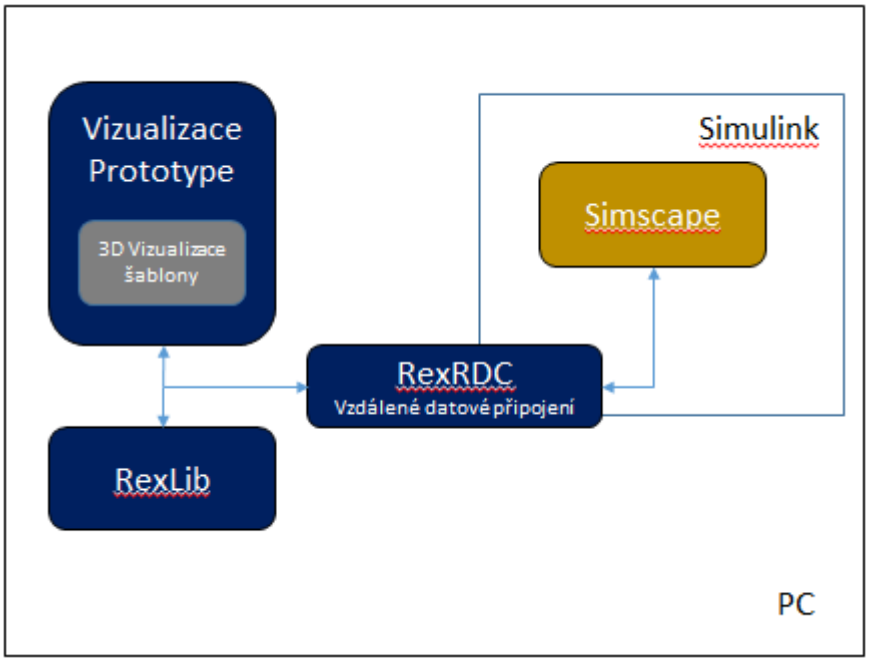
Jedná se o první návrh řídící jednotky, u nichž se neuvažuje vzájemné propojení zařízení, rychlost komunikace a výpočtový výkon. Matematický model a řídící systém je následně simulován v simulačním prostředí. Řízená soustava je za pomocí identifikace následně aproximována, neboť jsou vytvořené vstupy a výstupy. To má za následek, že vytvořená soustava a zároveň návrh, nemůže být dostatečné přesný. MIL simulace slouží primárně k ověření a optimalizace konstrukce stroje a návrhu struktury řízení. [17] [18]



Obr. 14) MIL schéma

5.2 SIL - Software in the loop

Je důležité upozornit, že v Simulinku se dá použít toolbox real time work pro generování funkčního kódu. Tento kód může pracovat jinak než požadovaný model, který je poskládaný z bloků, jenž obsahuje Simulink. Matematický model je společně s řídícím systémem simulován v reálném čase. Principem testování je porovnat výsledky dosažené z navrženého modelu a funkčnosti vygenerované S-funkce, která se spouští přes Simulink. [17] [18]



Obr. 15) SIL schéma

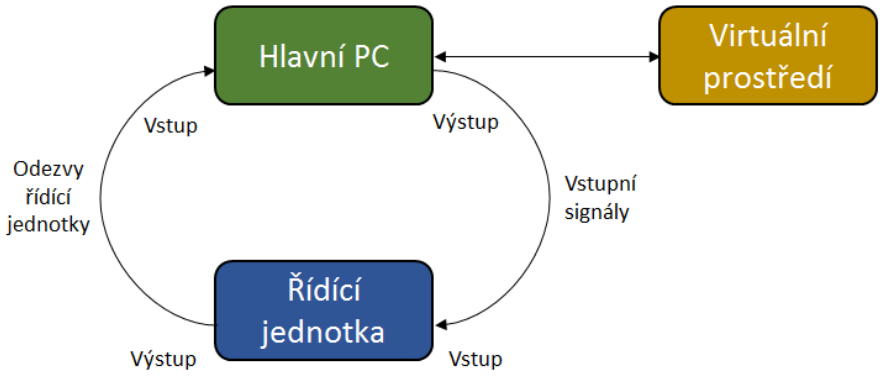
5.3 PIL - Processor in the loop

U této metody se využívá reálný procesor, jenž je použitý v reálné centrální elektronické řídící jednotce (ECU). Model ECU a požadovaný matematický model soustavy se přeloží do spustitelného kódu a následně se nahraje do hardwaru. Matematický model soustavy lze nahrát do PC a model řídící jednotky do vývojové desky osazené daným procesorem. Tento procesor je určený na budoucí využití v dané řídící jednotce. Tyto dvě zařízení lze připojit pomocí průmyslového komunikačního rozhraní, např. RS232, CAN nebo ethernet. Při propojení nejsou využity žádné vstupní a výstupní karty i čidla. [17] [18]

Hlavním cílem testu je ověřit a odladit výkon procesoru i rychlost komunikace mezi jednotlivými zařízeními a testování kritických situací, které mohou v reálném procesu nastat. Test probíhá v reálném čase. Je ovládán z libovolného PC, který splňuje požadavky na výkon. [17] [18]

5.4 HIL - Hardware in the loop

U metody Hardware in the loop je v řídící smyčce zapojená část reálného softwaru. Nejčastěji se jedná o řídící jednotku ECU společně s matematickým modelem systému, jež je simulovaný na HIL testovací platformě. [17] [18]



Obr. 16) HIL schéma

Model soustavy běží v reálném čase na speciálním Hardwaru (target PC), který řídí chování soustavy pomocí reálného komunikačního rozhraní (A/D, D/A převodníky, proudové slučky atd.), takže řídící jednotku nerozpozná, jestli komunikuje s reálnou soustavou, nebo simulátorem. Připojení ke snímačům, stejně jako zásahy do systému, se uskutečňují tak, jak by to bylo v reálném provozu. [17] [18]

Simulace je řízená z PC (tzv. Host PC), na kterém je celý model, nebo může být jednoúčelově řízený. Velká výhoda řízené simulace je možnost měnit parametry během běhu, sledovat její průběh a možnost získávat data. Jednoúčelová simulace se používá převážně u automatizovaných testů. [17] [18]

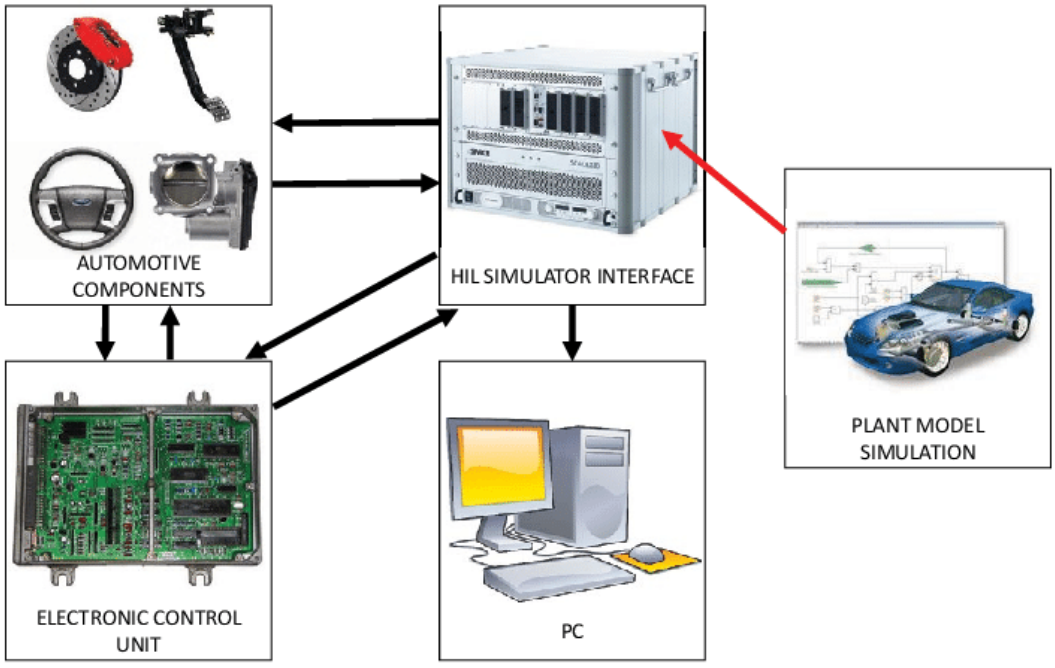
Díky nezávislosti simulace na reálné soustavě je možnost simulovat také podmínky, které by nebylo možné v realitě vytvořit (např. nastavit některé signály jako konstanty), což má za následek zjednodušení a tím pádem i urychlení ladění. Mezi další výhody patří opakovatelnost podmínek simulace a také možnost zrychlit simulaci, což by bylo v realitě nesmírně složité (né-li nemožné). Mezi největší výhody testování pomocí HIL simulace patří ušetření finančních prostředků a snížení času testování, protože soustavu nelze poškodit. [17] [18]

6 HARDWARE IN THE LOOP

Testovací metoda HIL má velké výhody především v šetření nákladů na vývoj a přesnosti získaných výsledků, které se téměř shodují s reálnými. Tento fakt klade vysoké nároky na hardware a samotná realizace simulace zároveň požaduje vysoce výkonné PC. Tyto náklady se však v budoucnosti několikanásobně vrátí ve formě vynaložených nákladů a hlavně času pro výrobu drahých prototypů a jejich testování. [17][18]

Testovací metoda HIL je poslední vývojová fáze MBD (model based design). Po odladění těchto řídících zařízení je ECU zapojené už do reálného zařízení a následuje pouze finální doladění. [18]

HIL testovací metoda má velmi široké možnosti uplatnění převážně v letectví, armádě, automobilovém průmyslu apod. Používá se především pro simulaci elektromagnetických systémů, které jsou řízené elektronickou řídící jednotkou ECU. [18] [19]



Obr. 17) Příklad užití HIL simulace [20]

6.1 Současný stav metod HIL

Testovací metoda HIL zatím není standardizovaná, ale i přesto je možné najít značnou podporu u komerčních poskytovatelů jako např. National Instruments, dSpace, ADI atd.

Řešení od těchto firem jsou komplexní. Poskytují hardware a zároveň software potřebný pro realizaci. Nevýhodou je, že všechny firmy mají založený vlastní vývoj na těchto produktech. Některé firmy využívají software od firmy Mathworks, a proto je pro ně nejvýhodnější sáhnout po toolboxech, které podporují komerčně dostupné DAQ (Data Acquistion) měřící

karty. Tento způsob realizace simulace umožňuje široké rozšíření a přizpůsobení parametrů na konkrétní požadavky. Nevýhodou je omezená podpora DAQ měřících karet. [17] [18]

6.2 Přehled metod

Rozdělení metod bylo vytvořené na základě kombinaci dvou parametrů embedded sytémů:

- počtu vstupní a výstupních signálů
- časové odezvy

Tyto dva parametry kladou primárně požadavky na hardwarovou konfiguraci dané metody a vyplývá z nich užší specifikace použitelného hardwaru pro simulaci daného sytému.

6.2.1 HIL simulátory základních jednoduchých a pomalých systémů

1-10 vstupních a výstupních kanálů a pomalá časová odezva > 10s.

Odezva těchto systémů je řádově desítek sekund a více. Převážně jsou určené pro ukázku a demonstraci HIL systémů a pro výukové účely. Jejich výhoda v této oblasti je zkrácení testovacího času. Omezení simulace vyplývá jen z rychlosti odezvy řídící jednotky, což urychlí vývoj. Na nenáročnou simulaci se mohou využít i zastaralé PC. [17]

Mezi nejčastější oblasti využití patří regulace: teploty v místnosti, hladiny v nádrži a polohy v prostoru. [17]

6.2.2 HIL simulátory středně náročných a rychlých systémů

10-100 vstupních a výstupních kanálů a časová odezva $10 – 10^{-6}$s.

Odezva těchto systémů se pohybuje řádově mezi sekundami a mikrosekundami. Jsou to běžné systémy v oblasti mechatroniky. HIL se požívá na vývoj nových řídících systémů a jejich ladění. Převážně na tuto oblast jsou zaměřené komerční řešení, protože tyto systémy jsou nejrozšířenější v oblasti mechatroniky. Využívá se hardware založený na PC platformě, nebo specializovaný hardware pro tyto účely např. National Instruments. [17]

Oblast použití: podpora řízení automobilu, ABS, start spalovacího motoru atd. [17]

6.2.3 HIL simulátory náročných a extrémně rychlých systémů

100-1000 vstupních a výstupních kanálů a časová odezva $10^{-6}– 10^{-9}$s.

Odezva systémů se pohybuje řádově v mikrosekundách až nanosekundách. Tyto systémy vyžadují rychlé a přesné řízení. Jejich simulace klade vysoké nároky na hardware, hlavně na jeho výpočtový výkon. Využívá se hrdlových polí PGA a FPGA, které umožňuje paralelní výpočty pro více operací najednou. Zde je kladen velký důraz na hard real-time. Pro využití této aplikace je nutné používat vysoce výkonné PC s několika procesory. [17]

Oblast použití: HUMS (Health and usage monitoring systems), řízení rychlosti a vojenské techniky, regulace výšky letadla a kardiostimulátory v oblasti medicíny. [17]

6.2.4 HIL simulátory komplexních systémů

1000 a více vstupních a výstupních kanálů, desítky subsystémů s různou časovou odezvou.

Jde o test komplexního systému, u kterého je možný paralelní vývoj a práce na jednotlivých subsystémech. Jednotlivé subsystémy si vzájemnou komunikací předávají výsledky simulace a zároveň mezi sebou komunikují. Například subsystém se vstupním parametrem teploty komunikuje s jiným subsystémem, který simuluje teplotní čidlo. Tuto metodu obsahují velké integrované laboratoře, které je možné řídit v reálném čase, kde PC vykonává veškeré výpočty. U této metody je kladen požadavek na to, aby všechny výpočty vykonával monolitický počítač. Zároveň vyžaduje synchronizaci všech simulátorů řízených v reálném čase a je náchylný na chyby způsobené při běhu simulace a v komunikaci. [17]

Oblast využití: vysoce výkonné testy proudových motorů, vývoj v automotiv, řízení letadel.

7 PRAKTICKÁ ČÁST

V této kapitole se nejprve seznámíme s měřící kartou, tedy reálným hardwarem, který zastupuje mikrokontroler Arduino UNO, a všemi komponenty, které jsou do něj zapojeny. Dále s programovacím prostředím Matlabem a simulačním prostředím. Pro tuto úlohu jsem zvolil simulační prostředí od firmy CoppeliaRobotics tedy V-REP, což je popsáno v kapitole Volba softwaru. Na závěr této kapitoly jsou popsány dvě vzorové úlohy, kde je popsána sériová komunikace mezi Arduinem a Matlabem. Dále pak spojení přes API mezi Matlabem a simulačním prostředím V-REP.

7.1 Arduino

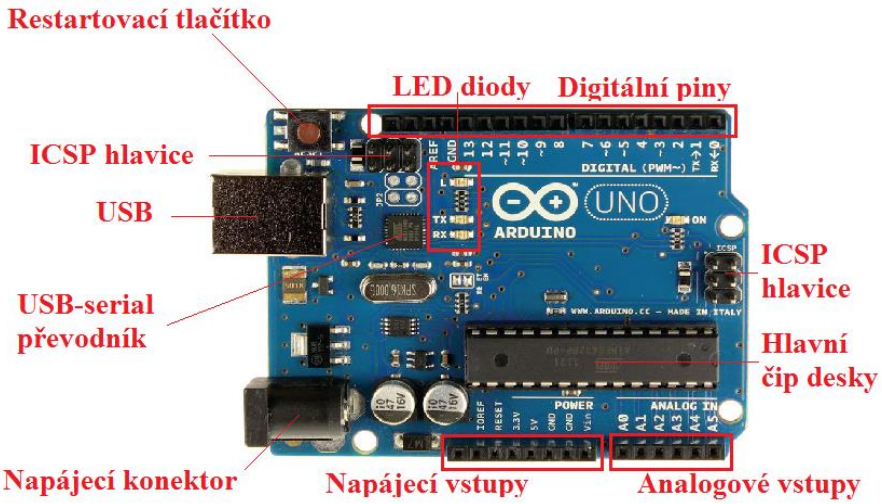
Arduino je open-source platforma založená na mikrokontrolerech (MCU) ATmega vyráběné firmou Atmel. Je to velice malý jednodeskový počítač. Používají jej inženýři, umělci, programátoři, studenti nebo lidé, které mají elektrotechniku jen jako koníček. Avšak Arduino mohou užívat i úplní nováčci ve světě programování a elektrotechnice, a to z toho důvodu, že je k nalezení nespočet návodů a projektů. K dispozici zcela zdarma přímo na stránkách Arduina nebo na fórech. Na druhé straně můžeme tento mikroprocesor najít v mnoha komplikovaných a komplexních projektech. Komunitní logo Arduino na Obr. 18 zdůrazňuje, že se jedná o open-source platformu, kterou si může uživatel libovolně přizpůsobit. [21] [22]

Obr. 18) Logo Arduino open-source community [21]

7.1.1 Arduino UNO

Jedná se o nejpoužívanější desku platformy Arduino. Desku používá procesor od firmy Atmel, přesněji ATmega328 pracující na frekvenci 16MHz. Samotná paměť programu a jednotlivých dat je oddělená, proto je zde využita Harvardská architektura počítače. V dnešní době se deska prodává už ve své třetí revizi nesoucí označení R3 nebo Rev3. Největší výhoda této desky je poměr velikosti, snadný přístup k pinům a cenou. Pro Arduino Uno je vytvořena většina shieldů, rozšíření jako například internetové připojení. Uno lze připojit k počítači pomocí USB portu, bluetooth nebo wifi. V našem případě se jedná

o propojení přes USB kabel přímo k PC. Arduino je v operačním systému připojeno na COM porty, které umožňují oboustrannou komunikaci. [22] [23]



Obr. 19) Arduino UNO

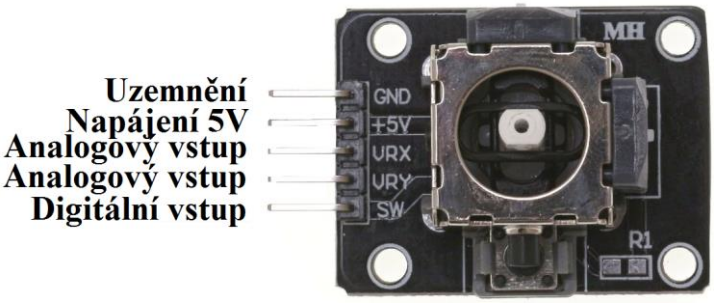
Tab 2) Parametry mikrokontroleru UNO [23]

Mikroprocesor	ATmega328P
Provozní napětí (logická úroveň)	5 V
Vstupní napětí (doporučeno)	7-12 V
Vstupní napětí (maximální)	6-20 V
Počet digitálních I/O pinů	14 pinů, z toho 6 s PWM
Počet analogových vstupů	6 pinů
Proudové zatížení na 1 pin	20 mA
Flash paměť	32 kB
SRAM	2 kB
EEPROM	1 kB
Výška x šířka	68.6 mm x 53,4 mm
Hmotnost	25 g

7.1.2 Joystick PS2

Jedná se o herní modul, který snímá pohyby páčky ve dvou osách. Joystick se skládá ze dvou potenciometrů o velikosti 10 kΩ a tlačítka, které detekuje stisknutí páčky. V základním postavení je potenciometr nastaven na polovinu svého rozsahu. Při pohybu se pak jeho odpor mění v rozsahu ±5 kΩ. Následně lze v softwaru přepočítat výsledné napětí na souřadnice. [24]

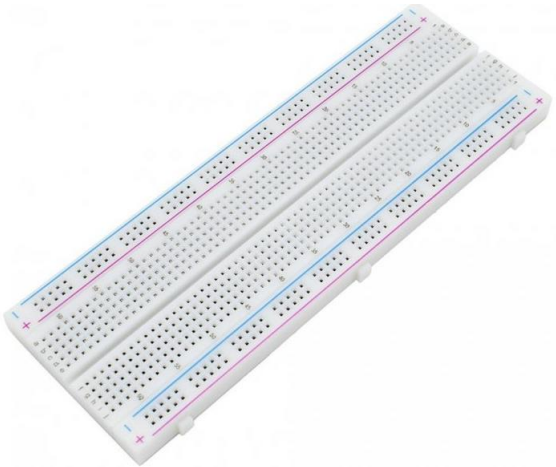
Obr. 20) Joystick ps2 [24]



Obr. 21) Joystick ps2 – piny [24]

7.1.3 Nepájivé kontaktní pole 830 pinů

Je to opakovatelně použitelná deska sloužící pro navrhování prototypů elektrických obvodů pro experimenty bez nutnosti pájení. Je vyrobena z plastu a otvory mají průměr 2,54 mm. Propojení desky a modulu je zajištěno pomocí kontaktních hřebínků s pružinami, které jsou zaseknuty zespod desky. Drátky je možné opakovaně měnit a různě přepojovat bez poškození. [25]



Obr. 22) Nepájivé kontaktní pole 830 pinů [25]

7.1.4 Ostatní komponenty

Tlačítko, spínač - do obvodu se přidávají, protože po jejich stisknutí se spojí nebo rozpojí dva body v obvodu např. rozsvícení LED diody atd.

LED dioda - v obvodu zastupuje funkci informativní. V momentě, kdy je splněna podmínka (např. stisknuto tlačítko), LED dioda se rozsvítí. Podmínka může být vytvořena

pouze v softwaru. Po zadání příkazu *digitalWrite(led,HIGH);* se LEDka rozsvítí. Naopak příkaz *digitalWrite(led,LOW);* světlo zhasne.

Obr. 23) Komponenty Arduino [26]

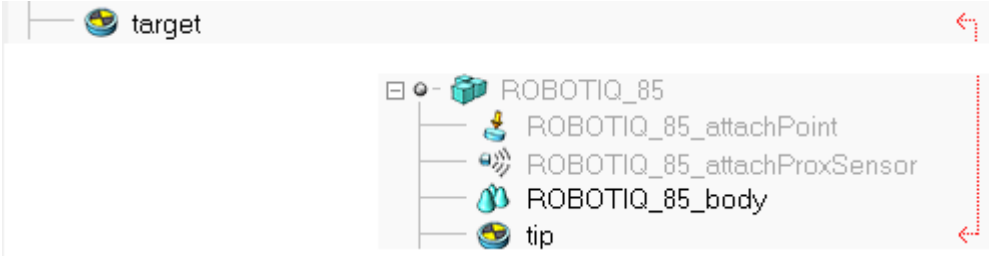
7.2 Matlab

Je programové prostředí a skriptovací programovací jazyk. Program pracuje na operačních systémech Windows, Linux a Mac OS X. Původně byl program vyvinut pro programovací účely. Postupem času došlo k vývoji a značnému rozšíření a nyní lze software využívat u mnoha aplikací např. analýza a prezentace dat, vytváření aplikací, a to včetně uživatelského rozhraní, modelování a simulace, vědecká a inženýrská grafika atd. To z něho tvoří multifunkční aplikaci, která je využívána ve velké spoustě projektů. Základní vlastnost Matlabu je, že všechny objekty v Matlabu jsou považovány za prvky pole (matice). Výkonnost je rozšiřována pomocí navazujícího softwaru, které tvoří tzv. „toolboxy“. Ty jsou orientované a přímo upravené danému problému.

V práci plní funkci prostředníka. Předává informace mezi virtuálním prostředím a hardwarem. Rychlost přenosu je dána výkonem PC. U simulátoru je primárně nastavena rychlost přenosu informace na periodu 50 ms. Periodu je možné měnit a to až na hodnotu 200 ms.

7.3 V-REP

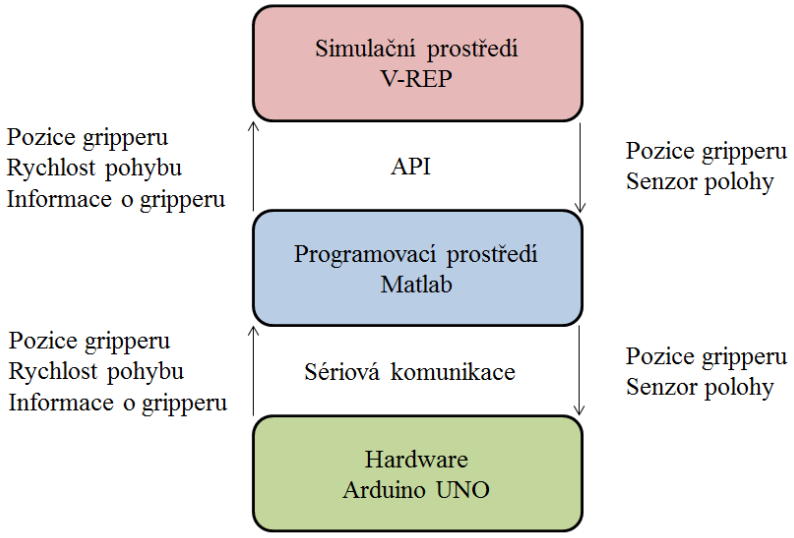
V simulačním prostředí jsou vytvořeny dvě scény pro dvě úlohy. Pro obě úlohy platí, že pohyb manipulátoru je realizován pomocí inverzní kinematiky. V tomto případě to znamená, že je známa poloha na konci ramena a je zadaná nová poloha, kterou manipulátor následuje. V tomto případě je vytvořený bod tip a target. Tip je umístěný na koncovém bodu ramene robota a následuje polohu bodu target, viz Obr. 24.



Obr. 24) Propojení bodů přes inverzní kinematiku

7.4 Komunikace

Na Obr. 25 je znázorněné schéma přenosu informací od simulačního prostředí po reálný hardware. Přenášíme pouze nezbytně nutné informace, aby byl přenos plynulý. Mezi informace patří: poloha koncového bodu manipulátoru, informace o gripperu a rychlost pohybů robota.

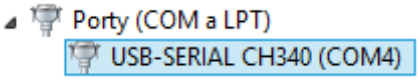


Obr. 25) Schéma komunikace

7.4.1 Komunikace přes sériovou komunikaci

Pro správné fungování HIL simulace bylo nutné vytvořit komunikační spojení mezi reálným hardwarem (Arduino UNO) a softwarem (Matlab). Toto spojení je realizováno pomocí fyzického USB portu, jenž je propojen s PC a programově s ním pracuje jako s dalším sériovým portem.

Pro správné fungování sériové komunikace je nutné nejprve stáhnout aktualizované software ovladače. COM porty nalezneme na adrese: Ovládací panely\Hardware a zvuk\Správce zařízení. Následně vybereme požadovaný Port (COM a LPT), (viz Obr. 26) a spustíme aktualizaci. Po aktualizaci ovladačů můžeme využívat COM port pro sériovou komunikaci.



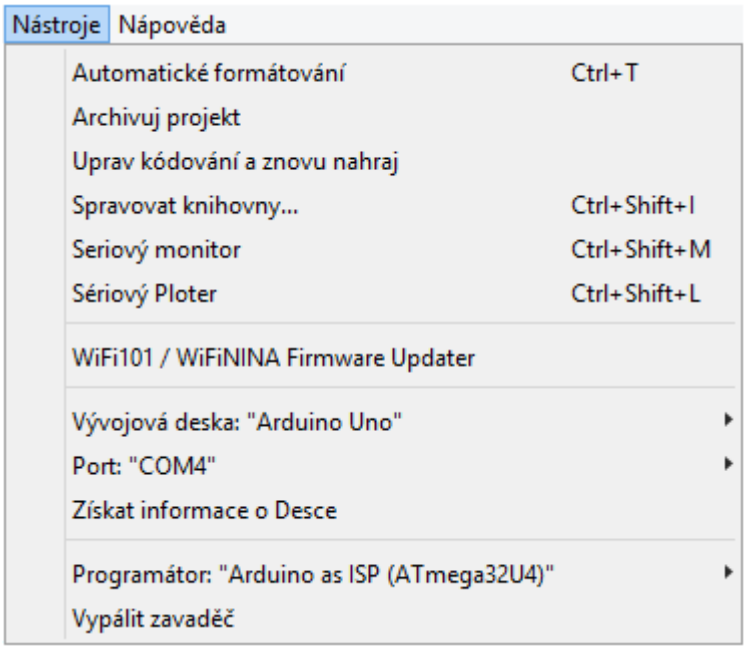
Obr. 26) COM port

Následně je potřeba správně nakonfigurovat Arduino v aplikaci Arduino IDE, která je volně dostupná na internetových stránkách.

Všechna nastavení se provádějí v záložce Nástroje, viz Obr. 27.

- Nejprve vybereme zvolenou desku, v tomto případě se jedná o vývojovou desku Arduino UNO.

- Dále vybereme port pro sériovou komunikaci. Volíme port, který jsme si v předchozím kroku aktualizovali tedy COM4.
- Na závěr nastavíme programátor: Arduino as ISP (ATmega32U4).



Obr. 27) Nastavení Arduina

Pro správné fungování JSON v arduinu je nutné stáhnout knihovnu a následně ji přivolat na začátku úlohy pomocí příkazu: <ArduinoJson.h>.

7.4.2 Komunikace Matlabu s V-REP

Matlab komunikuje se simulačním softwarem pomocí vzdáleného připojení API. Po zadání příkazu `vrep=remApi('remoteApi');` je možné komunikovat přes API, které umožňuje čtení signálů, viz Obr. 28. Do proměnné j1, j2 se nám uloží hodnoty z gripperu, do kuka_target uložíme polohu bodu a poslední proměnná představuje hodnotu senzoru uloženou do proměnné Proximity_sensor.

```
[res,j1]=vrep.simxGetObjectHandle(clientID,'ROBOTIQ_85_active1',vrep.simx_opmode_blocking);
[res,j2]=vrep.simxGetObjectHandle(clientID,'ROBOTIQ_85_active2',vrep.simx_opmode_blocking);
[res,kuka_target]=vrep.simxGetObjectHandle(clientID,'target',vrep.simx_opmode_blocking);
[res,Proximity_sensor]=vrep.simxGetObjectHandle(clientID,'Proximity_sensor',vrep.simx_opmode_blocking);
```

Obr. 28) Načtení hodnot ze simulačního prostředí

Poté co Matlab obdrží požadované hodnoty od měřící karty, odešle tyto hodnoty zpět do simulátoru, kde prostředí V-REP vykoná akci na základě přijaté informace. Jedná se o informace ohledně polohy, rychlosti pohybu robota atd.

Pohyby robota jsou ovládány pomocí vytvořené funkce „MoveL“. Pro spuštění funkce je potřeba zadat, který bod se má dostat na zadanou cílovou polohu a rychlost otáčení robota. Ve funkci pobíhá výpočet dráhy v závislosti na nastavení rychlosti simulace v simulačním prostředí.

```
function moveL(clientID,target, pos, speed)
vrep=remApi('remoteApi');
%Získání cílové pozice
[r,p]=vrep.simxGetObjectPosition(clientID, target, -1, vrep.simx_opmode_blocking);
%Výpočet pohybu robota
old_pos=[p] ;
delta_pos=pos - old_pos;
distance=norm(delta_pos);
samples_number=round(distance*50);
for i=1:samples_number;
    intermidiate pos= old pos + (delta pos/samples number);
```

Obr. 29) Část kódu z funkce „MoveL“

Ovládání gripperu je ošetřeno vytvořenou funkcí „gripper“. Funkce ovládá dva body gripperu, které zajistí uchycení objektu. Hlavní informace, která do této funkce vstupuje je, jestli chceme gripper otevřený, nebo zavřený.

```
function gripper (clientID,closing,j1,j2)
vrep=remApi('remoteApi');
[r,p1]=vrep.simxGetJointPosition(clientID,j1,vrep.simx_opmode_blocking);
[r,p2]=vrep.simxGetJointPosition(clientID,j2,vrep.simx_opmode_blocking);
if (closing==1)
    if (p1 <(p2-0.008))
        vrep.simxSetJointTargetVelocity (clientID,j1,-0.2,vrep.simx_opmode_blocking);
        vrep.simxSetJointTargetVelocity (clientID,j2,-0.2,vrep.simx_opmode_blocking);
    else
```

Obr. 30) Část kódu z funkce „gripper“

V-REP komunikuje s Matlabem přes vzdálené API. Pro použití API je potřeba povolit vzdálené připojení a v kódu přivolat funkci remAPI. Po úspěšném připojení dostaneme hlášku Vzdálený API server připojen, viz Obr. 31.

```
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
disp('Vzdálený API server připojen');
```

Obr. 31) Připojení API k Matlabu

Propojení COM portu v Matlabu je popsáno na Obr. 32. Komunikace mezi Matlabem a Arduinem, je založena na sériové komunikaci ve formě JSON.

JSON (Java Scrip Object Notation) je jednoduše čitelný a zapisovatelný. Díky tomu, že využívá známé programovatelné jazyky rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších), je ideálním jazykem pro výměnu dat. Vzhledem k tomu, že podporuje velké množství programovacích jazyků, je JSON ideální výměnný formát nezávislý na programovacím jazyku.

```
%% Kontrola sériových objektů a připojení portu
    port = 'COM4';
    oldSerial = instrfind('Port', port);
    if (~isempty(oldSerial))
        disp('POZOR:  Port je používán. Zkontroluj port. Zavírám program.')
        delete(oldSerial);
        a = true;
    else
        disp('Připojuji serial port...');
    end
%%  Nastavení seriove komunikace
    %Připojení Arduina přes COM port
    s = serial(port,'baudrate',9600,'terminator','CR','tag','Quad');
    %Počkej než přitečou data
    set(s, 'TimeOut', 5);
    % Nastavení dostatečně velké paměti pro všechna data
    set(s, 'InputBufferSize',  390 );
    % Před zápisem dat je nutné port otevřít
    fopen(s);

    disp('Serial port otevřen');
```

Obr. 32) Otevření COM portu v Matlabu

7.5 Vytvoření komunikace

Sériová komunikace mezi Matlabem a Arduinem je založená na vysílání dat rozložených na jednotlivé bity. Kromě užitečných dat se přenos doplňuje o další informace. Typicky se jedná o informace sloužící pro synchronizaci, hlášení chyb nebo jejich korekci. Přenos bitů musí být synchronizován, aby přijímací strana od sebe jednotlivé bity rozeznala a správně přečetla. Pro správnou komunikaci a přenos proměnných a jejich hodnot v práci využívám formát JSON. Formát podporuje velké množství programovacích jazyků.

Po zpracování příchozích dat z V-REP jsem v Matlabu vytvořil strukturu (test = {};). Struktura je umístěna v nekonečném cyklu while a neustále načítá data ze simulačního prostředí.

```
while 1
    % Načtení senzoru polohy
    [res,PSensor_distance, detectedPoint]= vrep.
    test.PSensor=PSensor_distance;
    test.cube_size = detectedPoint(1,3)*1000;
    test.senzor = 1;
    test.time = 0;
    test.gripper = p;
```

Obr. 33) Načtení proměnných do struktury „test“

Proměnné jsou zabaleny do formátu JSON. Následně jsou postupně načítány na sériovou linku. Po zapsání posledního znaku, uloženého do struktury test, Matlab přeruší psaní a odešle strukturu, kterou si Arduino načte.

```
%Vytvoř strukturu JSON a odešli do portu COM
encoded_json = jsonencode(test);
fprintf(s, '%s\n', encoded_json);
```

Obr. 34) Kódování do formátu JSON

V Arduinu se nachází string s názvem matlab, do kterého se ukládají informace ze sériové komunikace. Po načtení posledního znaku lze proměnné dále zpracovat.

```
        char c = Serial.read();
        matlab += c; //vytvoření stringu matlab
        if(c == "\n"){
          break;
        }
       }
      }
//Nacteni struktury matlab ze seriove komunikace format JSON
   if(matlab == "\n"){
     return;
```

Obr. 35) String „matlab“ v Arduinu

Sériová linka je neustále v provozu. Jestliže se změní hodnota, která nás zajímá, Arduino si ji vytáhne a např. pozice, která přišla ve formě matice je potřeba zpracovat. V Arduinu ji je třeba rozdělit a pro každý prvek vytvořit proměnnou zvlášť. Pro ostatní platí, že se uloží do další proměnné, s kterou bude pracovat Arduino. Po úspěšném načtení všech informací, pokračuje kód Arduina.

```
//Nacteni promenych z Matlabu
long x0 = readDoc["pozice"][0];
long y0 = readDoc["pozice"][1];
long z0 = readDoc["pozice"][2];

long vrepspeed = readDoc["speed"];
long vrepopen_close = readDoc["open_close"];
long time_ = readDoc["time"];


long PSensordist = readDoc["PSensor"];
long detectPoint = readDoc["cube_size"];
long rvrepkuka_target = readDoc["vrepkuka_target"];
long rvrepProximity_sensor = readDoc["vrepProximity_sensor"];
```

Obr. 36) Čtení dat ze sériové komunikace pro Arduino

Na obrázku níže je ukázka kódu vytvořeného v Arduinu IDE. Pokud se manipulátor nachází v požadované poloze, tak se načte poloha nová a s tím informace, jakou rychlostí se má přesunout. Dále informace ohledně otevření/uzavření gripperu. V další části kódu do hry vstupuje informace ze senzoru polohy, který zjistí, zda na páse stojí malá nebo velká kostka.

Schéma rozhodovacího mechanismu je na Obr. 37.

```
if (time_ > 0.5) {
   if (PSensordist > 0) {
       if (detectPoint < 167) {
        cube = 1;
       }
       else  {
        cube = 2;
   }
```

Obr. 37) Rozhodovací podmínka v Arduinu

Na základě rozhodovacího mechanismus hardware určí, kam kostka poputuje.

```
  //Zpracovani dat
if (time_ < 0.5){
  writeDoc["pozice"][0] = -0.11;
  writeDoc["pozice"][1] = -0.3;
  writeDoc["pozice"][2] = 0.75;
  writeDoc["speed"] = 5;
  writeDoc["open_close"] = 0;
}
```

Obr. 38) Zpracování dat v Arduinu

Na konci smyčky se nachází kód, který načtené hodnoty z Arduina opět zabalí a odešle zpět na sériovou linku, odkud si ji může Matlab vytáhnout a zpracovat.

```
//Odeslání dat do matlabu
writeDoc["time"] = time_;
writeDoc["PSensor"] = PSensordist;
writeDoc["cube_size"] = detectPoint;
writeDoc["hodnotakostky"] = cube;
writeDoc["vrepkuka_target"] = rvrepkuka_target;
writeDoc["vrepProximity_sensor"] = rvrepProximity_sensor;
```

Obr. 39) Odeslání dat na COM port

Jakmile se změní hodnota ve struktuře, Matlab ji načte do proměnné arduino, dekóduje ji tak, aby mohly být proměnné zpracovány a odeslány do simulačního prostředí. Tím se smyčka uzavírá a komunikace probíhá, dokud není přerušena.

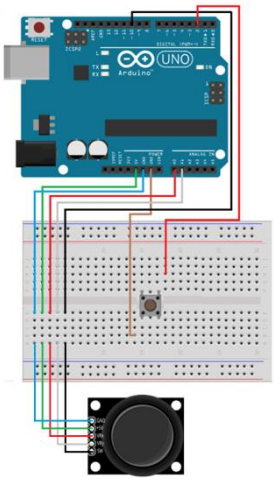
```
try
    %Dekoduj strukturu JSON pro matlab
    b = jsondecode(arduino);
    fprintf(s, '%s\n', encoded_json);
    act_pozice = reshape(b.pozice,1,3);
```

Obr. 40) Dekódování struktury JSON v matlabu

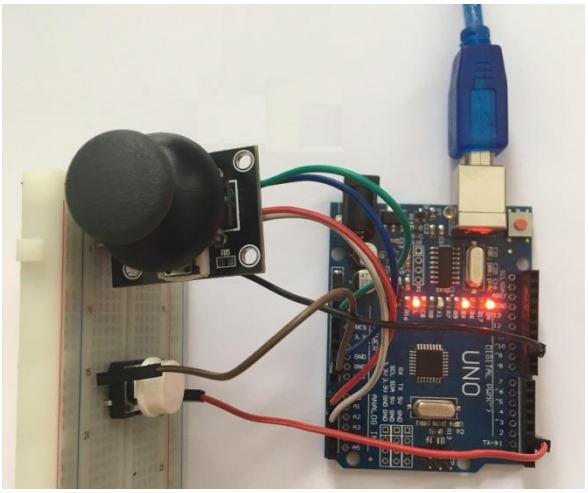
7.6 ÚLOHA 1 - Ovládání robota pomocí joysticku PS2

Hardware

Na Obr. 40 je vidět propojení jednotlivých komponentů Arduina. Joystick PS2 je propojen pomocí 5V a zemnícího pinu. Pro vstupní signály slouží dva piny analogové (A0 a A1) a jeden digitální pro stisk joysticku (D10). Dále je na nepájivé pole připojeno tlačítko pomocí dvou pinů. Jeden pin je uzemněný a druhý digitální, který slouží pro přenos signálu po stisknutí tlačítka (D2).



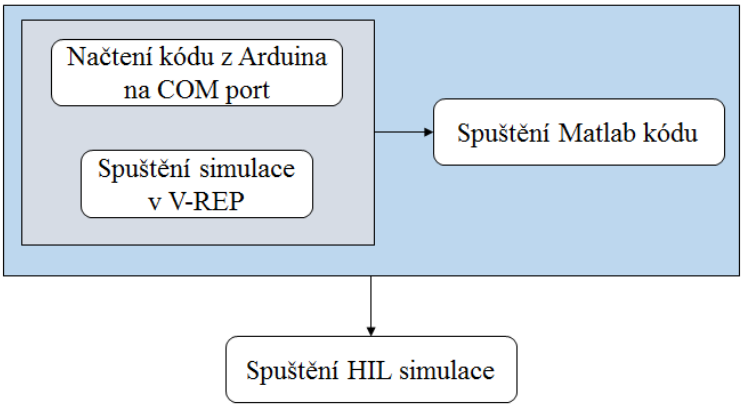
Obr. 41) Schéma zapojení - Arduino UNO



Obr. 42) Zapojení komponentů do Arduina UNO

Simulační prostředí V-REP obsahuje:

- KUKA KR16
- gripper ROBOTIQ_85
- dopravník
- odkládací stůl
- senzor polohy



Obr. 43) Schéma spuštění simulace

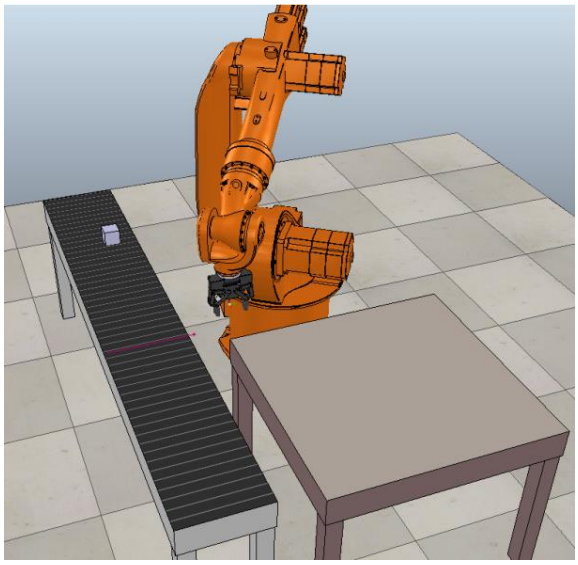
Poté, co se otevřou všechny programy a připojí se Arduino, lze spustit simulaci na jednotlivých programech, viz Obr. 41. Nejprve načteme kód z Arduina na COM port. Následně spustíme simulaci v simulačním prostředí V-REP a na závěr spustíme skript v Matlabu. Po spuštění vyčkáme na hlášku: „POZOR – Začni ovládat robota!“ (viz Obr. 42). Poté lze joystickem ovládat pohyby robotu.

```
Command Window
  Připojuji serial port...
  Serial port otevřen
  Note: always make sure you use the corresponding remoteApi library
  (i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
  Vzdálený API server připojen
  POZOR - Začni ovládat robota!
  Note: always make sure you use the corresponding remoteApi library
  (i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

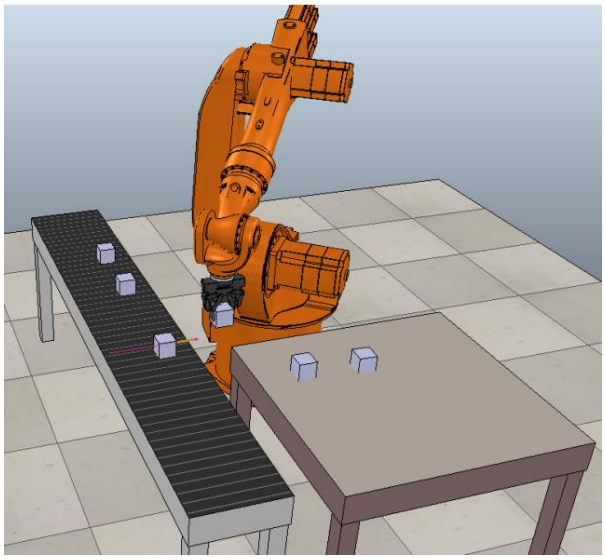
Obr. 44) Spuštění simulace

Popis úlohy:

Na dopravník jsou generovány kostky. Jakmile je senzor polohy zachytí, pás se zastaví a čeká, dokud nebude kostka posunuta mimo zónu senzoru. V mezičase operátor může robotem libovolně manipulovat. Koncový bod ramene robota se pohybuje ve stálé výšce 0.75 m. Manipulátorem lze libovolně pohybovat ve dvou osách. Jakmile najedeme nad kostku, stiskneme joystick a gripper změní svoji výšku tak, aby mohl kostku uchytit. Pro aktivaci gripperu je potřeba držet bílé tlačítko (viz Obr. 45), které dá pokyn, aby se gripper zavřel. Po uvolnění joysticku koncový bod najede na původní výšku a můžeme přemístit kostku na pracovní stůl. Po uvolnění tlačítka se gripper otevře a upustí kostku.



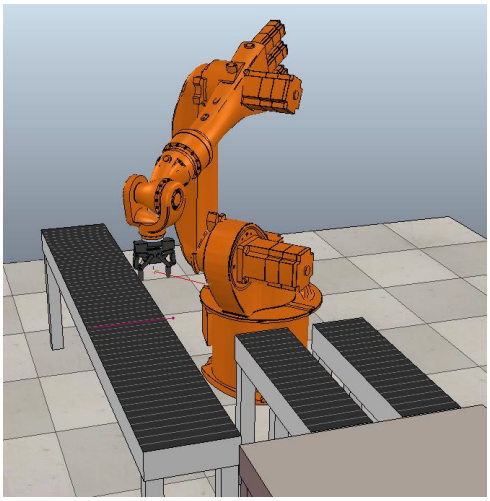
Obr. 45) Počáteční stav



Obr. 46) Ukázka simulace

7.7 ÚLOHA 2 – Hardware in the loop

V této úloze je řešena komunikace mezi V-REP a Matlabem stejně jako u předešlé úlohy. Liší se tím, že pohyby robota neovládá operátor, ale je ovládán reálným hardwarem, který zde zastupuje mikrokotroler Arduino UNO.



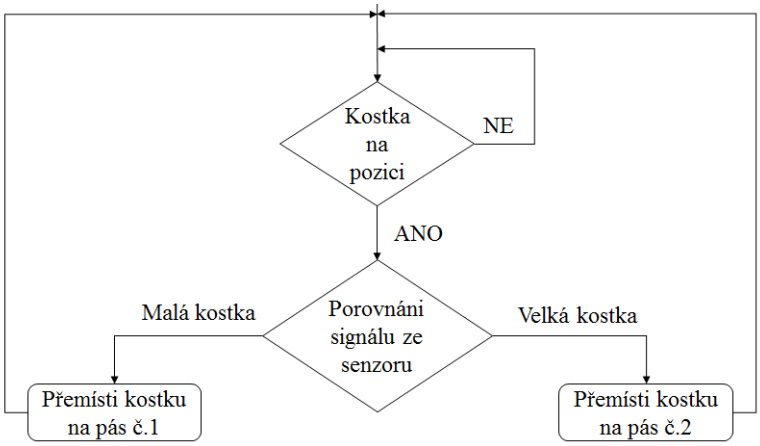
Obr. 47) Ukázka simulační scény

Simulační prostředí V-REP

- KUKA KR16
- gripper ROBOTIQ_85
- dopravník hlavní
- vedlejší dopravníky
- odkládací stůl
- senzor polohy

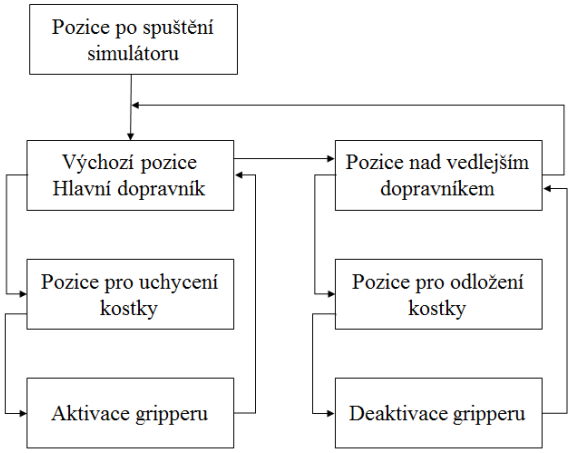
Popis úlohy:

Po hlavním dopravníku přijíždějí kostky různých velikostí. Po propojení simulačního prostředí s Matlabem i Arduinem najede robot do výchozí polohy nad Proximity senzor a čeká. Poté, co senzor dostane signál, že je kostka na místě, odešle do Arduina hodnotu ze senzoru. Arduino signál zpracuje a vyhodnotí, zda se jedná o velkou či malou kostku, viz Obr. 48.



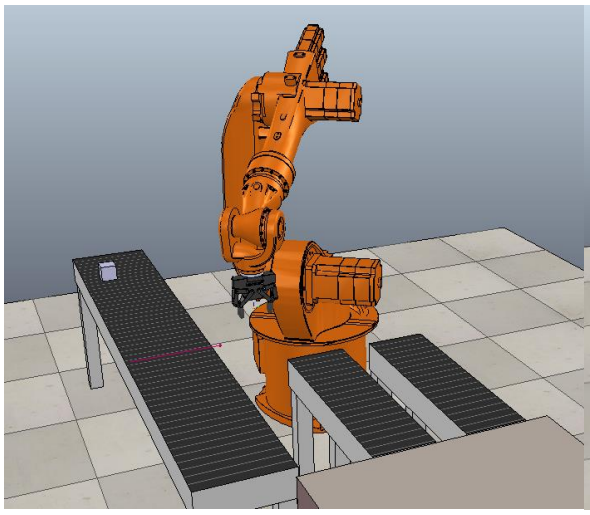
Obr. 48) Rozhodovací algoritmus pro Arduino

Naprogramovaná cesta gripperu manipulátoru znázorňuje schéma na Obr. 49. Následně gripper najede do polohy pro uchycení kostky, aktivuje gripper a kostku přesune na vedlejší pás. Volba vedlejšího pásu se odvíjí od velikosti kostky. Malé kostky odkládá na vybraný pás a velké kostky na pás druhý. Poté co kostky odloží na pás, najíždí koncový bod ramene robota na výchozí polohu, tedy nad senzor a čeká na příjezd další kostky.

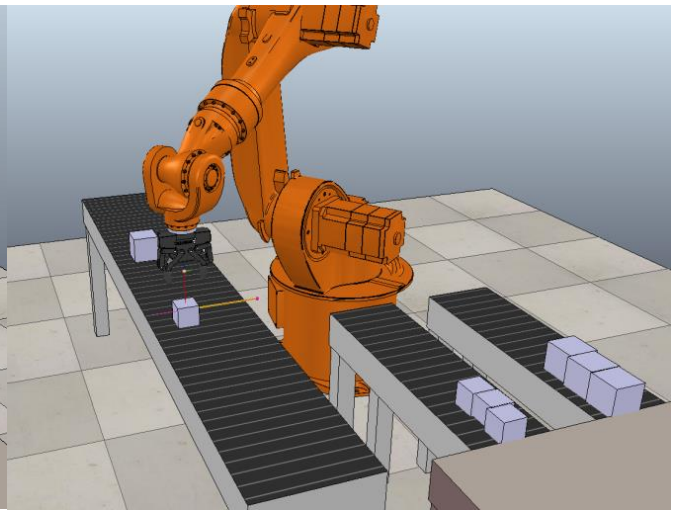


Obr. 49) Vývojový diagram pohybů manipulátoru

V simulačním prostředí je naprogramováno generování nekonečného počtu kostek. Liší se svojí velikostí podle, kterých je manipulátor třídí na další dopravníky. Prozatím je na konci dopravníku odkládací stůl, který kostky zastaví. Ty se na páse hromadí, viz Obr 51. Zde je místo pro vytvoření propojení s dalšími modely, které lze do prostředí přidávat.



Obr. 50) Počáteční stav



Obr. 51) Ukázka simulace

8 ZÁVĚR

Cílem mé diplomové práce bylo vytvoření rešerše v oblasti simulačního softwaru. Dále vybrat a popsat vhodný software pro simulaci. Hlavním úkolem bylo vytvoření virtuálního modelu, který je propojen s reálným hardwarem ve smyčce.

Nejprve je stručně popsána historie simulace a rešerše v oblasti simulačního software pro testování metodou HIL (Hardware in loop). V práci jsem se zaměřil jak na jednodušší robotické simulátory, které poskytují pouze omezené možnosti simulace vlivem menšího počtu modelů, tak i na ty větší a podstatně známější robotické simulátory jako jsou Webots nebo V-REP.

Pro diplomovou práci jsem si zvolil simulační prostředí V-REP. Tento simulační program převyšoval ostatní v mnoha aspektech. Je vhodný nejen pro simulaci stacionárních robotů, ale v budoucnu lze využít i pro mobilní roboty. Hlavním důvodem, proč jsem zvolil V-REP simulátor, je velmi intuitivní ovládání, uživatelské prostředí a uspořádání jednotlivých panelů a karet. Simulátor V-REP má velice dobrou dynamiku v chování jednotlivých modelů, což je důležité pro reálnost simulace. To zajišťují dva fyzikální enginy (Ode a Bullet). Zároveň poskytuje velmi dobrou technickou podporou. Na internetových stránkách nebo fórech lze najít mnoho knihoven, modelů, ale i cenných rad a návodů pro vytvoření simulačních prostředí. Při výběru manipulátoru jsem se inspiroval na fórech a našel jsem velice vhodný manipulátor KUKA KR16, který využívám ve své diplomové práci.

Další část práce se zabývá propojením reálného hardwaru Arduino UNO s virtuálním prostředím V-REP. Pro komunikaci mezi programy jsem zvolil program Matlab. Arduino s Matlabem komunikuje přes COM port po sériové lince ve struktuře JSON. Arduino obdrží informace o poloze koncového bodu robota a vyhodnotí další akci manipulátoru, které odešle do Matlabu. Matlab informace rozbalí ze struktury JSON a odešle do simulátoru V-REP, s kterým komunikuje přes API server. Jakmile dojde ke změně polohy, případně senzor zachytí nějakou jakoukoliv informaci, odešle ji do Matlabu, který ji zpracuje a zabalí do struktury JSON. Poté odešle zabalená data přes sériovou linku do Arduina. Tento proces programů představuje nekonečnou smyčku.

Komunikaci bylo třeba otestovat, a proto jsem vytvořil dvě vzorové úlohy. První se zabývá ovládáním manipulátoru pomocí joysticku PS2. Pro tuto úlohu bylo důležité správně načíst hodnoty ze vstupů Arduina a následně zpracovat signál z joysticku. Zpracované informace o výsledné poloze manipulátoru (robotu) jsou odeslány do simulačního prostředí. Jedná se o informace, které popisují souřadnice pohybů manipulátoru. Vzhledem k tom, že joystickem můžeme ovládat manipulátor pouze ve dvou osách, bylo nutné do obvodu připojit tlačítko, které aktivuje gripper robota. O nastavení výšky robota se stará joystick. Po stisknutí joysticku robot sjede ze své výchozí pozice, aby mohl uchopit kostku na dopravníkovém pásu. Kostky jsou generované v simulačním prostředí. Jejich počet není omezen.

Druhá úloha potvrzuje zapojení hardwaru do smyčky. Manipulátor pracuje zcela samostatně a to na základě informací, které si vyměňuje společně s hardwarem. Úloha obsahuje dopravník, po kterém se posouvají kostky. Jakmile dosáhnou polohy senzoru, dostane manipulátor z Arduina informaci, jestli se jedná o velkou, popřípadě malou kostku.

Velikost kostky určuje, jak má robot s danou kostkou dále pracovat, jelikož jsou tříděny podle velikosti na jednotlivé dopravníkové pásy.

Na základě vytvořených úloh jsem otestoval komunikaci mezi virtuálním prostředím a reálným hardwarem, což bylo hlavním cílem mé diplomové práce. Vytvořené propojení lze dále modifikovat podle potřeby. Do virtuálního prostředí je možné přidat širokou škálu objektů a zapojit je do simulované úlohy. Dále je možnost kompletního ovládání manipulátoru, což by znamenalo do nepájivého pole přidat několik komponentů, např. joysticků nebo tlačítek, nebo by mohla být přidaná zpětná vazba ve stylu rozsvícení LEDky.

Problematika komunikace a samotné simulace HIL nabízí další možnosti testování. V budoucnu ji lze doplnit o další modely, které by byly zapojeny do simulace. Dalším řešením by mohlo být komplexnější řízení manipulátoru. Hardware by mohl ovládat nejenom koncový bod manipulátoru, ale jeho všechny ramena. Pro tuto aplikaci by bylo nutné navýšit výkon PC. Závěrem mohu konstatovat, že veškeré cíle mé diplomové práce se mi podařilo naplnit i přes to, že jsem v prostředí V-REP a Arduinem pracoval poprvé.

9 ZDROJE

[1] Buffon's Needle Experiment. *http://web.mit.edu/* [online]. [cit. 2021-5-20]. Dostupné z: http://web.mit.edu/urban_or_book/www/book/chapter3/3.3.1.html

[2] Buffon-Laplace Needle Problem. *https://mathworld.wolfram.com/* [online]. [cit. 2021-5-20]. Dostupné z: https://mathworld.wolfram.com/Buffon-LaplaceNeedleProblem.html

[3] FABIAN, František; KLUIBER, Zdeněk. Praha: PROSPEKTRUM s.r.o., 1998. ISBN 80-7175-058-1. Kapitola 1.2, s. 13. (česky)

[4] FABIAN, František; KLUIBER, Zdeněk. Praha: PROSPEKTRUM s.r.o., 1998. ISBN 80-7175-058-1. Kapitola 1.2, s. 14. (česky)

[5] A brief history of simulation. https://www.researchgate.net/ [online]. [cit. 2021-5-20]. Dostupné z: https://www.researchgate.net/publication/224123869_A_brief_history_of_simulation

[6] ENIAC is formally announced, February 15, 1946. https://www.edn.com [online]. [cit. 2021-5-20]. Dostupné z: https://www.edn.com/eniac-is-formally-announced-february-15-1946/

[7] [ŠŤASTNÝ, Martin. Modelování a simulace robotických aplikací [online]. Brno, 2015 [cit. 2021-5-18]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/83779. Diplomová práce.]

[8] Coppelia Robotics. V-REP home. https://www.coppeliarobotics.com [online]. 2013 [cit. 2021-5-20]. Dostupné z: https://www.coppeliarobotics.com/

[9] Coppelia Robotics. Dynamics. https://www.coppeliarobotics.com [online]. [cit. 2021-5-20]. Dostupné z: https://www.coppeliarobotics.com/helpFiles/en/dynamicsModule.htm

[10] Coppelia Robotics. Main Features. https://www.coppeliarobotics.com [online]. 2013 [cit. 2021-5-20]. Dostupné z: https://www.coppeliarobotics.com/features

[11] Webots home. https://www.cyberbotics.com [online]. [cit. 2021-5-20]. Dostupné z: https://www.cyberbotics.com/#cyberbotics

[12] Webots features. https://www.cyberbotics.com [online]. [cit. 2021-5-20]. Dostupné z: https://www.cyberbotics.com/#features

[13] Gazebo features. http://gazebosim.org [online]. [cit. 2021-5-20]. Dostupné z: http://gazebosim.org/#features

[14] ŠŤASTNÝ, Martin. SW pro simulaci autonomních robotu [online]. Brno, 2013 [cit. 2021-5-20]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/64625. Bakalářská práce.

[15] Spiderland. The breve Simulation Environment [online]. 2006 [cit. 2021-5-20]. Dostupné z: http://spiderlang.org/

[16] Breve Simulation Environment 2.7.2. Https://macintoshgarden.org [online]. [cit. 2021-5-20]. Dostupné z: https://macintoshgarden.org/apps/breve-simulation-environment-272

[17] BENKO, B Hardware in the loop simulácia s využitím xPC target toolboxu. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 90 s. Vedoucí bakalářské práce Ing. Stanislav Klusáček

[18] Rapid development, rt simulace a modelovani. https://www.rexcontrols.cz [online]. [cit. 2021-5-20]. Dostupné z: https://www.rexcontrols.cz/rapid-development-rt-simulace-modelovani

[19] SHOKRY H., Hinchey M. „*Model-Based Verification of Embedded Software*“, Computer, vol.42, no4, pp. 53-59, Apr. 2009, doi:10.1109/MC.200.125

[20] Different parts of HIL simulation. *https://www.researchgate.net* [online]. [cit. 2021-5-20]. Dostupné z: https://www.researchgate.net/figure/Different-parts-of-HIL-simulation_fig1_327017114

[21] Arduino. https://www.arduino.cc/ [online]. [cit. 2021-5-20]. Dostupné z: https://www.arduino.cc/en/Main/Products

[22] BARTONĚK, Jan. *Programování Arduino pomocí Matlab/Simulink* [online]. Brno, 2020 [cit. 2021-5-20]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=42153. Bakalářská práce.

[23] Arduino UNO. *https://www.arduino.cc/* [online]. [cit. 2021-5-20]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[24] Joystick ps2. *https://www.hwkitchen.cz* [online]. [cit. 2021-5-20]. Dostupné z: https://www.hwkitchen.cz/ky-023-joystick-modul-ps2-s-tlacitkem/

[25] Nepájivé kontaktní pole. https://www.laskarduino.cz [online]. [cit. 2021-5-20]. Dostupné z: https://www.laskarduino.cz/nepajive-kontaktni-pole-830-pinu--bile/

[26] Ovládací prvky. *https://www.laskarduino.cz* [online]. [cit. 2021-5-20]. Dostupné z: https://www.laskarduino.cz/ovladaci-prvky/

10 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

10.1 Seznam zkratek

HIL	Hardware in the loop
V-REP	Virtual Robot Experimentation Platform
PS2	Playstation 2
SW	Software
DAQ	Data acquisition
GSP	General simulation program
API	Application programming interface
GUI	Graphical user interface
SDF	Simulation description format
MIL	Model in the loop
SIL	Software in the loop
PIL	Processor in the loop
ECU	Electronic control unit
A/D	Analog/digital
MBD	Model based design

10.2 Seznam tabulek

TAB 1) POROVNÁNÍ SOFTWARŮ 27
TAB 2) PARAMETRY MIKROKONTROLERU UNO [23] 38

10.3 Seznam obrázků

OBR. 1) JEHLOVÝ EXPERIMENT [2] 17
OBR. 2) JEHLOVÝ EXPERIMENT [1] 18
OBR. 3) ENIAC PC [6] 19
OBR. 4) SIMULAČNÍ PROSTŘEDÍ V-REP 22
OBR. 5) OVLÁDACÍ PANEL Č. 1 22
OBR. 6) OVLÁDACÍ PANEL Č. 2 22
OBR. 7) OVLÁDACÍ PANEL Č. 3 22
OBR. 8) OVLÁDACÍ PANEL Č. 4 23
OBR. 9) HIERARCHIE SCÉNY 23
OBR. 10) SIMULAČNÍ PROSTŘEDÍ WEBOTS [7] 24
OBR. 11) SIMULAČNÍ PROSTŘEDÍ GAZEBO [7] 25
OBR. 12) GAZEBO - KNIHOVNA OBJEKTŮ/MODELŮ 25
OBR. 13) SIMULAČNÍ PROSTŘEDÍ BREVE [16] 26
OBR. 14) MIL SCHÉMA 29
OBR. 15) SIL SCHÉMA 30
OBR. 16) HIL SCHÉMA 31
OBR. 17) PŘÍKLAD UŽITÍ HIL SIMULACE [20] 33
OBR. 18) LOGO ARDUINO OPEN-SOURCE COMMUNITY [21] 37
OBR. 19) ARDUINO UNO 38
OBR. 20) JOYSTICK PS2 [24] 39
OBR. 21) JOYSTICK PS2 – PINY [24] 39
OBR. 22) NEPÁJIVÉ KONTAKTNÍ POLE 830 PINŮ [25] 39
OBR. 23) KOMPONENTY ARDUINO [26] 40
OBR. 24) PROPOJENÍ BODŮ PŘES INVERZNÍ KINEMATIKU 40
OBR. 25) SCHÉMA KOMUNIKACE 41
OBR. 26) COM PORT 41
OBR. 27) NASTAVENÍ ARDUINA 42
OBR. 28) NAČTENÍ HODNOT ZE SIMULAČNÍHO PROSTŘEDÍ 42
OBR. 29) ČÁST KÓDU Z FUNKCE „MOVEL“ 43
OBR. 30) ČÁST KÓDU Z FUNKCE „GRIPPER“ 43

OBR. 31) PŘIPOJENÍ API K MATLABU 43
OBR. 32) OTEVŘENÍ COM PORTU V MATLABU 44
OBR. 33) NAČTENÍ PROMĚNNÝCH DO STRUKTURY „TEST“ 44
OBR. 34) KÓDOVÁNÍ DO FORMÁTU JSON 44
OBR. 35) STRING „MATLAB“ V ARDUINU 45
OBR. 36) ČTENÍ DAT ZE SÉRIOVÉ KOMUNIKACE PRO ARDUINO 45
OBR. 37) ROZHODOVACÍ PODMÍNKA V ARDUINU 46
OBR. 38) ZPRACOVÁNÍ DAT V ARDUINU 46
OBR. 39) ODESLÁNÍ DAT NA COM PORT 46
OBR. 40) DEKÓDOVÁNÍ STRUKTURY JSON V MATLABU 46
OBR. 41) SCHÉMA ZAPOJENÍ - ARDUINO UNO 47
OBR. 42) ZAPOJENÍ KOMPONENTŮ DO ARDUINA UNO 47
OBR. 43) SCHÉMA SPUŠTĚNÍ SIMULACE 47
OBR. 44) SPUŠTĚNÍ SIMULACE 48
OBR. 45) POČÁTEČNÍ STAV 48
OBR. 46) UKÁZKA SIMULACE 48
OBR. 47) UKÁZKA SIMULAČNÍ SCÉNY 49
OBR. 48) ROZHODOVACÍ ALGORITMUS PRO ARDUINO 49
OBR. 49) VÝVOJOVÝ DIAGRAM POHYBŮ MANIPULÁTORU 50
OBR. 50) POČÁTEČNÍ STAV 50
OBR. 51) UKÁZKA SIMULACE 50