Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Porovnání a měření šestiosého robota v modelovém a reálném prostředí

Diplomová práce

Bc. Václav Kocar

Školitel: Ing. Ladislav Ptáček, Ph.D.

České Budějovice 2023

Obsah

Anotace

Bibliografické údaje

Kocar, V., 2023: Porovnání a měření šestiosého robota v modelovém a reálném prostředí. [Comparison and measurement of a six-axis robot in a model and real environment, Mgr. Thesis, in Czech.] – 87 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Diplomová práce se zabývá srovnáním pohybu reálného průmyslového robota s pohybem v simulaci. Součástí práce je i navržení a sestrojení měřícího zařízení, které pohyb na reálném robotu zaznamená. Z dat jsou následně zrekonstruovány trajektorie porovnávané ve dvou analýzách, polohové a časové. Na základě těchto dvou analýz jsou navrženy optimalizace pro pohyb robotu.

Annotation

The master thesis compares the motion and simulation of a real industrial robot. Designing and creating motion-measuring devices is also a part of the thesis. We reconstructed the trajectories from the data using positional and time analyses. Finally, we suggest a robot movement optimization based on both analyses.

Prohlášení

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne 11.4.2023 Bc. Václav Kocar

Poděkování

Tímto bych rád poděkoval vedoucímu práce Ing. Ladislavu Ptáčkovi, Ph.D. za odborné vedení a vždy vstřícný přístup. Poděkování patří též mému školiteli Ing. Petru Falcovi i firmě Kuboušek s.r.o. za vytvoření podmínek pro měření a práci na průmyslových robotech. Také děkuji mé rodině a přítelkyni za podporu.

1 Úvod

Motivací k této práci pro mě byla moje několikaletá zkušenost se simulačními programy a roboty ABB v reálném průmyslu. V zaměstnání se věnuji implementaci robotů do automatizované výroby a školení personálu pro jejich obsluhu a programování.

Při návrhu automatizovaného řešení výroby se často využívá simulačních programů, které ovšem mají jisté limity a nepřesnosti při simulování reálné stanice. Povšimnul jsem si, že tyto rozdíly vůči realitě nejsou nikde v dostupných dokumentacích a návodech popsány. Nejsou nijak parametrizované a zobecněné, aby s nimi bylo možno počítat a popřípadě kompenzovat jejich vliv.

Pro účely mé práce jsem zvolil dva základní parametry pro simulovaný a reálný pohyb robotu, které budu porovnávat. Na základě těchto zjištění bude navržena doporučená optimalizace. Prvním porovnávaným parametrem bude poloha robotu v simulaci a v reálném prostředí, kde bude provedeno srovnání přesnosti dosažených poloh pro oba případy. Druhým parametrem bude čas provedení jednotlivých pohybů a porovnání případných rozdílů se zjištěním trendu.

2 Cíl práce

Cílem mé práce je namodelovat reálné prostředí pro šestiosého robota ABB s využitím dedikovaného software. Na vybraných parametrech budu porovnávat chování v reálném a namodelovaném prostředí.

U reálného robota provedu komplexní měření zrychlení všech ramen v rozdílných podmínkách. K tomuto měření musím navrhnout a sestrojit měřící zařízení. Pro jeho konstrukci jsem se rozhodl využít platformy Arduino. Ze získaných měření bude vypočítána trajektorie pohybu. Získaná data pak využiji k optimalizaci pohybu robotu.

Jak je uvedeno v zadání mé práce, hlavní porovnávaný parametr bude doba cyklu, který je nejdůležitější parametr pro sériovou výrobu. Lze předpokládat, že reálný robot se bude opožďovat oproti simulaci.

3 Seznam použitých zkratek

Zkratka	**Celý název**	**Popis**
TCP	Tool centre point	Specifický bod nástroje robotu
MUX	Multiplexor	Digitální přepínač
SDA	Serial Data Line	Datová linka
SCL	Serial Clock Line	Časová linka
UTP	Unshielded Twisted Pair	Standard pro síťové připojení
MPU	Motion Processing Unit	Jednotka pro záznam pohybu
I/O	Input/Output	Vstupně – výstupní
ČSN	České Technické Normy	České technické normy
PLC	Programmable Logic Controller	Programovatelný logický automat
ICSP	In-Circuit Serial Programming	Způsob programování mikrokontrolérů
ABB	Asea Brown Boveri	Značka průmyslových robotů
IMM	Injection Moulding Machine	Vstřikovací stroj
ISO	International Organization for Standardization	Mezinárodní organizace pro normalizaci

4 Teoretická Část

4.1 Šestiosý robot

Běžně užívaný název „robot“ použitý Karlem Čapkem již v roce 1920, je sice správný, ale velmi obecný. V běžné technické praxi se využívá označení spíše průmyslový robot.

Definic takového robotu existuje velmi mnoho a záleží na tom, jak moc obecná či specifická taková definice je, například:

a) (Nof, 1999) v 2. vydání své knihy „Handbook of Industrial Robots“ uvádí, že průmyslový robot je mechanické zařízení, které může být naprogramováno pro vykonávání různých úkolů manipulačních a pohybových, při automatickém řízení.

b) Podle (Havel, 1980) je průmyslový robot automatické ústrojí, který obsahuje a zvládá následující schopnosti (ve větší či menší míře) - manipulační způsobilost, automatická činnost, snadná změna programu, univerzálnost, zpětné vazby, prostorová soustředěnost.

c) Dle ČSN ISO 8373 je průmyslový robot definován jako automaticky řízený reprogramovatelný, víceúčelový manipulátor programovatelný ve třech nebo více osách, který může být buď pevně umístěný nebo mobilní pro použití v průmyslových aplikacích s použitím automatů (‘Manipulační průmyslové roboty - Slovník’, 1998), (Skařupa, 2007).

Ovšem obecně se všechny definice shodují na tom, že se jedná o víceúčelové programovatelné manipulační zařízení, které se pohybuje ve třech a více osách (Skařupa, 2007).

V této práci se výhradně zaměřím na běžně užívaný a rozšířený průmyslový robot, který disponuje šesti pohyblivými osami, dále v práci referovaný jako „robot“.

4.1.1.1 Popis robotu

Pohyblivé osy se popisují vždy od základny. Značí se indexací velkými písmeny, jako je na *obrázku 1*, nebo čísly 1, 2, ...6 popřípadě označením J1, J2, …J6, kde „J“ zastupuje zkráceně v anglickém jazyce *Joint*, kloub. Použité rozsahy pro jednotlivé klouby jsou konkrétně pro robot **IRB 1600 značky ABB**. Pro ostatní roboty se tento rozsah může lišit.

(A) První kloub robotu zajišťuje rotaci jeho těla kolem základny s (obvyklým) rozsahem pohybu ±180°.

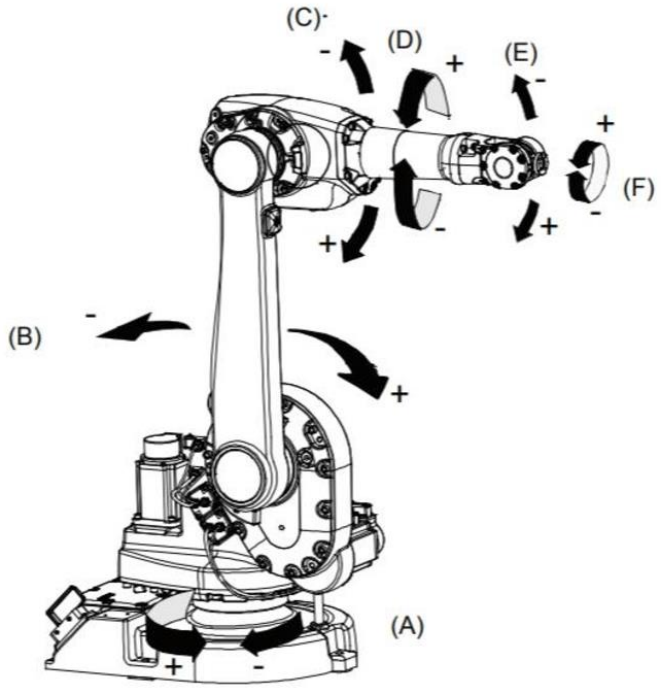
(B) Druhý kloub robotu též dle analogie s lidskou paží je nazýván rameno s rozsahem +110° -63°.

(C) Třetí kloub zvaný loket s rozsahem +55° - 235°.

(D) Čtvrtý kloub popisovaný jako předloktí s rozsahem +200° -200°.

(E) Pátý kloub pojmenovaný zápěstí s rozsahem +115° - 115°.

(F) Šestý kloub někdy také nazýván jako prsty s rozsahem +400° -400°, (Ratiu, Alexandru Rus and Balas, 2018).



Obrázek 1: Schéma průmyslového robotu IRB 1600 s označením kloubů, (Ratiu, Alexandru Rus and Balas, 2018).

Robot může plnit programové příkazy dvojím způsobem:

a) má příkaz nastavit osy do určitých úhlů a robot tak vykoná pohyb všemi osami zároveň v rychlostech a ve zrychleních, aby pohyb opět všech os skončil v ten samý časový okamžik. Takováto úloha se nazývá „Přímá kinematická úloha“. V robotické terminologii se jedná o „jointový pohyb“.

Přímá kinematická úloha je zobrazení z prostoru kloubových souřadnic do prostoru poloh chapadla (více viz. **4.1.1.2 Nástroj robota**). To znamená, že známe polohy všech (nebo jen některých) kloubů, a hledáme polohu chapadla ve světovém souřadnicovém systému (Smutný, 2004).

Matematické zobrazení:

$$\vec{q} \rightarrow \boldsymbol{T}(\vec{q}) \tag{1}$$

b) může robot plnit programové příkazy tak, že mu jsou zadány souřadnice ve světovém souřadnicovém systému a robot hledá vhodné nastavení úhlů kloubů tak, aby splnil tento požadavek. Jedná se o Inverzní kinematickou úlohu.

Inverzní kinematická úloha je zobrazení z prostoru poloh chapadla do prostoru kloubových souřadnic. To znamená polohu chapadla ve světovém souřadném systému a hledáme polohy všech kloubů (Smutný, 2004).

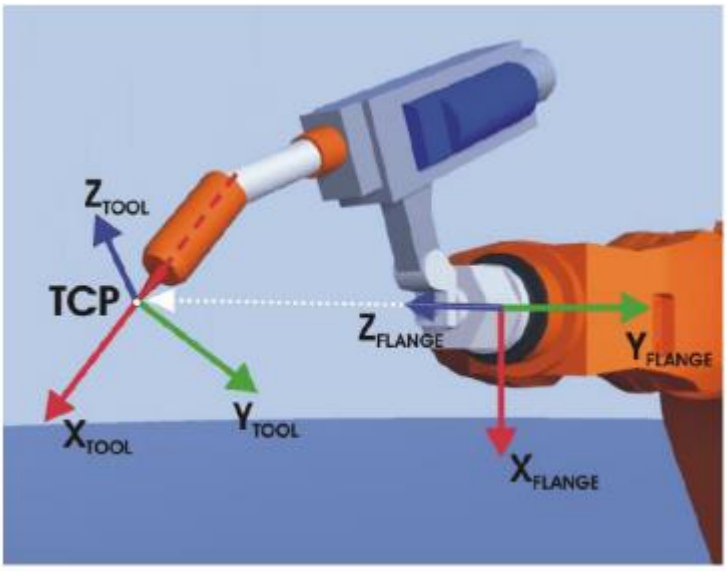
Matematické zobrazení:

$$\boldsymbol{T} \rightarrow \vec{\boldsymbol{q}}(\boldsymbol{T}) \tag{2}$$

4.1.1.2 Nástroj robota

Zařízení, které má robot připevněné na přírubě šesté osy a provádí s ním například uchopovací operace atd. se někdy také označuje jako chapadlo či end effector. Když robot nemá žádný nástroj definovaný, tak má nastaven výchozí. Výchozí nástroj je běžně označovaný jako Nástroj_0, Tool0 nebo Flange. Každý nástroj je definován specifickým bodem TCP a souřadným systémem nástroje.

TCP je zkratka *Tool Centre Point*. Nástroj_0 má TCP umístěn ve středu příruby 6. kloubu. Tímto bodem následně plní pohybové programové příkazy. To znamená, že když zadáme robotu příkaz pro pohyb na souřadnice x = 1000; y = 1000; z = 1000 mm, robot do těchto souřadnic v prostoru umístí právě TCP.



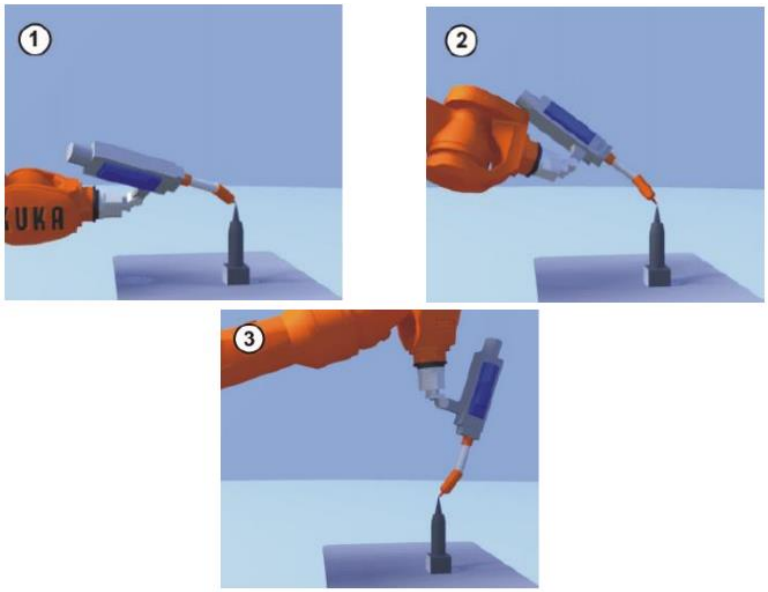
Obrázek 2: TCP, souřadný systém nástroje a souřadný systém Flange, (Jean-Louis Boimond, 2020).

Ovšem je možné si vytvořit nástroj vlastní.

Příklad:

Robot je osazen kleštěmi, které mají délku 200 mm, je tedy vhodné nastavit vlastní nástroj tak, aby měl TCP na špičce kleští.

Vytvoření nástroje se provádí například přímým zadáním relativních souřadnic nového bodu TCP od TCP Nástoje_0, nebo tzv metodou třech bodů, kdy je nutné robota přibližně nastavit v ručním režimu do třech různých poloh. Poté robot sám vypočte, kde se TCP nachází a uloží jeho relativní pozici.

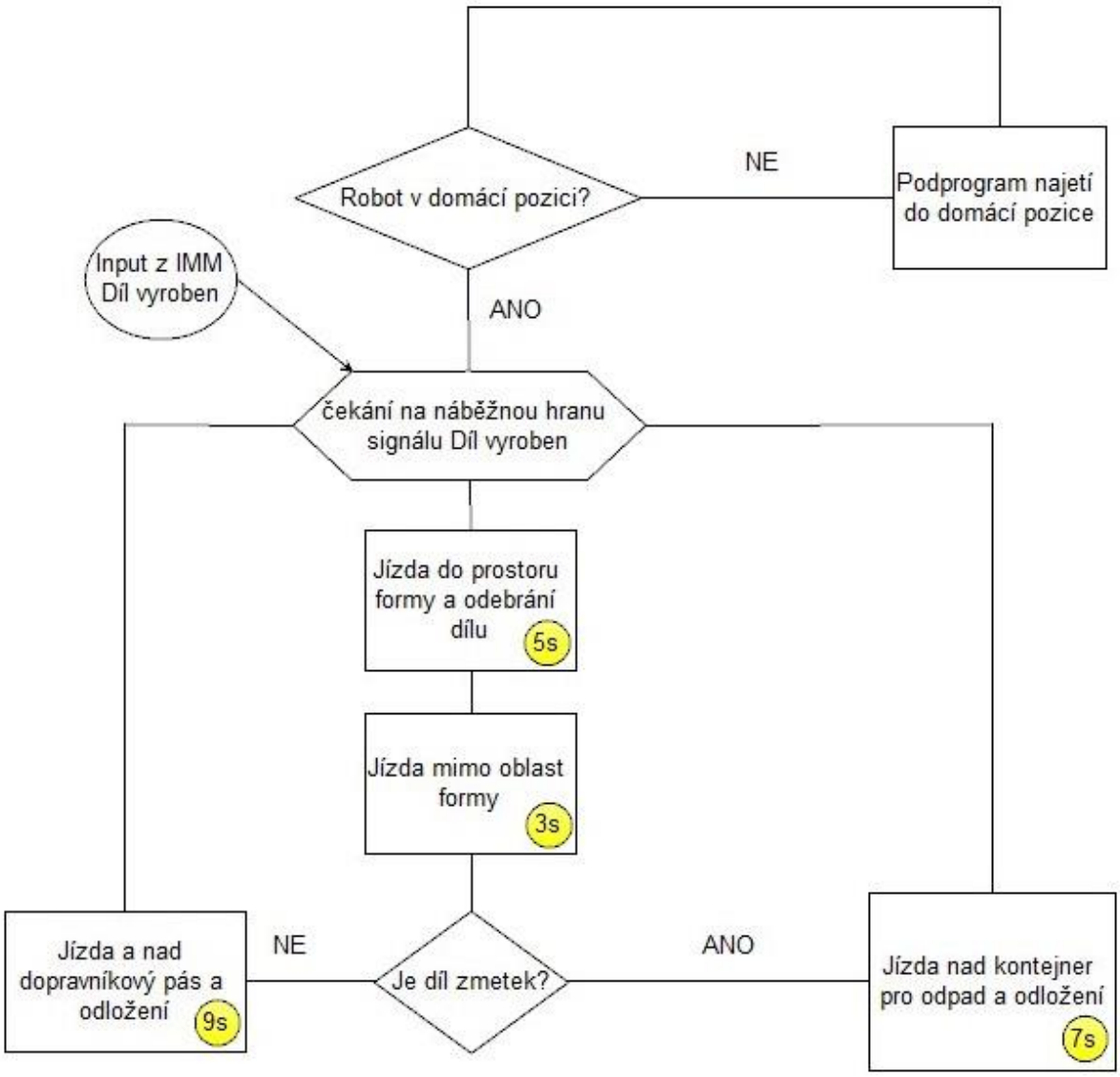


Obrázek 3: Metoda tří bodů pro učení nástroje, (Jean-Louis Boimond, 2020).

Nástroj také obsahuje vlastní souřadnicový systém, jako je vidět na obrázku č.2 Pro nástroj_0 je již předdefinovaný, ale když vytváříme nástroj vlastní, je třeba jej definovat. Opět se robot v ručním režimu přestaví do tří různých bodů a to 1. Bod pro počátek souřadného systému, 2. bod pro vytyčení kladného směru osy X (od počátku) a 3. bod pro vytyčení kladného směru osy Y. Třetí osu si robot vygeneruje sám, protože se v robotice výhradně používá pravotočivý souřadnicový systém.

4.1.1.3 Čas cyklu

Čas cyklu je označení doby, za kterou proběhne právě jeden celý cyklus robotu. Například při odebírání dílu ze vstřikovacího stroje cyklus vypadá zjednodušeně takto:



Vývojový diagram 1: Příklad automatizované výroby IMM s robotem (vlastní diagram)

U procesů v tzv. výrobní smyčce jsou uvedeny i orientační časy, za které robot danou operaci vykoná. Čas cyklu je tedy v případě, že je vyroben:

a) Zmetkový díl 5 s + 3 s + 7 s = **15 s**.
b) Dobrý díl 5 s + 3 s +7 s = **17 s.**

Z toho jasně vyplývá, že čas cyklu nemusí být vždy stejný. Aby se jednalo o optimalizovaný cyklus, je potřeba následující:

1) Robot by měl mít stejný, nebo kratší cyklus, jako cyklus IMM.

   Cyklus IMM, nebo také takt stroje, by robot neměl prodlužovat tím, že na něj bude IMM čekat (například z důvodu, že by odložení dílu na dopravníkový pás trvalo příliš dlouho).

2) Robot by měl být co nejrychlejší v prostoru formy (jízda pro díly, jeho odebrání a výjezd zpět).

   Tento čas je nejdůležitější z pohledu celé automatizace. Robot svým působením v prostoru formy v podstatě blokuje výrobu dalšího dílu a tento čas je většinou přímo úměrný počtu vyrobených dílů (tedy potažmo i potenciálnímu zisku). Je zde proto kladen velký nárok na optimalizaci a zároveň se při těchto pohybech robot pohybuje maximální přípustnou rychlostí. Ovšem se zvýšenou rychlostí robotu nesmí dojít ke snížení stability celého procesu.

   Příklad:

   a) Neoptimalizovaného pohybu:

   Robot jede do prostoru formy pro díl 70 % rychlostí a s dílem odjíždí mimo prostor formy 70 % rychlostí. Robot dopraví všechny díly na dopravníkový pás.

   b) Špatně optimalizovaného pohybu:

   Robot jede do prostoru formy pro díl 100 % rychlostí a s dílem odjíždí z prostoru formy 100 % rychlostí. Robotu upadne (například) každý osmý díl na zem.

   c) Správně optimalizovaného pohybu:

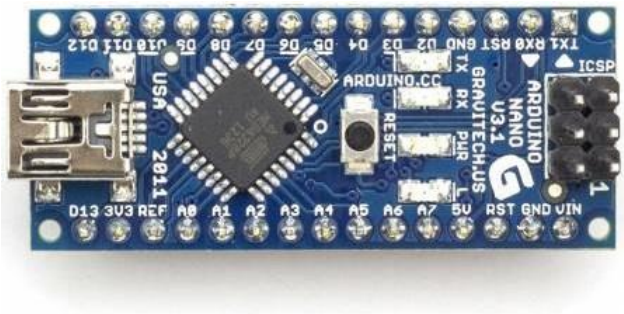
   Robot jede pro díl do prostoru formy 100 % rychlostí a s dílem odjíždí z prostoru formy 70 % rychlostí. Robot dopraví všechny díly na dopravníkový pás.

3) Robot by měl snížit rychlost mimo prostor formy, když je to možné.

   Robot by měl mimo prostor formy jezdit pomaleji, tak aby nebrzdil cyklus IMM, ale zároveň, aby sníženou rychlostí docílil co nejnižší spotřebu a opotřebení mechanických součástek.

Různé průmyslové aplikace mají různé požadavky a limitace, proto je výše zmíněný popis jen obecný a nemusí platit pro všechna použití průmyslových robotů. V našem případě byl již od začátku zvolen modelový případ odebírání dílu ze vstřikovacího stroje průmyslovým robotem.

4.2 Arduino



Obrázek 4: Vývojová deska Arduino Nano, (Ahmed Jasim, 2020).

„Arduino je otevřená vývojová platforma s grafickým vývojovým prostředím. Na rozdíl od Raspberry Pi není Arduino zamýšleno jako plnohodnotný stolní počítač. Řídící program je vyvíjen zvlášť (na stolním počítači) a do Arduina je poté nahrán. Uvnitř Arduina je pak spuštěn jen tento nahraný program, který typicky obsahuje smyčku, která se neustále opakuje (Arduino neustále zjišťuje stav svého okolí a na změny reaguje). Díky tomu má nízkou spotřebu (je možné napájení baterií) a hodí se například pro řízení dronů, robotů a podobně“ (‘Arduino’, 2017).

Desky Arduino obsahují 8bitové mikrokontrolery od firmy Atmel a množství dalších podpůrných obvodů. Oficiální vydání Arduina, používají čipy ATMega8, ATMega168, ATMega328, ATMega1280 a ATMega2560. Každá deska má většinu I/O pinů přístupných přes standardizované patice, do kterých se jednoduše připojují další obvody, kterým ve světě Arduina říká *Shieldy*. Na deskách bývá několik diod, resetovací tlačítko, konektory pro ICSP programování, napájecí konektor, oscilátor a obvod zprostředkovávající komunikaci USB (Maik Schmidt, 2015; Arduino Official Store, 2017; ‘Arduino’, 2017).

Před zvolením Arduina, jako vhodný mikrokontroler pro tento projekt, jsem analyzoval klady a zápory této platformy.

Výhody Arduina (převzato z: Maik Schmidt, 2015).

- Relativně nízká cena a dostupnost v porovnání s komerčními či průmyslovými řešeními (například PLC systémy či mikroprocesorové desky typu AVRPLC16).
- Široká podpora komunity, kterou tvoří nadšenci, často i lidé z praxe s technickým vzděláním a několikaletými profesními zkušenostmi.
- Open-source projekt, volně dostupné IDE, schémata a diagramy.
- Snadná dostupnost informací, návodů, schémat a zdrojových kódů.
- Předchozí zkušenost s Arduinem v bakalářské práci (Kocar, 2019).

Nevýhody Arduina (převzato z: Maik Schmidt, 2015).

- Z pohledu vývoje komerčních produktů, které se mají distribuovat, udržovat a servisovat, není tato platforma příliš vhodná a to hlavně kvůli krátkému životnímu cyklu produktů. S příchodem nového modelu přijde stará deska za rok či dva o podporu a dostupnost na trhu.
- Podobná situace panuje i v případě shieldů, které přestanou být dostupné a přijdou o podporu. S tím souvisí i podpora knihoven a ovladačů, které převážně vytváří komunita.
- Komerční aplikace často vyžadují certifikaci ke splnění náročných požadavků. Certifikace je cenově a technicky náročný proces, který se u produktů s krátkým životním cyklem nevyplatí.
- Platforma nenabízí možnost šifrování protokolů, či jiných bezpečnostních mechanismů, na které jsou v komerční praxi kladeny vysoké nároky.
- Vhodné spíše pro prototypování, nikoliv pro výslednou sériovou výrobu.

Jelikož cílem mé práce je vytvořit jedno zařízení pro měření pohybu robotu (tedy *prototyp*) a nikoliv zařízení pro komerční užití, byla zvolena platforma Arduino. Pro potřeby této práce jí lze považovat za dostatečnou a cenově výhodnou. Ovšem pro případnou komerční produkci těchto měřicích zařízení, na základě výsledků této práce a

na základě zde získaných zkušeností, striktně doporučuji nutnost využít profesionální elektroniku.

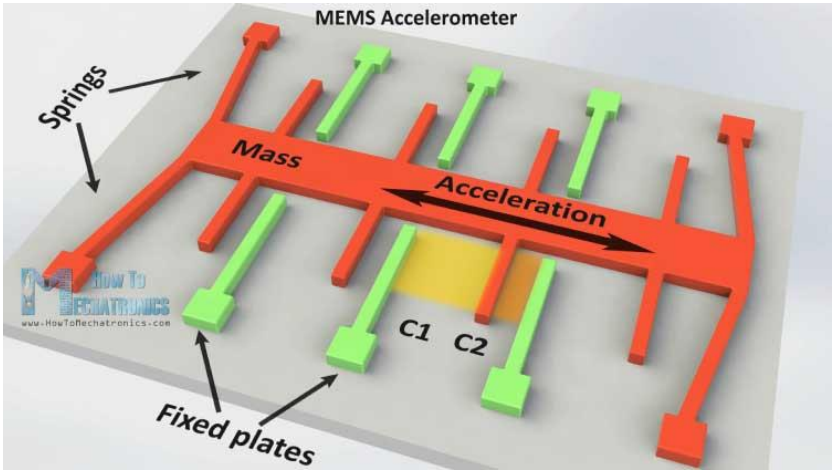
Pro naši aplikaci byl vybrán typ Arduino Nano, převážně díky svým minimálním rozměrům 43x 18 mm, jenž vyhovuje našim požadavkům. Zároveň také disponuje celkem 14 I/O digitálními a analogovými piny. Tento počet je více než dostačující.

4.3 Akcelerometry

Akcelerometry patří do skupiny MEMS (Mikro-Elektro-Mechanické Systémy). Jak je již patrné z názvu, tak tyto senzory využívají elektro-mechanických vlastností na mikroskopické úrovni. Akcelerometry jsou automatické senzory pro měření zrychlení, detekci a měření vibrací.

Většina dnes běžně používaných akcelerometrů funguje na principu změny kapacity v důsledku setrvačné síly. Změna kapacity je poté převedena na elektrickou veličinu. Ta je dále zpracována pomocí A/D převodníku na digitální signál a ukládaný do registrů. Hodnota signálu, uloženého v registrech, je tedy úměrná aktuálnímu zrychlení.

Na obrázku č. 5 je demonstrován princip akcelerometru. Vlivem pohybu senzoru vlevo, dojde ke změně vzájemné polohy oranžové a zelené části, tedy se změnila vzájemná kapacita, jelikož jde o dvě elektrody, mezi kterými je dielektrikem vzduch.



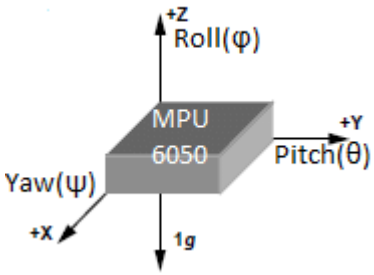
Obrázek 5: Schéma principu kapacitního akcelerometru MEMS, (ITNetwork.cz and Čápka, 2020).

Pro lepší představu reálné konstrukce viz obrázek *č.21: Akcelerometr pod elektronovým mikroskopem SEM (vlevo), Gyroskop pod elektronovým mikroskopem SEM (vpravo).*

Pro účely práce jsem vybral dva akcelerometry MPU6050 a ADXL345. Jejich popis je uveden v kapitole 5.3.3. Akcelerometry

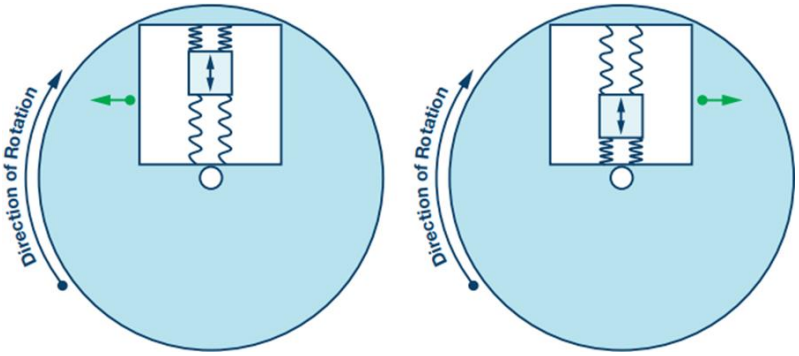
4.4 Gyroskop

Gyroskop je zařízení, které nám dává informaci o úhlové rychlosti [°/s]. Přepočtem tak lze zjistit aktuální polohu zařízení od začátku měření. Sklonu se v osách x, y, z se nejčastěji říká dle letecké terminologie X-Yaw, Y-Pitch, Z-Roll.



Obrázek 6: Schéma značení sklonu Yaw, Pich, Roll, (Abdullah Al Mamun and Mr. Fakir Mashuque Alamgir, 2017).

Princip získávání hodnot je založen na působení Coriolisovy síly na hmotné tělísko, které je ovšem oproti akcelerometru buzené.



Obrázek 7: schéma působení Coriolisovy síly na hmotné tělísko gyroskopu, (Jeff Watson, 2016)..

Působení Coriolisovy síly opět vyvolá změnu kapacity, jako u akcelerometru.

Z akcelerometru MPU6050 lze vyčíst buď aktuální úhlovou rychlost, či přímo sklon. Znát aktuální sklon je důležité, kvůli filtraci gravitačního zrychlení. Na akcelerometry působí samozřejmě i gravitační zrychlení. To nás ovšem při následné rekonstrukci pohybu robotu nezajímá. Je tedy nutné jej odečíst od každé vyčtené hodnoty.

Pokud by byl senzor stále ve stejné poloze a nenatáčel se kolem žádné z os a byl v rovině os x a y, bylo by možné odečíst hodnotu gravitačního zrychlení 9,81 m/s^2 jen od hodnoty zrychlení osy z. Pokud ovšem senzor mění sklon při pohybu, je potřeba zaznamenat i údaj o jeho aktuálním sklonu, aby bylo možné odečíst složku gravitačního zrychlení *gx, gy, gz [m/s²]* z příslušných naměřených zrychlení *ax, ay, az [m/s²]*.

4.5 Počítačová simulace

Simulaci (z latinského *simulō,* „napodobit“) můžeme pro potřeby řešení úloh technologického projektování chápat jako proces inženýrského modelování systému (= výrobní proces nebo výrobní systém). Podle jedné z definic je simulace výzkumnou metodou, jejíž podstata spočívá v nahrazení zkoumaného systému simulačním modelem, se kterým provádíme pokusy s cílem získat informace o původním zkoumaném systému (Volf, Beránek and Mikeš, 2010).

Výhody počítačové simulace pro roboty:

- Levnější řešení. (je levnější vyzkoušet nejprve v simulaci, zda robot dokáže projet určitou trajektorii, aniž by došlo ke kolizi s oplocením, než pořídit robot a poté zjistit, že dochází ke kolizi s oplocením).
- Je časově méně náročná na přípravu.
- Je bezpečná.
- Lze provést měření mnoha konfigurací.

Nevýhody počítačové simulace pro roboty:

- Potřeba kvalitního modelu.
- Obtížný popis fyzikálních interakcí.
- Simulační model zanedbává některé skutečnosti, které mohou být někdy stěžejní (např.: teplota okolí, vůle ozubení apod.).
- Pořizovací cena simulačního programu.

4.6 Numerická integrace

Pro výpočet polohy *r(t)* ze zrychlení *a(t)* je dle vztahu (25) v kapitole **5.6 Sběr a zpracování dat**, potřeba dvojnásobné integrace. V našem případě máme jen funkční hodnoty v bodech, a nikoliv předpis funkce, kterou bychom mohli integrovat. Proto je vhodné použít numerickou integraci, jež je běžně používaný nástroj pro výpočet přibližné hodnoty integrálu.

Při numerické integraci se provádí přibližné řešení určitého integrálu:

$$\int_a^b f(x)dx \tag{3}$$

kde $f(x)$ je spojitou funkcí v intervalu $\langle a, b\rangle$, a, b představují meze určitého integrálu.

Interval $\langle a, b\rangle$ se rozdělí na n stejně velkých intervalů

$$h = \frac{b-a}{n} \tag{4}$$

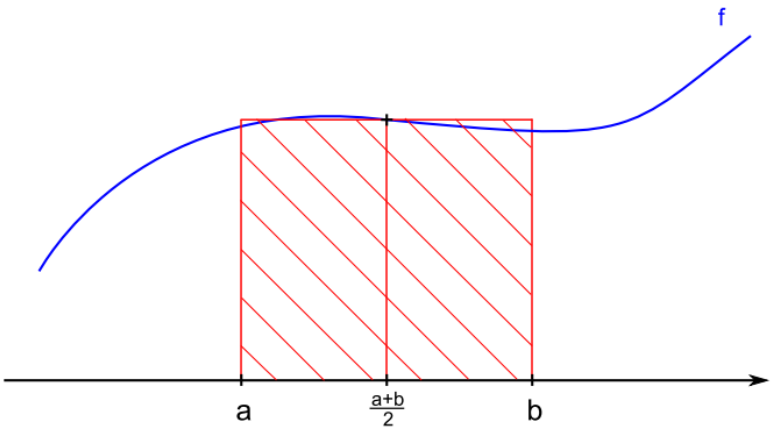
V každém sub intervalu se aproximuje integrovaná funkce $f(x)$ jednodušší interpolační nebo aproximační funkcí (polynomem stupně m) $\phi m(x)$:

$$\int_a^b f(x)dx = \int_a^b \phi m(x)\, dx \; + \; Rm(f) \tag{5}$$

kde $Rm(f)$ je chyba použité výpočetní metody.

1) **Obdélníková metoda**

   V případě použití obdélníkové metody numerické integrace se integrovaná funkce $f(x)$ aproximuje v každém ze subintervalů polynomem nultého stupně, tedy konstantní funkcí $\phi 0(x)$ = konst., (Krejsa, 2023).



Obrázek 8: Numerická integrace- Obdélníková metoda, (Vondrák and Pospíšil, 2011).

Výpočet numerické integrace:

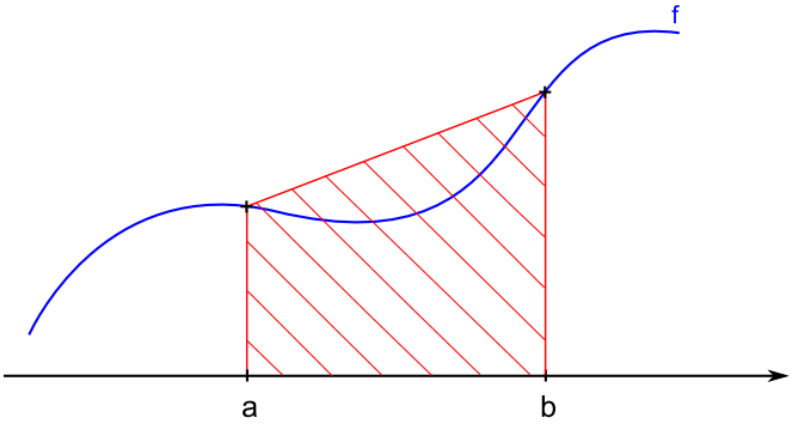
$$\int_a^b \phi m(x)\,dx \approx (b-a)\cdot f\left(\frac{a+b}{2}\right) = I_{Obd.} \tag{6}$$

Chyba této metody:

$$Rm(f) = \frac{1}{24}(b-a)^3 f''(\xi) \tag{7}$$

2) Lichoběžníková metoda

Pokud se k numerickému integrování použije lichoběžníková metoda numerické integrace, na jednotlivých subintervalech se integrovaná funkce $f(x)$ aproximuje polynomem prvního stupně, tedy lineární funkcí $\phi 1(x) = k \cdot x + q$ (Krejsa, 2023).



Obrázek 9: Numerická integrace- Lichoběžníková metoda, (Vondrák and Pospíšil, 2011).

Výpočet numerické integrace:

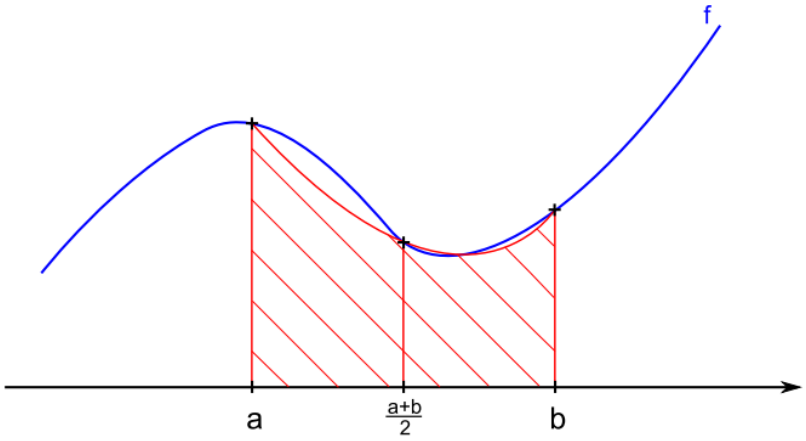
$$\int_a^b \phi m(x)\,dx = \frac{b-a}{2}\big(f(a)+f(b)\big) = I_{Lich.} \tag{8}$$

Chyba této metody:

$$Rm(f) = -\frac{1}{12}(b-a)^3 f''(\xi) \tag{9}$$

3) Simpsonova metoda

Zvolí-li se pro aproximaci funkce $f(x)$ na jednotlivých subintervalech polynomy druhého stupně, tedy kvadratické funkce $\phi 2(x) = a \cdot x\ 2 + b \cdot x + c$, provádí se numerické integrování Simpsonovou metodou numerické integrace. Počet subintervalů n přitom ale musí být sudý (Krejsa, 2023).



Obrázek 10: Numerická integrace- Simpsonova metoda, (Vondrák and Pospíšil, 2011).

Výpočet numerické integrace:

$$\int_a^b \phi m(x)\,dx \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right) = I_{Sim.} \tag{10}$$

Chyba této metody

$$Rm(f) = -\frac{1}{90}(b-a)^5 f^{IV}(\xi) \tag{11}$$

„Ve všech případech je ξ blíže neurčený bod z intervalu (a, b).“ (Kučera and Morávková, 2015).

Důležitým parametrem pro výběr numerické metody byl formát získaných dat. Jelikož očekáváme, že naměřená data budou diskrétní hodnoty zrychlení, musíme brát v potaz, že nebude možné vypočítat funkční hodnotu v polovině intervalu. To by bylo možné, jen kdyby byly diskrétní hodnoty interpolovány. Nebo, by musely být použity tři naměřené hodnoty pro výpočet jednoho subintervalu, který by byl dvojnásobný. Byla zvolena vhodnější metoda pro tuto aplikaci a to **lichoběžníková**. Tato metoda totiž neužívá funkční hodnoty ze středu intervalu, ale jen krajní.

5 Praktická část

5.1 Měření trajektorie v simulaci

Proto, aby bylo možné měřit trajektorii ze simulace, je zapotřebí takovou simulaci vytvořit a zaručit, že všechny nastavené parametry reálného robotu odpovídají parametrům v simulaci.

Pro vytvoření simulace jsem použil pracovní prostředí *Robotstudio*, které slouží k programování robotů a simulací firmy ABB. Vytvořil jsem stanici, která je podobná reálnému uspořádání pro výrobu plastových dílů v průmyslu. Na obrázku (č.11). jsou znázorněny prvky, ze kterých je simulace složena. Jedná se o:

A) Vstřikovací stroj

V praxi i v simulaci komunikuje s robotem pomocí komunikace Euromap 67, jež je komunikační standard robot ↔ vstřikovací stroj. Jedná se o sadu 24V I/O signálů, díky které dochází k posloupnosti úkonů obou strojů, tak aby byl zaručen plynulý provoz a stále byla hlídána bezpečnost. Vstřikovací stroj se ve zkratce označuje jako IMM, Injection Moulding Machine. IMM také pohybuje formou (světle modré zařízení uvnitř IMM), proto, aby došlo k tzv. „odformování“ plastového výrobku a aby robot tento vyrobený díl mohl odebrat a byla tak umožněna výroba dalšího dílu.

B) Rameno robotu

Pro simulaci byl použit robot značky ABB, model IRB 1600. Jeho primární úkol je odebírat vyrobené díly vstřikovacím strojem a odkládat je na dopravníkový pás C. Zároveň tento proces musí probíhat v optimalizovaném čase, tak aby nezpožďoval cyklus IMM. Zároveň se ale robot nesmí zbytečně opotřebovávat. Více v kapitole **4.1.1.3 Čas cyklu**.

C) Dopravníkový pás

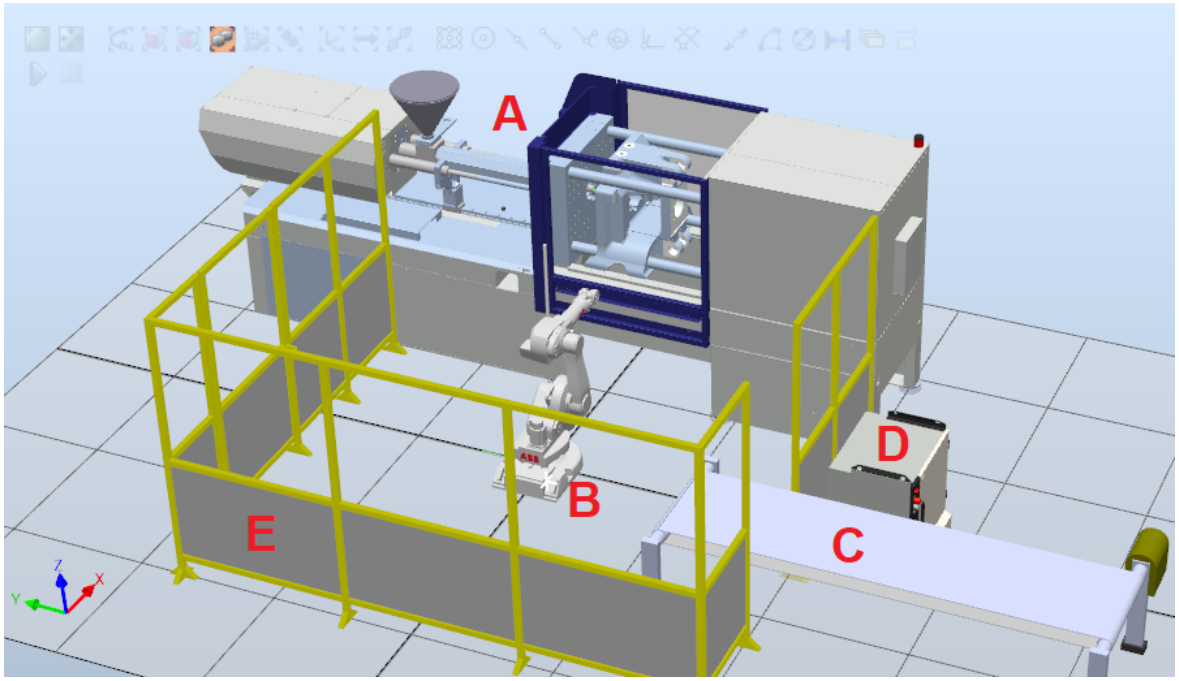
Elektro – mechanické zařízení sloužící k transferu odložených dílů robotem z výrobní buňky. Například do úložného boxu, či k obsluze, která vyrobený díl zkontroluje a uskladní jej.

D) Kontrolér robotu

Je řídící jednotka robotu. Je nedílnou součástí robotu a zprostředkovává veškeré řízení, napájení a výpočty pro rameno robotu B). Také hlídá veškerou bezpečnost výrobního procesu.

E) Oplocení

Slouží jako ochrana obsluhy robotu, aby nebylo možné dostat se k robotu během jeho provozu v automatickém režimu.



Obrázek 11: Model stanice pro vstřikování platu s robotem IRB 1600. Vytvořeno v programu Robotstudio 2022 (vlastní obrázek).

Aby byly zaručené stejné podmínky jak pro simulaci, tak pro reálný robot, musel jsem zajistit následující:

a) Jako první byl naprogramován, vyzkoušen a odlazen cyklus pro reálný robot.
b) Následně byla vytvořena jeho záloha, pomocí připojení notebooku síťovým UTP kabelem a tato záloha byla nahrána do Robotstudia a použita pro simulaci.

Tím je zaručeno, že simulace má totožné nastavení, jako reálný robot.

5.2 Měření trajektorie na reálném robotu

Aby bylo možné získat trajektorii reálného robotu a další potřebná data pro analýzu, bylo potřeba navrhnout a sestrojit měřící systém, který všechny potřebné informace bude schopný zaznamenat.

Jako hlavní parametry pro návrh takového zařízení jsem stanovil:

1) **Cena**

   Kladl jsem důraz na to, vytvořit levné zařízení, které bude schopno pracovat v reálném provozu, aby bylo možné například za nízké náklady osadit i více robotů v automatizační lince těmito zařízeními a sbírat data přímo z provozu.

2) **Kompatibilita**

   V dnešní době je většina hlav robotů (příruba na posledním kloubu robotu) vyráběna se standardizovaným vrtáním závitů, a to osm závitů symetricky rozložených na kružnici a jeden aretační trn. Zvolil jsem způsob uchycení jen na čtyři šrouby, aby osazení robotu měřícím zařízením bylo snadné a rychlé. Při předpokládané hmotnosti <0,5 kg lze upustit o plného uchycení osmi šrouby, které garantuje dle výrobce nosnost 10 kg (ABB Robotics, 2022).

3) **Modularita**

   Měřící zařízení jsem již od návrhu připravoval na budoucí vylepšení a případné změny. Proto je možné celé zařízení rozebrat a vyměnit například jen senzory za jiné, a to během několika minut. Také je možné mít více měřících zařízení, díky vhodné konstrukci supportu (více v kapitole Support). Je možné během minuty vyměnit celé měřící zařízení za jiné, protože je s mezikusem, který je připevněn na robot, a zároveň je aretován jen jedním aretačním šroubem.

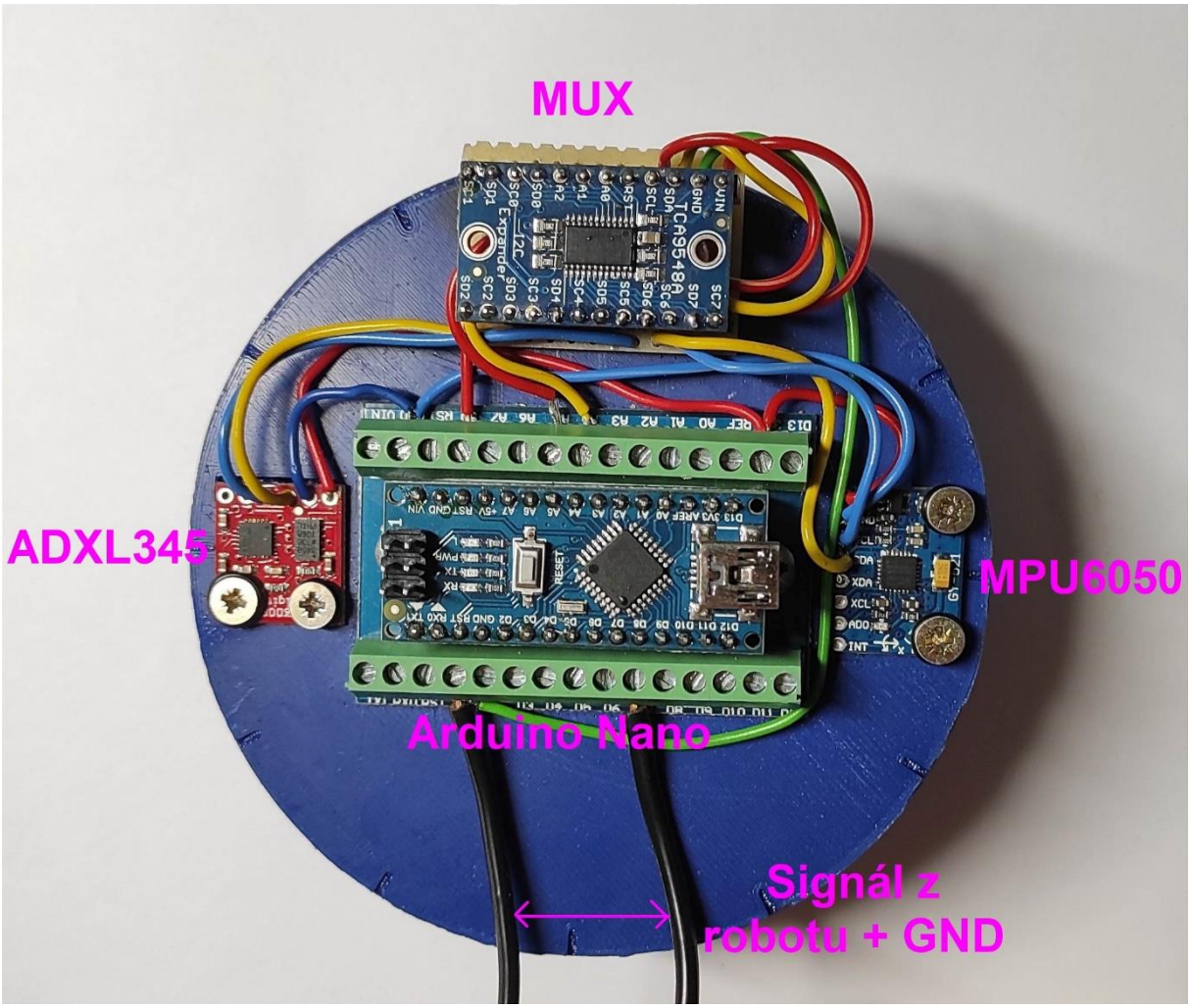
4) **Spolehlivost**

   Zařízení jsem v základu osadil dvěma různými senzory. Ovšem je připraveno na zapojení až osmi senzorů. Tyto senzory zajišťují nezávislý sběr dat. Výhodou je, že když jeden senzor vypoví službu, či je zatížen chybou, ať už

jakéhokoliv charakteru, a neposkytuje data, která by měl, pořád máme k dispozici data z ostatních senzorů.

Měřící zařízení bylo navrženo a sestrojeno. Jeho bližší popis a funkce jsou podrobně popsány v následujících kapitolách.

5.3 Měřící zařízení



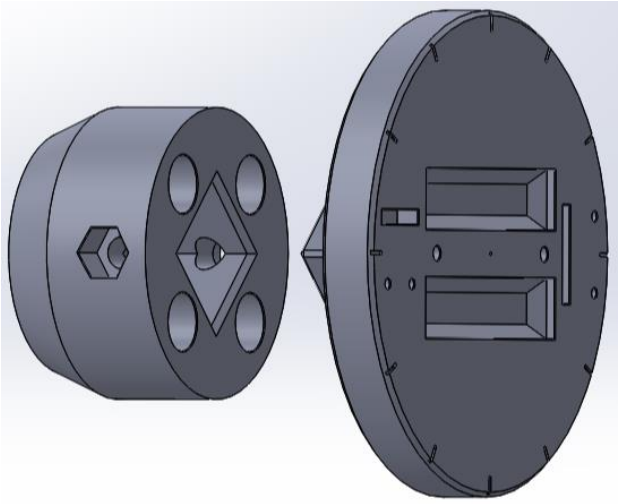
Obrázek 12: Měřící zařízení (vlastní fotografie).

5.3.1 Support

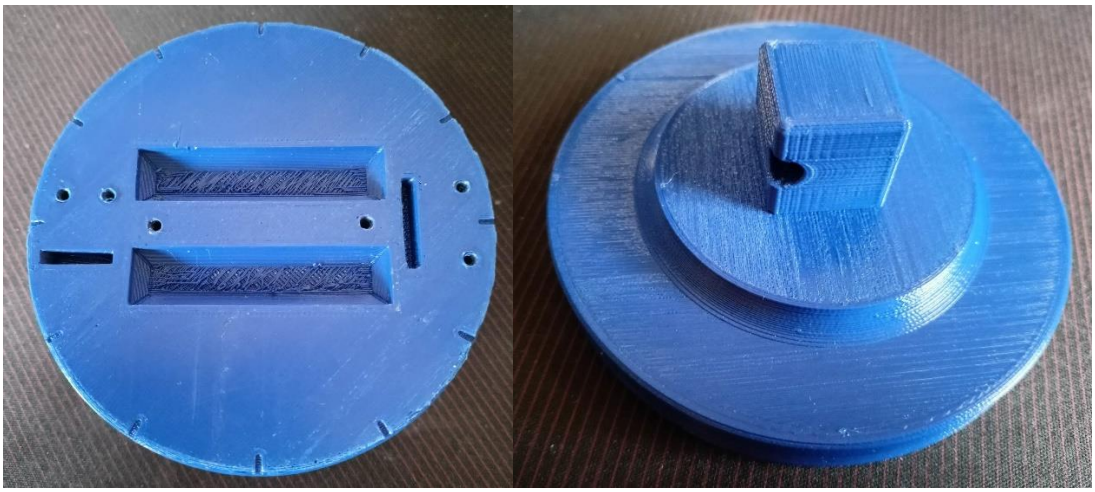
Pro potřebu měření reálného zrychlení robotu je zapotřebí umístit na *end effector* měřící zařízení, které bude zaznamenávat aktuální zrychlení, v a reálném čase jej odesílat do počítače, kde se data uloží a dále budou zpracována. Pro tento účel jsem musel vymyslet, namodelovat a vyrobit support, na kterém bude měřící elektronika upevněna.

Byl zvolen jeden z nejčastějších způsobů prototypování v automatizační praxi, a to 3D tisk. Tento způsob jsem zvolil na základě nízké pořizovací ceny a vysoké variability.

Support jsem namodeloval v programu Solid Works a následně zpracoval v programu pro 3D tisk *PrusaSlicer*, kde byly nastaveny parametry (např. hustota plnění, tloušťka vrstev, teplota zpracování apod.). Support se skládá ze dvou částí, které se do sebe zasunou a aretují pomocí šroubu. První část (obr. 13 vlevo) slouží k upevnění přímo na robot a druhá část (obr. 13 vpravo) k upevnění veškeré měřící a výpočetní elektroniky.



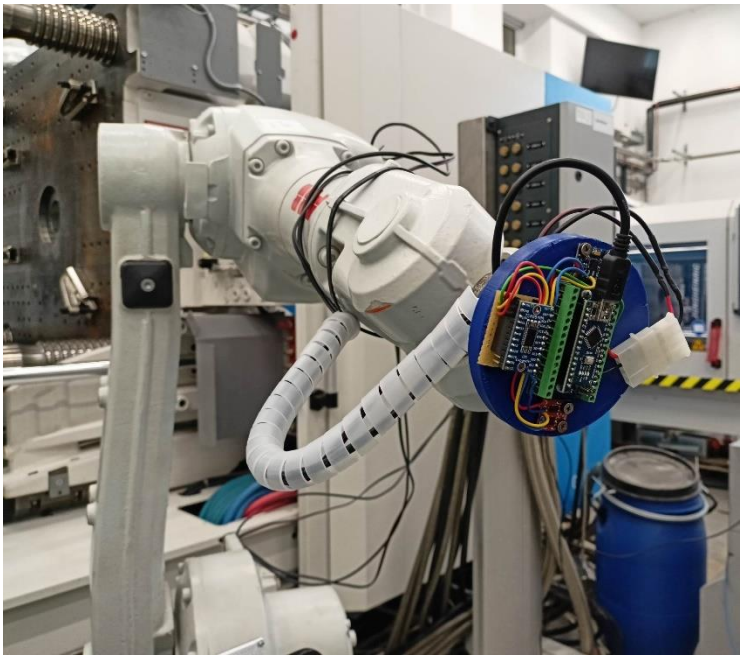
Obrázek 13: 3D model supporu vytvořený v programu Solidworks (vlastní obrázek).



Obrázek 14: Support, vlevo pohled shora, vpravo pohled ze spodu (vlastní fotografie).

Na obrázku 14 (vlevo) je vidět vybrání do povrchu, aby bylo možné elektroniku upevnit i po připájení vodičů, které na zadní straně vždy spolu s cínem, vytvoří nerovnost. Je tedy možné usadit všechny součástky bez vzájemného naklonění, což je důležité

hlavně pro uložení akcelerometrů. Dále je support po obvodu označen ryskami po 30°. Ty slouží pro lepší orientaci, až bude support usazen přímo na robot.



Obrázek 15: Robot osazený měřícím zařízením (vlastní fotografie).

5.3.2 Mikroprocesor

Pro získání dat ze senzoru je zapotřebí užití vhodné výpočetní techniky, která zaručí dostatečně rychlý a stabilní přenos dat ze senzoru a jejich následnou úpravu vhodnou pro export na sériovou linku (USB).

Bylo zvolen mikroprocesor ATmega328, osazen na vývojové desce značky Arduino, typ Nano. Tato konfigurace byla zvolena na základě předchozích zkušeností s touto vývojovou platformou při zpracovávání bakalářské práce (Kocar, 2019).

Z důvodu omezeného prostoru na supportu (a obecné tendence vytvořit co možná nejmenší měřící zařízení), jsem zvolil typ Arduino Nano, který disponuje téměř čtvrtinovými rozměry oproti originální desce UNO, užité v již zmíněné bakalářské práci. Jelikož se jedná o zařízení, které může sloužit i v reálné pracovní praxi, je možné, že na základě výsledků této práce dojde ke zlepšení tohoto zařízení a výměně některých méně kvalitních komponent. Proto byla, pro snadnější manipulaci a pro případnou budoucí výměnu čidel, užita svorka, do které se mikrokontroler zasadí a na jehož piny je nyní možné upevňovat vodiče pomocí šroubků.

5.3.3 Akcelerometry

Pro minimalizaci chyb, vnesených užitým hardwarem, byly použity dva akcelerometry různých značek a kvalit. Záměrem bylo využít alespoň dva akcelerometry, různých cenových kategorií. Porovnáním jejich hodnot resp. chyb, získat odpověď na otázku, jaký z použitých modelů je vhodnější pro konkrétní situace (měření časů, polohy, využití pro rychlé pohyby, pomalé pohyby, apod.). Výběr akcelerometrů byl podřízen zejména mými finančními možnostmi. Snahou bylo zakoupení ještě třetího akcelerometru typu MTI- 670 který stojí kolem 40 tisíc korun. Bohužel jej nakonec nebylo možné z finančních důvodů pořídit.

Prvním akcelerometrem byl zvolen MPU6050, GY521 dále jen MPU 6050. Je vybavený 3osým akcelerometrem pro měření zrychlení v rozlišení ±2 g, ±4 g, ±8 g, ±16 g, kde g je gravitační zrychlení a gyroskopem pro měření náklonu ±250, ±500, ±1000, ±2000 °/s. Informace o aktuálním náklonu budou důležitá pro následující zpracování dat. Také disponuje integrovaným šestnácti bitovým AD převodníkem. MPU6050 odesílá data po sběrnici I2C na adrese 0x68. Tento sensor stojí v řádu desítek korun. Tento velmi levný model jsem převážně využil ke srovnání výsledků s akcelerometrem vyšší kvality ADXL345.

Obrázek 16: Akcelerometry MPU6050(vlevo), ADXL345 Sparkfun (vpravo), (‘Akcelerometry MPU6050’, 2017; SparkFun Electronics, 2023)

Druhý akcelerometr byl zvolen vice než 10tinásobně dražší sensor značky Sparkfun osazený čipem ADXL345, který také disponuje 3osým akcelerometrem také s přednastavitelným rozlišením ±2 g, ±4 g, ±8 g, ±16 g a 16 ti bitovým AD převodníkem. ADXL345 komunikuje také na sběrnici I2c a má defaultní adresu 0x53.

5.3.4 Multiplexor

Multiplexor, dále jen MUX, je elektronický člen, kde je na základě řídících signálů převeden jeden ze vstupů na výstup. Byl zvolen MUX TCA9548A, který používá sběrnici I2C, stejně jako oba akcelerometry.

Kdyby byly použity jen sensory na adresách 0x68 a 0x53, nebylo by zapotřebí používat MUX, protože sensory s rozdílnou adresou mohou komunikovat po stejné sběrnici zároveň. Jak bylo již zmíněno výše, byla snaha o vytvoření nejuniverzálnějšího měřícího zařízení, které bude možno v budoucnosti upravovat a konfigurovat tak, aby bylo možné dosáhnout lepších výsledků. Jedná se o osmi kanálový přepínač, tedy je teoreticky možné připojit a vyčítat data až z osmi stejných senzorů.

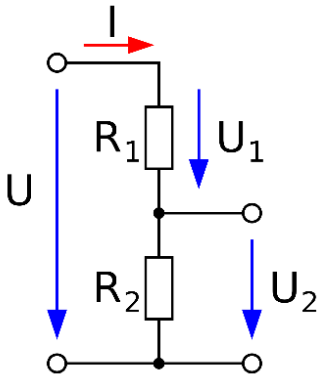
5.3.5 Signál z robotu

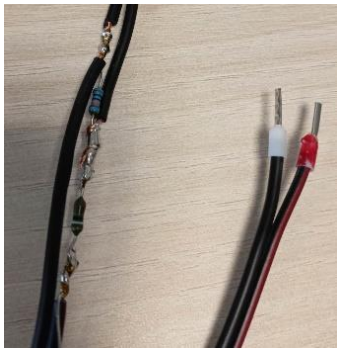
Pro časovou analýzu a lepší orientaci při práci se získanými daty byla zprostředkována komunikace mezi robotem a mikroprocesorem. Tato komunikace probíhá jednosměrně z robotu do mikroprocesoru, kam robot odesílá informaci o tom, zda se zrovna pohybuje, či projel určitým bodem trajektorie, nebo zda se nepohybuje. Tato informace je přenášena formou signálu, který robot generuje na své výstupní kartě na svorce "OUT 0". Tento signál byl nazván "*trigger*". Jeho stavy jsou uvedeny v tabulce č. 1.

Stav robotu	Stav signálu *„trigger"*
Robot se pohybuje	1
Pohybuje se nepohybuje	0
Robot projel definovaným bodem trajektorie	pulz 200 ms

Tabulka 1: Stav signálu "trigger" v závislosti na stavu robotu

Aby mohl mikroprocesor zpracovat signál z robotu, je zapotřebí upravit jeho analogovou hodnotu, která je pro průmyslové roboty standardně rovna 24 V. Mikroprocesor ovšem vyžaduje na vstupech maximálně vstupní napájení mikroprocesoru navýšené o hodnotu +0,5 V. Jelikož je Arduino napájeno pomocí USB z počítače, v našem případě to znamená maximálně +5,5 V. Je zapotřebí tedy snížit napětí z 24 V na 5 V. Jako nejjednodušší řešením jsem zvolil vytvoření napěťového děliče.





Obrázek 17: Schéma odporového děliče napětí (vlevo), (Pospíšil, 2020),Odporový dělič napětí (vpravo) (vlastní fotografie) .

Kde: U=24 V U_2=5 V U_1=19 V

Tato napětí a požadavek na proud do vstupu <40 mA (ideálně <20 mA), jsou požadavky pro konstrukci napěťového děliče. Maximální proud do vstupu Arduina je deklarován v *datasheetu.*

Dle vztahu (12) hledáme poměr R_1 ku R_2

$$\frac{U_1}{U_2} = \frac{R_1}{R_2} \tag{12}$$

$$\frac{19}{5} = \frac{R_1}{R_2} \tag{13}$$

Poměr R_1 ku R_2 je roven 3,8. Hledáme tedy rezistory vyhovující tomuto poměru. Vybrány byly rezistory ze standardní řady E24 1% tolerance přesnosti a to R_2=100 kΩ a R_1=390 kΩ, jejichž poměr je 3,9. Dopočteme tedy přesné hodnoty, které získáme užitím těchto rezistorů.

Dle vztahu (14) zjistíme U_2

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} \tag{14}$$

$$U_2 = 24V \cdot \frac{100k\Omega}{390k\Omega + 100k\Omega} \quad (15)$$

$$U_2 = 4{,}898V \quad (16)$$

Z toho vyplývá, že

$$U = U_1 + U_2 \quad (17)$$

$$U - U_2 = U_1 \quad (18)$$

$$24 - 4{,}898 = 19{,}102V \quad (19)$$

U_1 je rovno 19,102V

Při použití rezistorů 100 kΩ a 390 kΩ tedy dosáhneme napětí 4,898V na vstupu, což je dostačující, protože hranice pro detekci signálu LOW (tedy logická 0) je 0,3 · VCC a hranice pro detekci signálu HIGH (logická 1) je 0,6 · VCC.

Jak bylo již zmíněno výše, napájíme Arduino napájecím napětím z počítače, tedy VCC= 5 V. Potom tedy hranice pro detekci LOW = 0,3 · 5 V = 1,5V a hranice pro detekci HIGH 0,6 · 5 V=3 V.

Z toho vyplývá, že mohu užít:

Signál s velikostí 0–1,5V pro LOW.

Signál s velikostí 3–5,5V pro HIGH.

Napětí U_2 = 4,898 je tedy v rozmezí napětí pro signál HIGH. Nyní zbývá vypočítat maximální možný proud do vstupu a zjistit, zda nepřesáhne limitní hodnotu 40 mA.

Pro výpočet maximálního proudu budeme uvažovat, že veškerý proud poteče přes R_1 do vstupu Arduina a nulový proud přes R_2 na zem (GND).

$$I = \frac{U}{R_1} \tag{20}$$

$$I = \frac{24V}{100k\Omega} \tag{21}$$

$$I = \frac{24V}{390k\Omega} \tag{22}$$

$$I \doteq 0{,}6mA \tag{23}$$

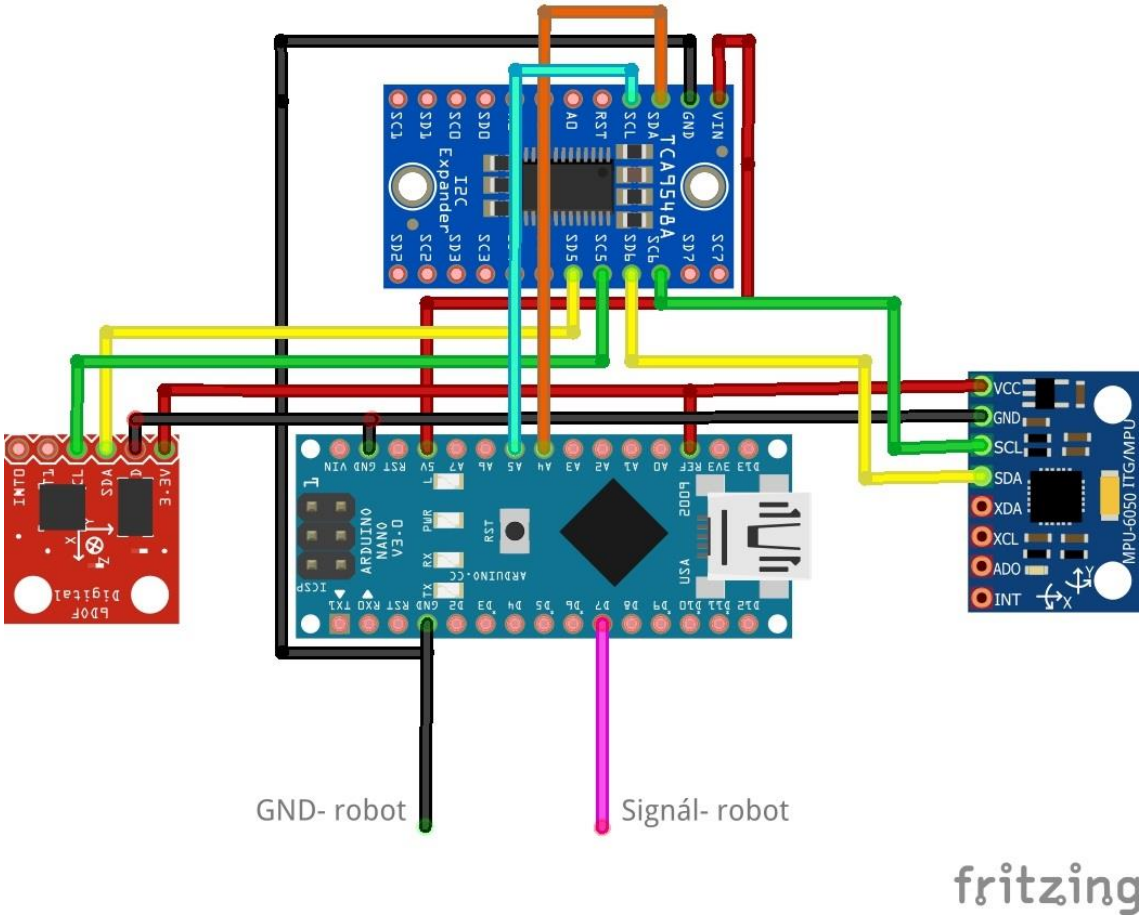
Proud do vstupu Arduina bude tedy maximálně 0,6mA <40 mA. Navržený dělič napětí tedy bude vyhovovat aplikaci.

Zapojení bylo provedeno pomocí dvojlinky 0,5mm, kde v jedné žíle je veden signál a druhá žíla slouží k propojení GND obou zařízení, robotu a mikroprocesoru, aby byly uvedeny na stejný potenciál.

5.4 Elektrické schéma zapojení

Elektronické schéma jsem vytvořil pomocí programu *Fritzing* 0.9.3 b, který obsahuje modely většiny užitých součástek a ostatní součástky je možné vyhledat a stáhnout v internetové databázi.

Vývojová deska Arduino Nano s mikroprocesorem ATmega328 je napájena pomocí USB mini z počítače. Tato skutečnost není jako jediná zanesena do schématu z důvodu přehlednosti. Oba akcelerometry, jak MPU6050, tak ADXL345 jsou napájeny z Arduina napětím 3,3 V. Akcelerometr ADXL345 přímo vyžaduje toto napájecí napětí dle *datasheetu*, ale MPU6050 je možné provozovat jak při 3,3V, tak při 5 V. Bylo zvoleno shodné napájecí napětí. Komunikační kanály MPU6050 SDA a SCL jsou připojeny na kanál multiplexoru č. 6 a akcelerometru ADXL345 na kanál č.5. MUX je napájen napájecím napětím 5 V z Arduina. Datové výstupy multiplexoru SDA a SCL jsou připojeny na Analogové piny Adruina A4 a A5, které jsou předdefinované pro připojení komunikace přes sběrnici I2C. Růžový vodič připojený do digitálního vstupu D7 je výše zmíněný signál z robotu *„Trigger“*.



Obrázek 18: Schéma elektronického zapojení měřícího zařízení, zpracované v programu Fritzing (vlastní obrázek).

5.5 Program Mikrokontroleru

Program mikrokontroleru jsem vytvořil v programovacím prostředí Arduino IDE 2.0.3. Shrnutí hlavních požadavků na program:

- Vyčíst data ze dvou různých akcelerometrů,
- vyčíst data z gyroskopu,
- převést veličiny z akcelerometrů na stejné jednotky,
- odeslat data na sériovou linku ve formátu vhodném pro další zpracování.

Pro tvorbu programu byly použity následující knihovny; všechny jsou volně ke stažení:

Název knihovny	Popis
TCA9548A.h	Nastavení a správa MUX
MPU6050_light.h	Nastavení a vyčítání dat z MPU6050
SparkFun_ADXL345.h	Nastavení a vyčítání dat z ADXL345

math.h	Rozšířené matematické operace a konstanty
Wire.h	Zprostředkování komunikace I2C

Tabulka 2:Použité knihovny v programu mikrokontroleru.

Oba akcelerometry i MUX mohou komunikovat po stejné sběrnici I2C. Akcelerometr může komunikovat i přes sběrnici SPI, ale pro jednoduchost byla zvolena I2C pro všechna zařízení.

Program obsahuje tři základní části:

- Definici a deklaraci.
- Nastavení.
- Smyčku.

5.5.1.1 Definice a deklarace

V této části programu jsou načteny knihovny v tabulce č.1. Definovány objekty I2C zařízení, v našem případě **adxl** pro akcelerometr ADXL345, **I2CMUX** Pro multiplexor a **mpu** pro akcelerometr MPU6050.

Dále jsou zde definice globálních proměnných používaných v následujících částech programu.

5.5.1.2 Nastavení

V programu se jedná o prázdnou funkci bez návratové hodnoty. Nese název ***void setup*** **()**, která je spuštěna při startu programu jako první a právě jednou. V našem případě obsahuje inicializaci měřícího zařízení, která je nutná pro následné měření. Funkce *void setup* () obsahuje v pořadí tyto kroky:

- Nastavení pinu 7 na digitální vstup (pro čtení signálu trigger).
- Spuštění sériové linky s modulační rychlostí 115200 baud.
- Spuštění komunikace I2C s MUX.
- Otevření komunikačního kanálu 5 na MUX (zde je připojen ADXL345).
- Spuštění akcelerometru ADXL345.
- Nastavení rozsahu ±4 g u ADXL345.
- Výpočet korekce pro jednotlivé osy. Tento výpočet se provádí na základě 200 iterací, při kterých dochází ke čtení zrychlení všech 3 os a sumaci jednotlivých měření. Výpočet korekce se musí provádět v klidovém stavu

akcelerometru, jinak dojde k znehodnocení všech následně naměřených dat. Po sumaci všech 200 čtení jednotlivých os je vypočtena průměrná hodnota opětovným dělením sumy číslem 200. Tímto získáme korekci pro klidový stav akcelerometru pro všechny osy a tyto hodnoty jsou uložené do globálních proměnných. Také jsou odeslány na sériovou linku pro orientační kontrolu správnosti výpočtu korekce.

- Uzavření kanálu 5 na MUX a následné otevření kanálu 6 (zde je připojen MPU6050).
- Spuštění akcelerometru MPU6050.
- Výpočet korekcí stejným způsobem jako u ADXL345.
- uzavření všech kanálů MUX.

Pozn.: Rozsah ±4 g MPU6050 je nastaven přímo v knihovně MPU6050_light.h a nikoliv ve funkci *void setup* ().

5.5.1.3 Smyčka

Jedná se o prázdnou funkci **void loop()** opět bez návratové hodnoty, která se cyklicky spouští, pokaždé když dojde k provedení její poslední instrukce. Jedná se o nekonečnou smyčku.

Funkce void loop() obsahuje v pořadí tyto základní kroky

- Spuštění časovače milis(), který ukládá hodnotu času v ms, od spuštění do proměnné time.
- otevření kanálu 5 na MUX (ADXL345)
- vyčtení dat zrychlení všech tří os a uložení hodnot do příslušných proměnných
- provedení převodu na jednoty m/s^{-2} a odečtení příslušné korekce získané při inicializaci.
- Uzavření kanálu 5 na MUX
- Otevření kanálu 6 na MUX (MPU6050)
- vyčtení dat zrychlení všech tří os a uložení hodnot do příslušných proměnných
- provedení převodu na jednoty m/s^{-2} a odečtení příslušné korekce získané při inicializaci.

- Vyčtení dat z registrů pro gyroskop (převod jednotek zajišťuje již knihovna MPU6050_light) a následné uložení hodnot do příslušných proměnných.
- Uzavření všech kanálu na MUX
- Vyčtení digitálního vstupu DI 7 a uložení stavu do proměnné.
- Následuje výpis na sériovou linku ve formátu: aktuální čas od spuštění, zrychlení x, y, z pro ADXL345, zrychlení x, y, z pro MPU 6050, úhly natočení z MPU6050, signál trigger (Input 7). Přesný formát výpisu na sériovou linku je popsán v kapitole **5.6.4 Zpracování dat z akcelerometrů**
- Časová prodleva 10ms před spuštěním cyklu znovu.

5.6 Sběr a zpracování dat

Než začneme popisovat postupy sběru a zpracování dat, je zapotřebí si rozdělit sběr dat do dvou kategorií. První kategorie je sběr dat z měřícího zařízení (resp. z akcelerometrů) a druhá kategorie je sběr dat z počítačové simulace. Výsledem by měly být tři sady dat (dvě z akcelerometrů a jedna ze simulace), které budou spolu následně porovnány.

Vyšel jsem ze skutečnosti, že základním předpokladem je nutnost porovnávat stejné jednotky ve stejném čase. Zde se již objevuje první rozdíl mezi sadou z akcelerometrů a ze simulace. Předpokládáme, že data získaná z akcelerometrů budou úměrná aktuálnímu zrychlení v jednotkách m/s^{-2}. Na druhé straně data získaná ze simulace budou odpovídat poloze v prostoru v čase. Tedy polohovému vektoru *r(t).*

Aby bylo možné takové dvě sady dat porovnat je zapotřebí je převést na stejné jednotky. Přichází v úvahu tři možnosti:

1. Porovnat aktuální zrychlení *a(t),* aktuální rychlost *v(t),* nebo aktuální polohu *r(t).* Při zvolení porovnávané jednotky aktuálního zrychlení *a(t),* již není nutné data z akcelerometrů upravovat, jelikož nám přímo poskytují hodnotu aktuálního zrychlení. Ovšem u dat ze simulace by bylo zapotřebí provést dvojnásobnou numerickou derivaci abychom z získali zrychlení.

$$\frac{dr(t)}{dt} = v(t) \tag{24}$$

$$\frac{d^2r(t)}{dt^2} = a(t) \tag{25}$$

2. Další možností je porovnávat aktuální rychlosti, kde by bylo zapotřebí data z akcelerometrů jedenkrát numericky integrovat zrychlení a jedenkrát derivovat data ze simulace.

$$\frac{dv(t)}{dt} = a(t) \tag{26}$$

3. Třetí možnost je srovnat aktuální polohu *r(t)*, kde by odpadla práce ze zpracování dat ze simulace a byla by zapotřebí jen dvojnásobná integrace dat z akcelerometrů. Tato operace není v našem případě tak numericky náročná, když známe velikost kroku, proto byla zvolena tato možnost.

Data pro srovnání tedy budou ve tvaru *r(t),* kde r je vektor polohy v kartézských souřadnicích *r = (x, y, z).*

Bude tedy možné sestavit pro každou sadu dat řadu z diskrétních hodnot v čase a prostoru, které když se interpolují, tak zjistíme odhad celých trajektorií, které bude možné poté porovnat mezi sebou.

5.6.1 Sběr dat z akcelerometru MPU 6050

Data z akcelerometru MPU 6050 jsou vyčítána po sběrnici I2C z 16 ti bitových registrů. Jedná se o binární data, která nemají fyzikální význam. Po převedení do dekadické soustavy tato data mohou nabývat hodnot 0–65535. Je tedy zapotřebí je převést na fyzikální veličiny.

Akcelerometr lze provozovat ve čtyřech různých rozsazích a to ±2 g, ±4 g, ±8 g a ±16 g.

Příklad: Při nastavení rozsahu 4 g bude dekadická hodnota 32768 odpovídat klidovému stavu a hodnota 65535 bude odpovídat zrychlení +4 g, kde g je gravitační zrychlení, tady přibližně 4 · 9,81 [m/s^2] = 39,24 [m/s^2]. Naopak hodnota dekadická hodnota 0 bude znamenat pohyb v opačném směru, tedy -4 g.

Toto nastavení rozsahu zajišťuje volně dostupná a stažitelná knihovna MPU6050_light.

Tímto způsobem jsou tedy po sběrnici I2C data vyčtena z registrů do Arduina, kde jsou převedena do fyzikální podoby zrychlení [m/s^2] (InvenSense Inc, 2013).

5.6.2 Sběr dat z akcelerometru ADXL345

Data z akcelerometru ADXL345 jsou vyčítána opět po sběrnici I2C z deseti až třinácti bitových registrů. Počet užitých bitů závisí na nastaveném rozsahu.

Rozsah	Počet bitů
2 g	10
4 g	11
8 g	12
16 g	13

Tabulka 3: Tabulka užitých registrových bitů dle zvoleného rozsahu.

Pro měření jsem zvolil stejný rozsah pro MPU6050, a to ±4 g. Hodnota vyčtená z jedenácti registrů může tedy nabývat hodnot 0-2048. S touto dekadickou hodnotou postupujeme stejně jako u předešlého akcelerometru – tedy přepočteme ji na zrychlení v jednotkách [m/s^2].

Nastavení rozsahu zajišťuje volně dostupná a stažitelná knihovna SparkFun_ADXL345 (Jeff Watson, 2016).

5.6.3 Sběr dat ze simulace

Simulační program *Robotstudio*, nabízí při zakoupení plné licence provádět simulace včetně záznamu signálů a různých parametrů robotu. Data získaná ze záznamu simulace jsou data aktuální polohy ve všech třech souřadnicích pravoúhlého kartézského systému x, y, z. Jedná se o záznam aktuální polohy nástroje. Nástrojem se zde myslí koncový bod robotu, kde je u fyzického robotu upevněno měřící zařízení.

Takto získaná data lze z programu exportovat ve formátu *.xls. Tato data jsou v jednotkách vzdálenosti od počátku [mm] v čase [s]. Počátkem je v tomto případě

myšlena základna robotu. O této problematice víc v kapitole **5.6.4.5 Zpracování dat ze simulace**.

5.6.4 Zpracování dat z akcelerometrů

Z obou akcelerometrů se podařilo získat data ve stejném formátu [m/s^2]. Pro následné zpracování pracuji s oběma sadami stejně.

Nyní je zapotřebí aktuální zrychlení v čase převést na polohu v čase. Pro vyšší efektivitu byly některé výpočty provedeny až následně pomocí výpočetního výkonu počítače nikoliv mikrokontroleru. Z Arduina byla data odeslána na sériový monitor v následujícím formátu:

TIME:,2602,RED,0.2115,0.5223,0.3669,BLUE,0.0097,0.2367,.0,9981,GYRO: ,21.17,11.53,0.96 ,TRIGGER,0

Mezi jednotlivými vypsanými informace jsou použity čárky „,“. Ty slouží jako oddělovač pro následné zpracování dat a nemají žádný jiný význam.

TIME:

Tento název slouží k popisu následující vypsané veličiny času.

2602

Čas od spuštění Arduina v jednotkách [ms]. (Funkce millis()) slouží k následnému výpočtu časových kroků.

RED

Slouží k označení, že následující data budou patřit k akcelerometru ADXL345. Tento pracovní název byl zvolen, aby na první pohled byly akcelerometry rozeznatelné, protože ADXL345 je vyroben v červeném provedení.

0.2115 0.5223 0.3669

Hodnoty aktuálního zrychlení v [m/s^2] pro osy x, y, z v tomto pořadí. K oddělení celé a desetinné části čísla je použita desetinná tečka (.).

BLUE

Slouží k označení, že následující data budou patřit k akcelerometru MPU6050.

0.0097 0.2367 0,9981

Hodnoty aktuálního zrychlení v [m/s^2] pro osy x, y, z v tomto pořadí. K oddělení celé a desetinné části čísla je použita desetinná tečka (.).

GYRO

Slouží k označení, že následující data budou hodnoty natočení získané z MPU6050.

21.17,11.53,0.96

Hodnoty aktuálního náklonu gyroskopu MPU6050 ve [°].

TRIGGER

Slouží k označení, že následující hodnota je digitální signál z robotu

0

Nula značí, že robot je v pohybu, 1 že robot je v klidu. Více informací v kapitole **5.1 Měření trajektorie v simulaci**.

K Arduinu byl během testu připojen notebook pomoci USB-B ↔ mini USB kabelu. Díky tomu bylo možné provádět záznam dat vypsaných na sériový monitor. Bylo potřeba na počítač nainstalovat volně stažitelný ovladač CH340, aby bylo možné číst data z digitálního převodníku Arduina.

5.6.4.1 Sériový monitor → Excel

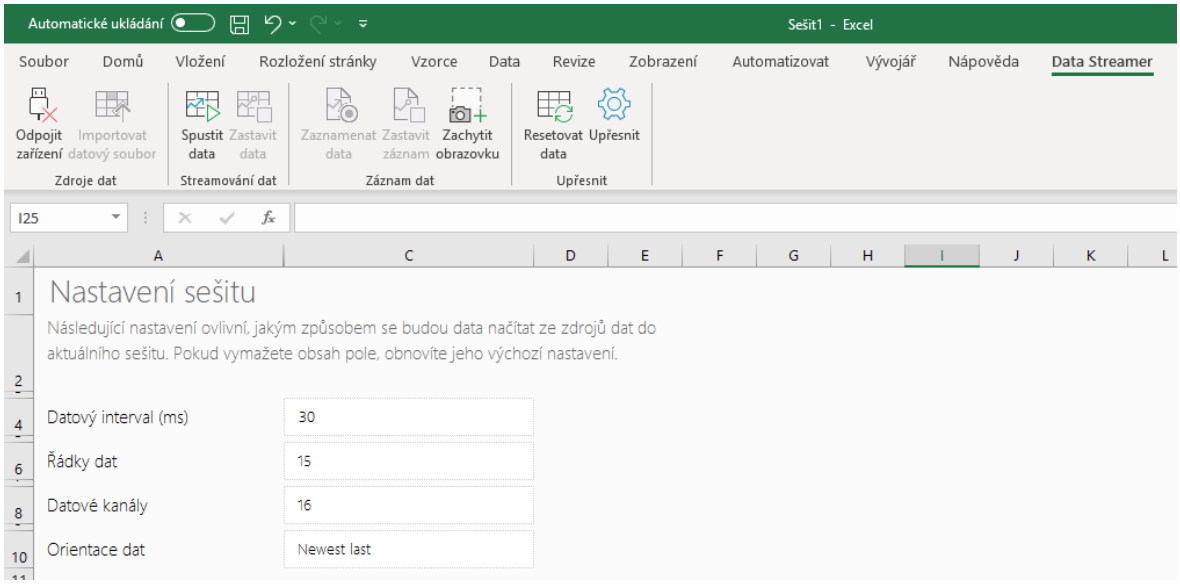
Dalším krokem zpracování dat je načtení dat do tabulkového procesoru Excel. Zde jsem využil doinstalování rozšíření s názvem *"Data Streamer"*, který slouží k vyčítání dat ze sériové linky a ukládání do souboru *.csv.

Je třeba připojit zařízení a zvolit odpovídající port ze seznamu zařízení. V našem případě COM 9.

Dále je zapotřebí *Data Streamer* nastavit pro přenos. K tomu slouží záložky 3 a 4, které se automaticky vygenerují po připojení zařízení.

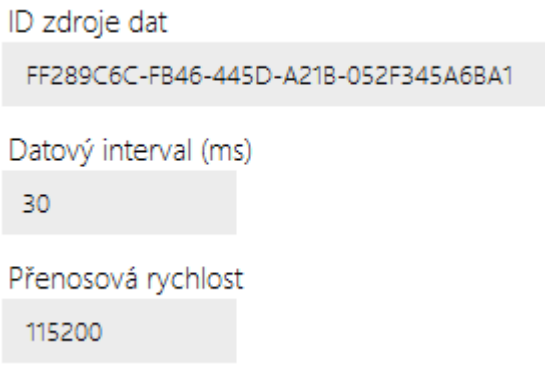
- Na kartě č.3 s názvem "Nastavení" je třeba nastavit frekvenci vyčítání dat "**Datový interval (ms)**".
- Kolonka "**Řádky dat**" není stěžejní. Udává jen v kolika řádcích se budou data zobrazovat. Toto nastavení nijak neovlivní záznam dat, proto byla ponechána výchozí hodnota.

- Vyplněním “**Datové kanály**” upravujeme počet sloupců, do kterých bude *Data Streamer* data třídit.
- Nastavením “**Orientace**” udáváme logiku řazení nově příchozích dat.



Obrázek 19:Nastavení Data Streameru v záložce č.3 “Nastavení“ (vlastní obrázek).

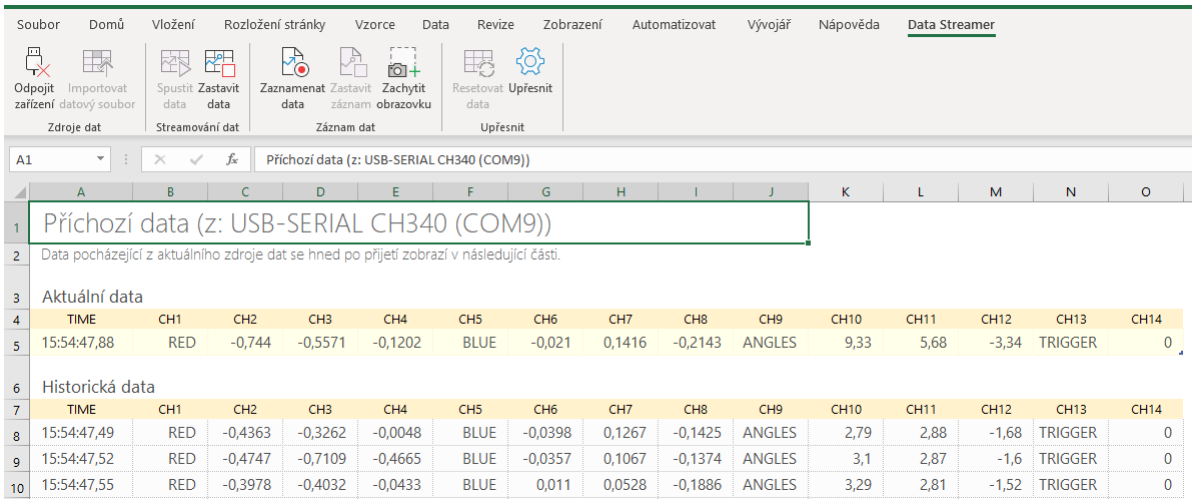
Na kartě č.4 se automaticky vyplní nastavení po připojení zařízení. Zde je jen potřeba údaje zkontrolovat, či opravit. Jedná se převážně o správně nastavenou přenosovou rychlost.



Nastavení Data Streameru v záložce č.4.“*Manifest*“ (vlastní obrázek).

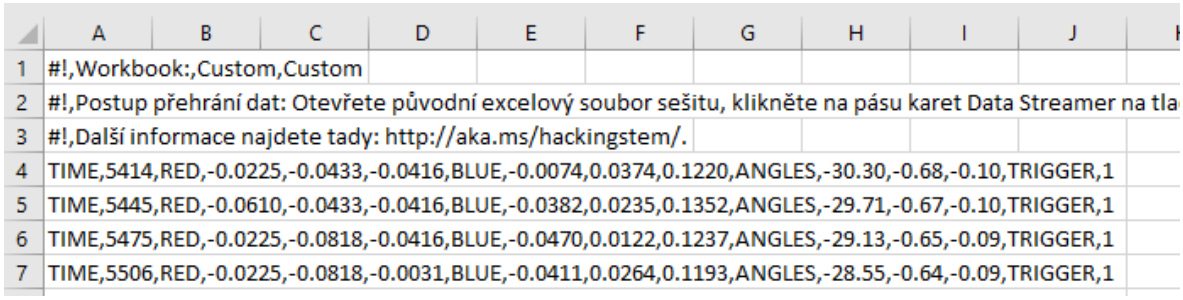
Nyní je vše nastaveno a může dojít ke spuštění dat pomocí tlačítka „Spustit data“. Pokud je vše správně nastavené, tak dojde ke streamování obsahu odeslaném na sériovou linku do tabulky na kartě č.1 „Vstupní data“.

Na obrázku č.20 je vidět, jak takový přenos probíhá. Jedná se jen o test připojení, proto neodpovídá např. počet kanálů a formát odesílaných dat.



Obrázek 20: Test spuštění Data Streameru (vlastní obrázek).

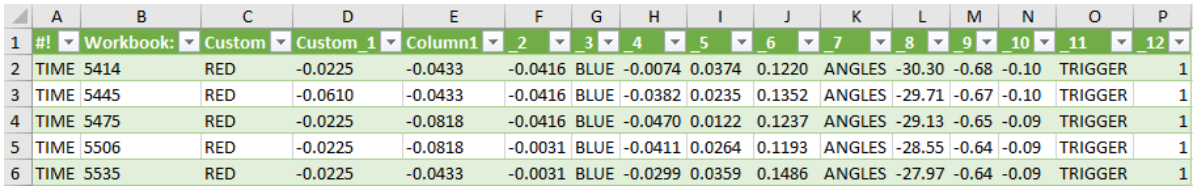
Pokud se data zobrazují správně, můžeme spustit záznam pomocí tlačítka „Zaznamenat data“ a následně jej ukončit kliknutím „Zastavit záznam“. Po kliknutí Tlačítka zastavit záznam vybereme cestu uložení souboru a formát souboru. V našem případě užíváme formát *.csv.



*Obrázek 21: Zaznamenaná data v souboru *.csv (vlastní obrázek).*

Soubor *.csv nahrajeme do nového souboru *.xlsx pomocí importu datového souboru. Kliknutím na **Data** v horní záložce a následně **Z text/CSV**. Po vybrání cesty k souboru *.csv je potřeba nastavit jaký je v datech použit oddělovat. V tomto případě byl zvolen oddělovač čárky (,) viz **5.6.4. Zpracování dat z akcelerometrů**.

Po načtení do souboru *.xlsx jsou data připravena pro další zpracování.



*Obrázek 22: Data nahraná ze souboru *.csv do souboru *.xlsx (vlastní obrázek).*

5.6.4.2 Excel → Octave

Nyní jsem stál před problémem provést matematické operace a získat z aktuálního zrychlení *a(t),* aktuální polohu *r(t).* V této fázi je zapotřebí pracovat s velkými objemy dat. Data jsou uložena v tabulce, čehož lze využít: Pohlížím na tabulku jako na velkou matici, s jejímiž členy budu pracovat. Pro práci s maticemi jsem zvolil program Octave, *freewere* alternativu k programu Matlab. Slouží k matematickým výpočtům a je v něm snadná práce s maticemi. Do programu Octave lze nahrát soubor *.xlsx a uložit obsah buněk jako členy matice.

5.6.4.3 Výpočet v Octave

Hlavním úkolem v programu Octave bude již zmíněný přepočet aktuálního zrychlení *a(t),* na aktuální polohu *r(t).* Pro tento úkol je nutné spočítat časový interval mezi zaznamenanými vzorky. Vzorky označíme n; n +1; …; n + k;, kde n + k je poslední vzorek. Tento výpočet provedeme odečtením aktuálního času TIME vzorku, jež označíme t_n a časový interval označíme h_i. Jak je již zmíněno na začátku této kapitoly **5.6.4 Zpracování dat z akcelerometrů**, čas TIME je aktuální čas počítaný od spouštění mikrokontroleru, tedy bude pro různá měření rozdílný. V našem případě to ovšem nevadí, protože nás zajímá jen z důvodu výpočtu velikosti kroku.

$$h_i = t_{n+1} - t_n \tag{27}$$

Bylo možné použít průměrný krok u všech iterací, ovšem zde se nejedná o nijak výpočetně náročnou operaci, proto si můžeme dovolit zpřesnit výsledek I přesnou velikostí kroku.

Pro výpočet aktuální polohy bude zapotřebí provést dvojnásobnou numerickou integraci. První integrací získáme ze zrychlení rychlost a druhou integrací získáme z rychlosti polohu. Pro numerické integrování byla zvolena Obdélníková metoda. Více v kapitole **4.6. Numerická integrace**.

5.6.4.4 Popis programu Octave

Pro popis programu jsem zvolil značení pomocí čísel řádků, resp. rozpětí řádků s doprovodným komentářem.

1 Slouží ke smazání obrazovky.

2 Načtení balíčku spravující vstupy a výstupy pro import a export z externích zdrojů.

3 Vytvoření matice K a načtení do ní simulační data ze souboru *.xlsx

4 Vytvoření matice S a načtení do ní data z měřícího zařízení ze souboru *.xlsx.

5–8 Definice všech potřebných matic pro výpočet, jejichž velikost je odvozena od velikosti matice S. Například matice V bude mít počet řádku o jedna nižší než matice S, protože při numerické integraci přijdeme o jednu hodnotu, viz kapitola **4.6. Numerická integrace**.

9 Proměnná t obsahující vzájemný úhel natočení souřadnicových systémů robotu oproti měřícímu zařízení, převedené na hodnotu v radiánech.

10 Rotační matice pro rotaci kolem osy Z ve třírozměrném tvaru.

12-14 FOR cyklus s počtem kroků vypočtených z počtu řádku matice S ponížený o 1. Provádí výpočet velikosti kroku a přepočet a převod z [ms] na základní jednotku [s].

16–20 FOR cyklus, který provede pootočení trajektorie ze simulace v prostoru o hodnotu v proměnné t [°] kolem počátku.

22–27 FOR cyklus, který přenásobením složek matice ve sloupcích s daty zrychlení x y z, upraví orientaci. (Nutné z důvodu, že akcelerometry jsou při experimentu otočené a směr jednotlivých os neodpovídá souřadnicovému systému robotu.

29–33 První numerická integrace. Získání matice aktuálních rychlostí V za užití obdélníkového pravidla a již známou velikost kroku.

35–39 Vytvoření matice V0, jež obsahuje časový vývoj rychlostí Vx, Vy, Vz.

41–45 Druhá numerická integrace. Získání matice P obsahující změnu polohu x, y, z.

47–51 Vytvoření matice P0 obsahující průběh poloh robotu, tedy trajektorii x, y, z.

52 Příkaz pro uložení matice P0 pro další zpracování a analýzy.

54 Vynesení do grafu pozice x, y, z z matice P0 (Data z měřícího zařízení) a proložení spojnicí bodů.

55 Vynesení do grafu pozice x, y, z z matice K (simulační data) a proložení spojnicí bodů.

56-59 Nastavení popisků os a názvu grafu

Mojí snahou bylo udělat program univerzální, aby bylo možné nahrávat libovolně dlouhá data (o různých počtech řádků) bez nutnosti provádění změn v kódu.

```
clear
pkg load io
K = xlsread('sim.xlsx');
S = xlsread('RED/2000_R.xlsx');
V = zeros(size(S,1)-1,3);
V0 = zeros(size(S,1)-1,3);
P = zeros(size(S,1)-2,3);
P0 = zeros(size(S,1)-2,3);
t=deg2rad(15);
Rz = [cos(t) -sin(t) 0; sin(t) cos(t) 0; 0 0 1];

for i=1:size(S,1)-1
H(i,1)=(S(i+1,1)-S(i,1))/1000;
end

for q=1:size(K,1)
R=K(q,2:4);
R=R';
K(q,2:4)=K(q,2:4)*Rz;
end

for i=1:size(S,1)

S(i,2)=S(i,2)*-1;
S(i,3)=S(i,3)*-1;
S(i,4)=S(i,4)*-1;
end

for i=1:size(S,1)-1
  V(i,1)=(S(i+1,2)+S(i,2))*H(i,1)/2;
  V(i,2)=(S(i+1,3)+S(i,3))*H(i,1)/2;
  V(i,3)=(S(i+1,4)+S(i,4))*H(i,1)/2;
end

for i=1:size(S,1)-1
  V0(i+1,1)=V0(i,1)+V(i,1);
  V0(i+1,2)=V0(i,2)+V(i,2);
  V0(i+1,3)=V0(i,3)+V(i,3);
end

for i=1:size(S,1)-2
  P(i,1)=(V0(i+1,1)+V0(i,1))*H(i,1)/2;
  P(i,2)=(V0(i+1,2)+V0(i,2))*H(i,1)/2;
  P(i,3)=(V0(i+1,3)+V0(i,3))*H(i,1)/2;
end

for i=1:size(S,1)-2
  P0(i+1,1)=P0(i,1)+P(i,1);
  P0(i+1,2)=P0(i,2)+P(i,2);
  P0(i+1,3)=P0(i,3)+P(i,3);
end
xlswrite('RED/2000_R_vysledek.xlsx',P0);

plot3(P0(2:size(P0,1),2),P0(2:size(P0,1),1),P0(2:size(P0,1),3),'b-');
plot3(K(:,2),K(:,3),K(:,4),'r-');
title('Trajektorie MPU 6050 při v=2000mm/s')
xlabel('Osa X')
ylabel('Osa Y')
zlabel('Osa Z')
```

Obrázek 23: Kód programu Octave (vlastní obrázek)

Po výpočtu aktuální polohy dojde k vykreslení grafu z těchto dat. Pokud vzneseme spojnicový graf, tak bude tato aproximace přibližně odpovídat reálné trajektorii robotu v kartézských souřadnicích.

5.6.4.5 Zpracování dat ze simulace

Data ze simulace jsou po exportu z *Robotstudia* v námi požadovaném formátu, tj. v souboru *.xlsx s výpisem aktuálních poloh v daném čase.

1	Time	X Position In Current Wobj	Y Position In Current Wobj	Z Position In Current Wobj
2	0,000	-53,530	328,130	1054,240
3	0,048	-53,530	328,130	1054,239
4	0,096	-53,530	328,130	1054,240
5	5,064	-53,530	328,130	1054,240
6	5,088	-52,180	328,482	1054,240
7	5,112	-42,037	331,121	1054,240
8	5,136	-26,016	335,287	1054,240
9	5,160	-8,463	339,853	1054,239

Obrázek 24: Formát dat ze simulace

Soubor se tedy opět nahraje do programu Octave a vykreslí se trojrozměrný graf.

5.7 Experiment

Aby mohlo dojít k porovnání dynamických vlastností simulace a reálného robotu bylo zapotřebí navrhnout tzv. referenční trajektorii. Referenční trajektorie je specifická pro tento experiment a bude použita pro všechna měření. Tato trajektorie musí být dostatečně komplexní, aby připomínala reálné užití robotu v praxi, ale zároveň dostatečně jednoduchá, aby jednotlivé pohyby nesplývaly a bylo je možné popsat.

Navrhnul jsem trajektorii, obsahující základní pohyby robotu:

- Pohyby lineární,
- pohyb po kružnici.

Pro popis jsem zvolil významné body trajektorie označené jako P1, P2, P3, P4 a P5. V programovacím prostředí *Robotstudia* jsem naprogramoval tuto trajektorii za užití programovacího jazyka RAPID, specifického pro roboty ABB.

Obrázek 25: Robotická stanice pro provádění experimentu (vlastní fotografie).

5.7.1 Popis cyklu robotu

Robot přestaví svá ramena, tak, aby dosáhl své domácí pozice „*home*“ (někdy označovaná jako „*základní pozice*“), jejíž poloha je shodná s bodem P1. Pozice „*home*“ je definovaná pomocí úhlů natočení všech šesti kloubů robotu a nikoliv bodem v prostoru. Dále čeká 5 sekund ve své domácí pozici. Poté jede lineárně do bodu P2. To znamená, že přizpůsobí pohyb všech svých kloubů tak, aby došlo k translačnímu posuvu nástroje (měřicího přístroje). Z bodu P2 pokračuje do bodu P3 také lineárně, kde jednu sekundu čeká a poté pokračuje reverzně zpět tj. lineárně do bodu P2, následně do P1. Z bodu P1 pokračuje pohybem po kružnici přes bod P4 do bodu P5. Tento pohyb je určen třemi body (P1, P4, P5), které jasně specifikují kružnici. Bod P4 slouží jen k definici pohybu a nebude dále zahrnut do analýz. Po dosažení bodu P5 také čeká 1 sekundu a následně se po stejné kružnici vrací. Tedy P5, P4, P1. Tímto cyklus končí.

V praxi robot odebírá díly ze vstřikovacího stroje a trajektorie vypadá obdobně. Čekání na začátku simuluje čekání na nově vyrobený díl. Čekání v bodě P3 simuluje prodlevu, která je zapotřebí k uchopení dílu robotem. Čekání v bodě P5 pak časovou prodlevu při odkládání dílu na dopravníkový pás.

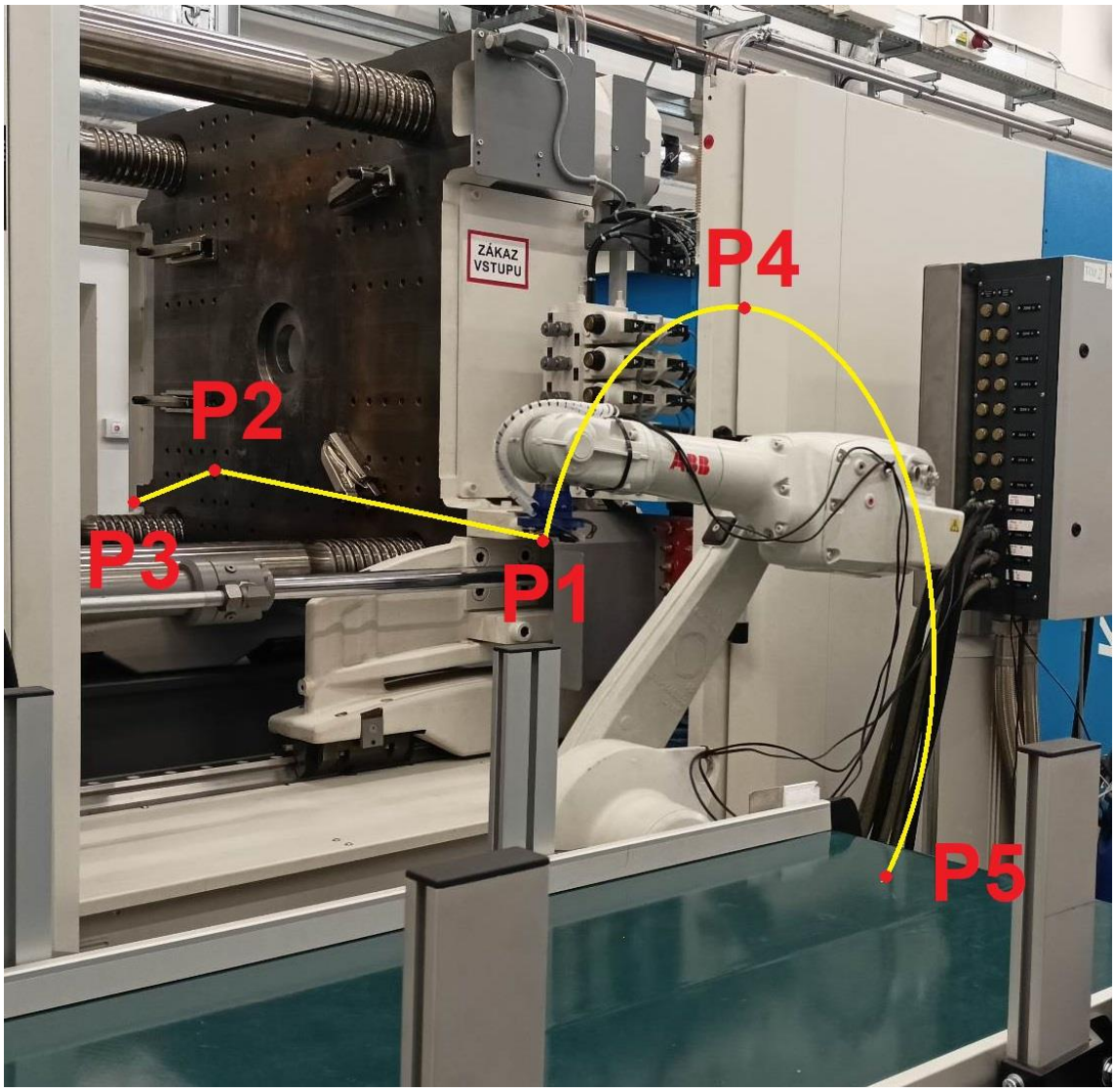
Zjednodušený zápis pohybu **P1→P2→P3→P2→P1→P4→P5→P4→P1**

Jeden z nejzásadnějších parametrů pohybu robotu je *maximální rychlost*. Tato rychlost se nastavuje pro každý jednotlivý pohyb a udává maximální dovolenou rychlost, které smí dosáhnout TCP. Více o TCP v kapitole **5.7.2 Program robotu**. Pro naše účely byla pro každý pohyb zvolena stejná maximální rychlost v rámci jednoho měření. Tato rychlost stanovuje maximální povolenou hranici, které se robot snaží dosáhnout a udržet po co nejdelší dobu. Jinými slovy, robot se této rychlosti snaží dosáhnout na dráze svého pohybu, a když je dráha moc krátká, popřípadě když je robot v nevyhovující konstelaci ramen, tak této rychlosti ani nedosáhne.

Rychlost lze zadávat pomocí % z maximální rychlosti robotu, nebo v [mm/s]. Pro účely měření byla zvolena přehlednější varianta zadávání v rychlosti v jednotkách [mm/s]. Bylo provedeno 10 měření při různých rychlostech a stejné trajektorii.

Číslo měření [-]	Maximální dovolená rychlost [mm/s]
1	200
2	400
3	600
4	800
5	1000
6	1200
7	1400
8	1600
9	1800
10	2000

Tabulka 4: Nastavené rychlostí pro jednotlivá měření.



Obrázek 26: Nákres trajektorie s významnými body pro reálný robot. Zjednodušený zápis pohybu ***P1→P2→P3→P2→P1→P4→P5→P4→P1*** *(vlastní fotografie).*

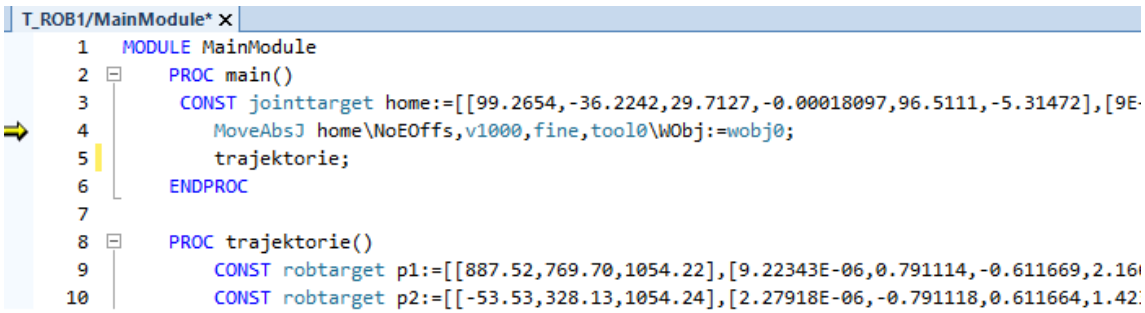
5.7.2 Program robotu

Jak uvádím výše, program robotu jsem vytvořil v *RobotStudiu* pomocí programovacího jazyku RAPID. Program obsahuje deklaraci proměnných:

- Bod v prostoru datového typu *jonttarget: home.*
- Body v prostoru datového typu *robtarget*: P1, P2, P3, P4, P5.
- Proměnná rychlosti datového typu *speedata*: Speed1.
- Dále byl definován výstup na I/O kartě d652 pracující na sběrnici *DeviceNet.* signál byl namapován na pozici „0“.
- Signál typu Digital Output: *trigger*.

Program je strukturován následovně:

Hlavní modul, který se načítá automaticky po spuštění programu *MainModule*, obsahuje proceduru *main ()*, která obsahuje pohyb do domácí pozice a proceduru trajektorie ().



Obrázek 27: Část kódu s hlavní strukturou programu

V proceduře trajektorie () je již naprogramovaný samotný cyklus.

Na obrázku č.27 je vidět hlavní tělo cyklu pro pohyb po definované trajektorii. Pro zlepšení podmínek pro následující analýzu pohybu byl po dobu pohybu robotu spínán a vypínán výstupní digitální signál trigger dle následujících pravidel:

- Když robot stojí trigger = 1
- Když je robot v pohybu trigger = 0
- Když robot projede daným bodem je spuštěn trigger ve formě pulzu po dobu 200ms. Může se tedy stát, že robot bude v pohybu a trigger bude roven 1, ovšem to je z dat velmi jasné, že se jedná o pulz, protože by se pulz neměl objevovat více než 6krát po sobě při čtení s frekvencí 30ms.

5.7.3 Slovní popis procedury trajektorie () s číslem řádku na začátku:

26 Nastavení výstupu trigger do hodnoty 1.

27 Robot čeká 5 s.

28 Nastavení výstupu trigger do hodnoty 0.

29 Lineární pohyb do bodu P2 rychlostí uloženou v proměnné speed1, fine udává pohyb přímo do bodu, tool0 je aktivní nástroj, WObj:=wobj0 Souřadnicový systém, ve kterém se robot pohybuje.

30 Spuštění 200ms pulzu výstupu trigger (program tento příkaz provede ihned a nečeká na dokončení instrukce. To znamená, že se rovnou začne pohybovat i když ještě trigger nedosáhl hodnoty 0).

31 Lineární pohyb do bodu P3.

32 Nastavení výstupu trigger do hodnoty 1.

33 Robot čeká 5 s.

34 Nastavení výstupu trigger do hodnoty 0.

35 Lineární pohyb do bodu P2.

36 Výstup trigger 200ms pulz.

37 Lineární pohyb do bodu P1.

38 Výstup trigger 200ms pulz.

39 Pohyb po kružnici začínající v aktuálním bodě, tedy bodě P1 a opisující část kružnice definované body P1, P4 a P5.

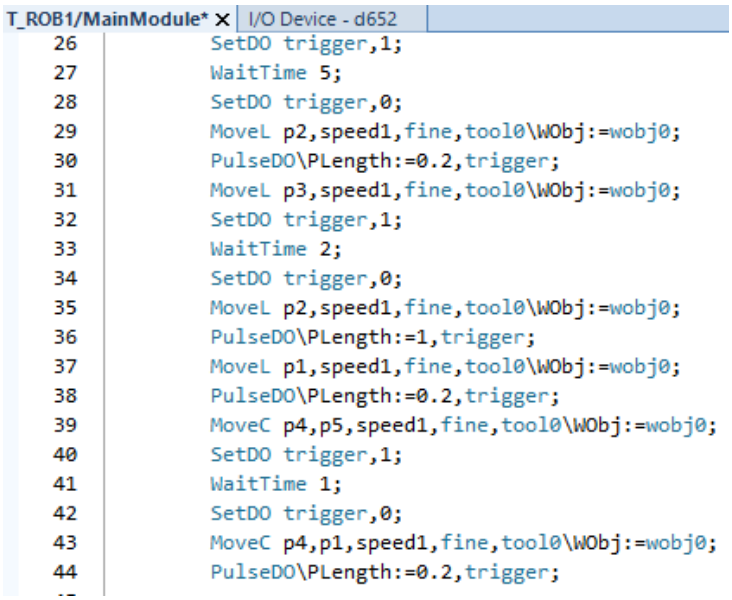
40 Nastavení výstupu trigger do hodnoty 1.

41 Robot čeká 1 s.

42 Nastavení výstupu trigger do hodnoty 0.

43 Pohyb po kružnici začínající v aktuálním bodě, tedy bodě P5 a opisující část kružnice definované body P5, P4 a P1.

44 Výstup trigger 200ms pulz.



Obrázek 28: Hlavní část procedury trajektorie (), (vlastní obrázek).

Program byl spouštěn v automatickém režimu robotu za splnění veškeré nutné bezpečnosti provozu. Změny parametru rychlosti *speed1* byly prováděny úpravou přímo v kontroleru pomocí *Teach pendantu*. Více o Teach pendantu v kapitole **5.7.1 Popis cyklu robotu**.

Aby byly zaručeny stejné podmínky pro každé měření, vždy jsem provedl restart Arduina před každým měřením, aby došlo k rekalibraci po předchozím měření.

5.8 Polohová analýza

Polohová analýza má za úkol zjistit rozdíly v trajektorii reálného robota a simulace. Tyto rozdíly budou zaneseny do tabulky a vyhodnoceny.

Jak bylo již zmíněno v kapitole **5.6 Sběr a zpracování datSběr a zpracování dat**, z naměřených dat aktuálního zrychlení byla pomocí Programu Octave provedena rekonstrukce trajektorie pomocí numerických výpočtů. Takto vypočtená trajektorie byla vynesena do stejného grafu, jako trajektorie zaznamenaná ze simulace. Data ze simulace jsou už ve formě polohy v prostoru (aktuální poloha *x*, *y*, *z* TCP v čase).

V následujících grafech jsou vždy vyneseny trajektorie:

Modrá pro trajektorii zrekonstruovanou z měření reálného robotu

Červená pro trajektorii zaznamenanou ze simulace (přesná poloha ±0.01mm)

Bylo zavedeno značení P‘ a P‘‘, udávající, že se jedná o ten samý bod, ovšem v jiné části trajektorie.

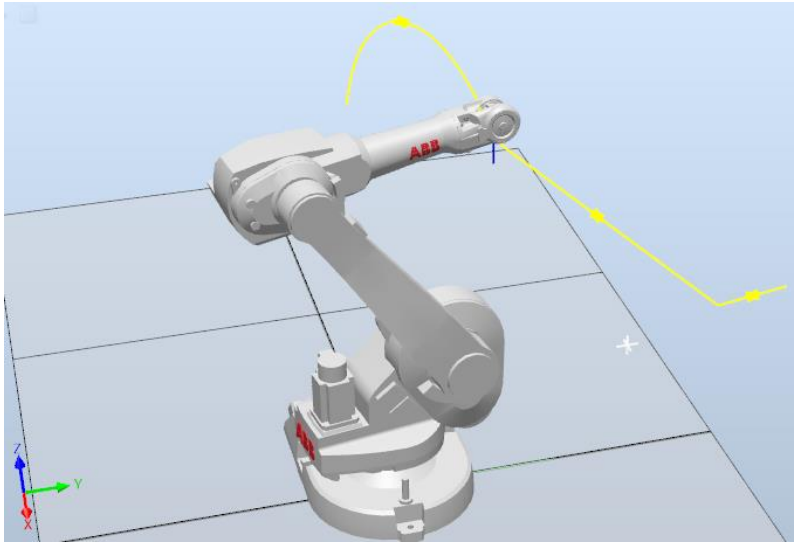
5.8.1 Trajektorie zkonstruované z dat ADXL345 a simulace

Souřadnice bodů získané při záznamu simulace jsou pro všechny nastavené rychlosti stejné. Proto je z důvodu přehlednosti nebudu dále uvádět ke každému měření zvlášť.

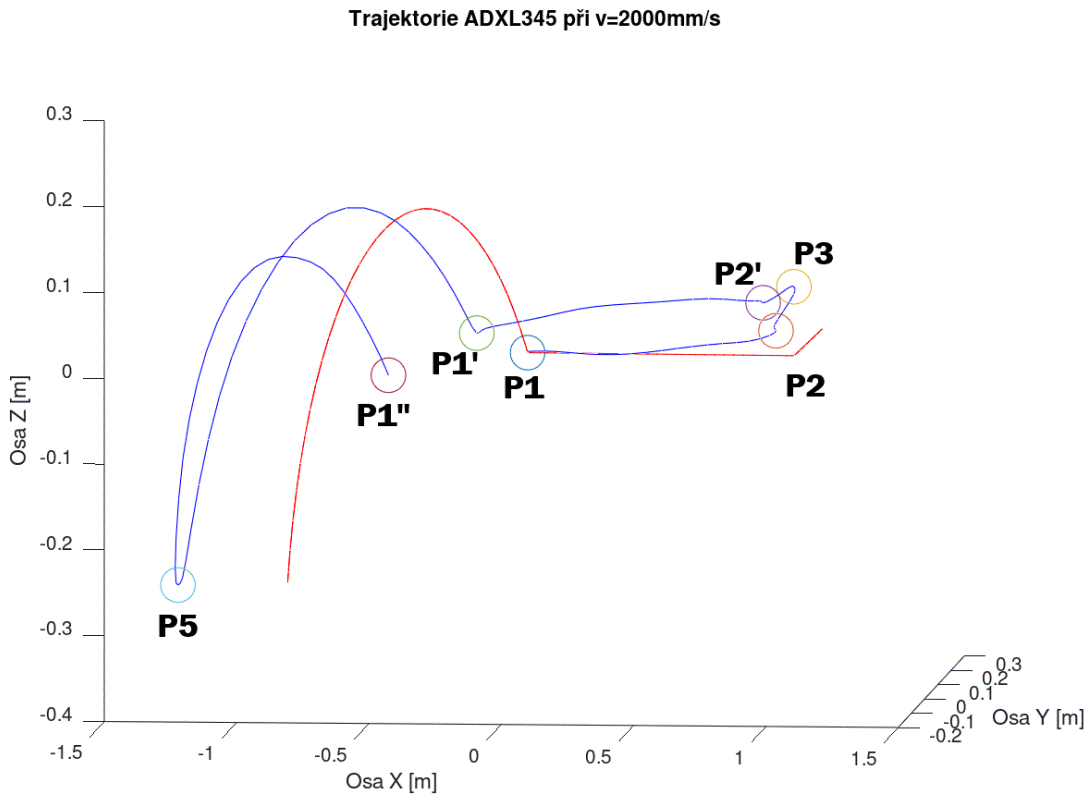
Souřadnice ze simulace			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	0,25437	0,97786	-0,00001
P3	0,44157	0,94105	-0,00002
P2'	0,25437	0,97786	-0,00001
P1'	0	0	0
P5	-0,21548	-0,89285	-0,27416
P1"	0	0	0

Tabulka 5: Souřadnice důležitých bodů získané ze simulací.

Hodnoty „Průměr“ (vzdálenost bodů [m]) v podstatě udávají chybu pozice. Níže pod jejich výčtem je vypočten aritmetický průměr, ovšem bez započtení prvního bodu, kde je vždy vzdálenost 0 m, aby nedocházelo k zatížení vypočtené hodnoty.



Obrázek 29: Trajektorie v simulaci (žlutá). Z celého modelu stanice je zobrazen jen robot (vlastní obrázek).



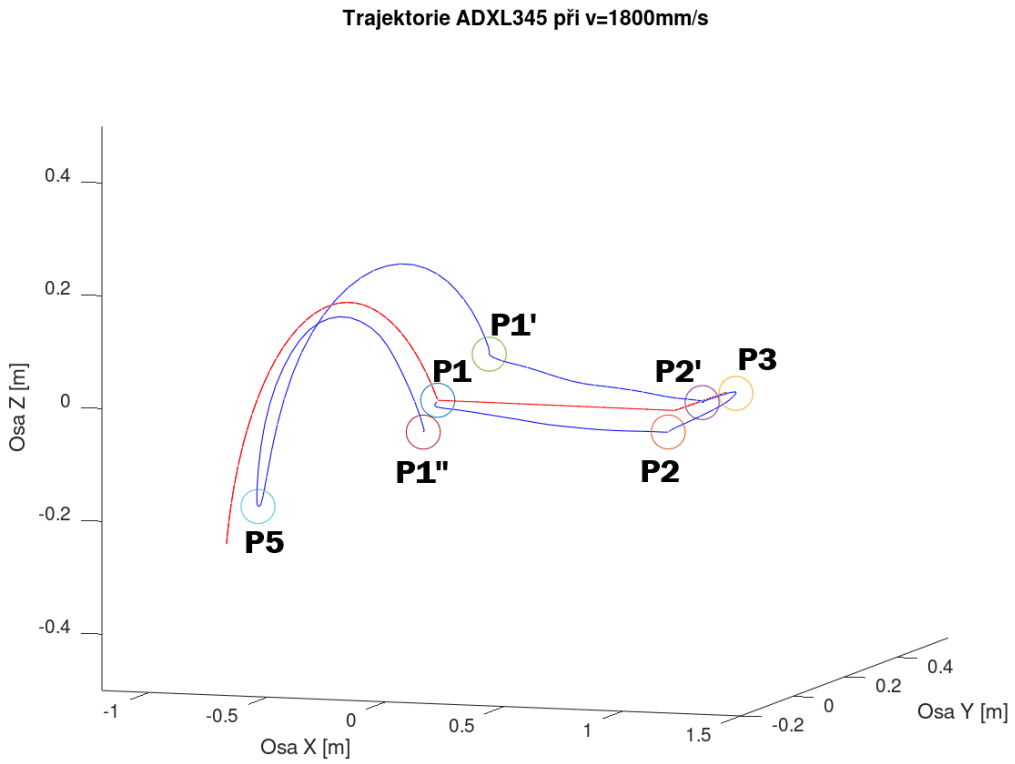
Graf 1: Trajektorie Reálného robotu při rychlosti 2000 mm/s (modrá) a simulace při rychlosti 2000 mm/s (červená). Použitý akcelerometr ADXL 345.

Souřadnice z reálného robotu pro v= 2000 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	-0,00251	0,93972	0,02806
P3	0,21548	0,89885	0,04300
P2'	0,04434	0,86751	0,05270
P1'	0,06893	-0,22672	0,01073
P5	0,02621	-1,33438	-0,27823
P1"	-0,11576	-0,46812	-0,00815

Tabulka 6: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 2000 mm/s. Použitý akcelerometr ADXL345.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,25688	0,03814	-0,02807	0,26121
0,22609	0,04220	-0,04302	0,23398
0,21003	0,11035	-0,05271	0,24304
-0,06893	0,22672	-0,01073	0,23721
-0,24169	0,44153	0,00407	0,50337
0,11576	0,46812	0,00815	0,48229
		Průměr	0,32685

Tabulka 7: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb. Použitý akcelerometr ADXL345.



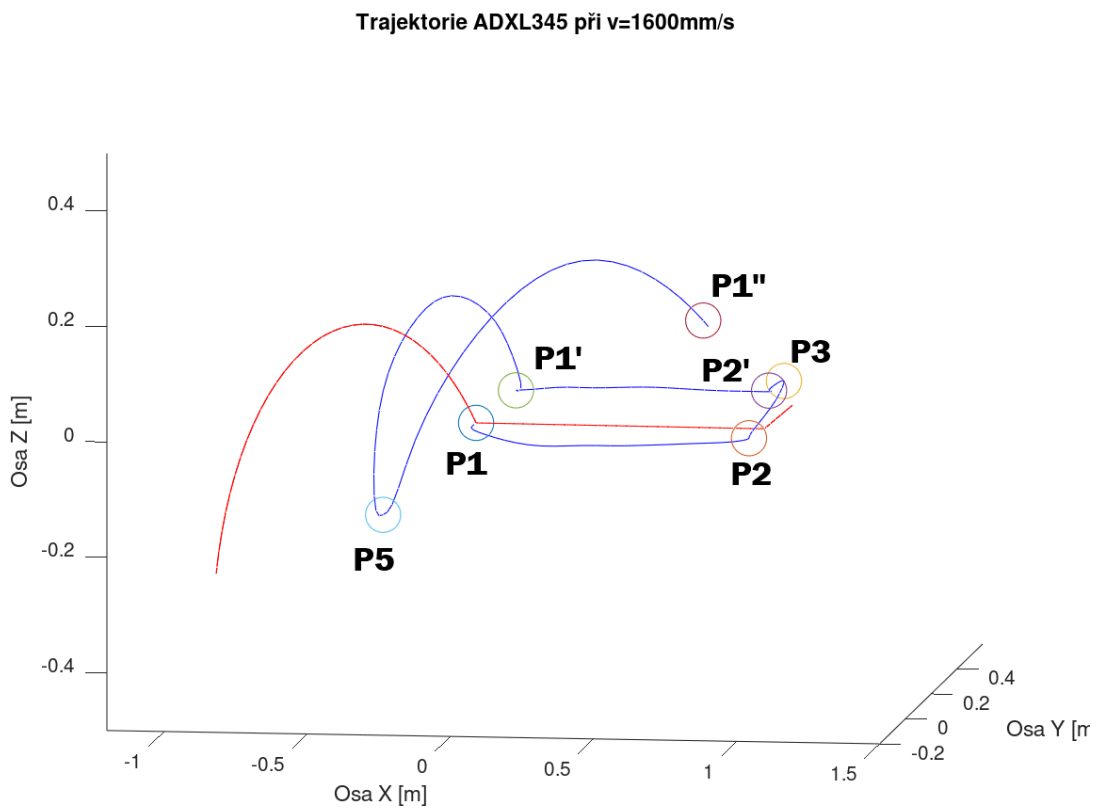
Graf 2: Trajektorie Reálného robotu při rychlosti 1800 mm/s (modrá) a simulace při rychlosti 1800 mm/s (červená). Použitý akcelerometr ADXL 345.

Souřadnice z reálného robotu pro v= 1800 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	-0,03503	1,01278	-0,03334
P3	0,18031	1,06278	-0,00129
P2'	0,02651	1,08845	0,00941
P1'	0,03617	0,17788	0,07844
P5	-0,02242	-0,73521	-0,19666
P1"	-0,11538	0,06527	-0,03539

Tabulka 8: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 1800 mm/s. Použitý akcelerometr ADXL345.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,28940	-0,03492	0,03333	0,29340
0,26126	-0,12173	0,00127	0,28822
0,22786	-0,11059	-0,00942	0,25346
-0,03617	-0,17788	-0,07844	0,19774
-0,19306	-0,15764	-0,07750	0,26101
0,11538	-0,06527	0,03539	0,13720
		průměr	0,23851

Tabulka 9:Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 1800 mm/s. Použitý akcelerometr ADXL 345.



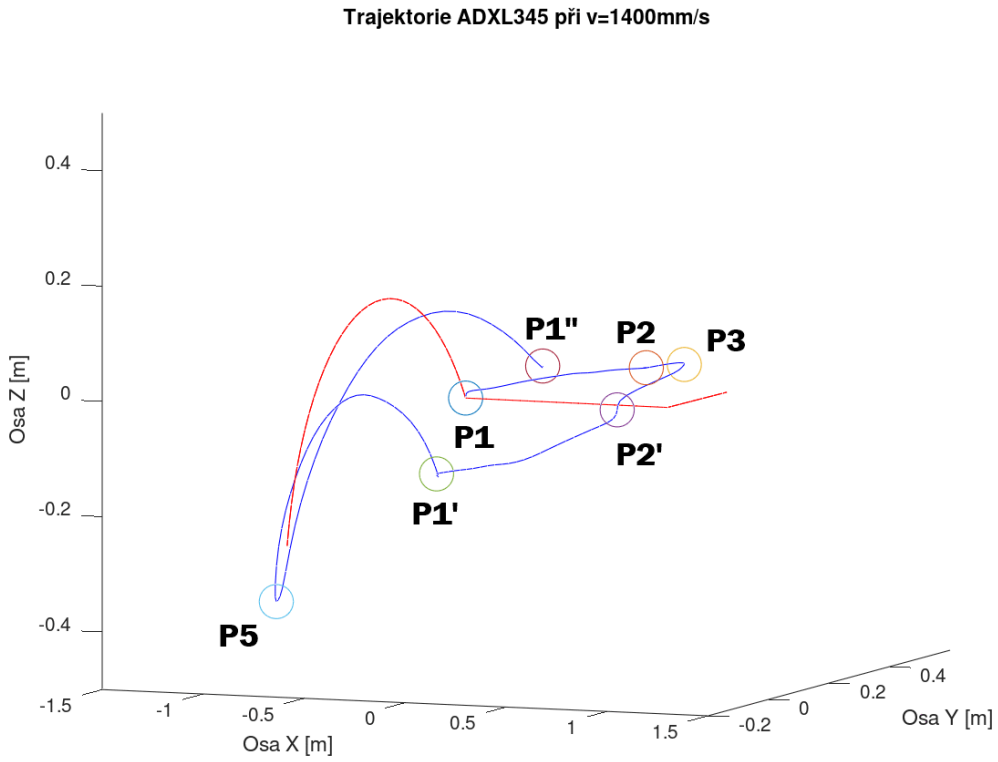
Graf 3: Trajektorie Reálného robotu při rychlosti 1600 mm/s (modrá) a simulace při rychlosti 1600 mm/s (červená). Použitý akcelerometr ADXL 345.

Souřadnice z reálného robotu pro v= 1600 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	-0,04969	0,97590	-0,00750
P3	0,12269	1,02061	0,05487
P2'	-0,02943	1,03697	0,07113
P1'	0,06147	0,11162	0,04406
P5	0,34736	-0,48123	-0,23854
P1"	0,59658	0,52518	0,05285

Tabulka 10: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 1600 mm/s. Použitý akcelerometr ADXL345.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,30406	0,00196	0,00749	0,30416
0,31888	-0,07956	-0,05489	0,33320
0,28380	-0,05911	-0,07114	0,29849
-0,06147	-0,11162	-0,04406	0,13482
-0,56284	-0,41162	-0,03562	0,69820
-0,59658	-0,52518	-0,05285	0,79656
		průměr	0,42757

Tabulka 11: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 1600 mm/s. Použitý akcelerometr ADXL 345.



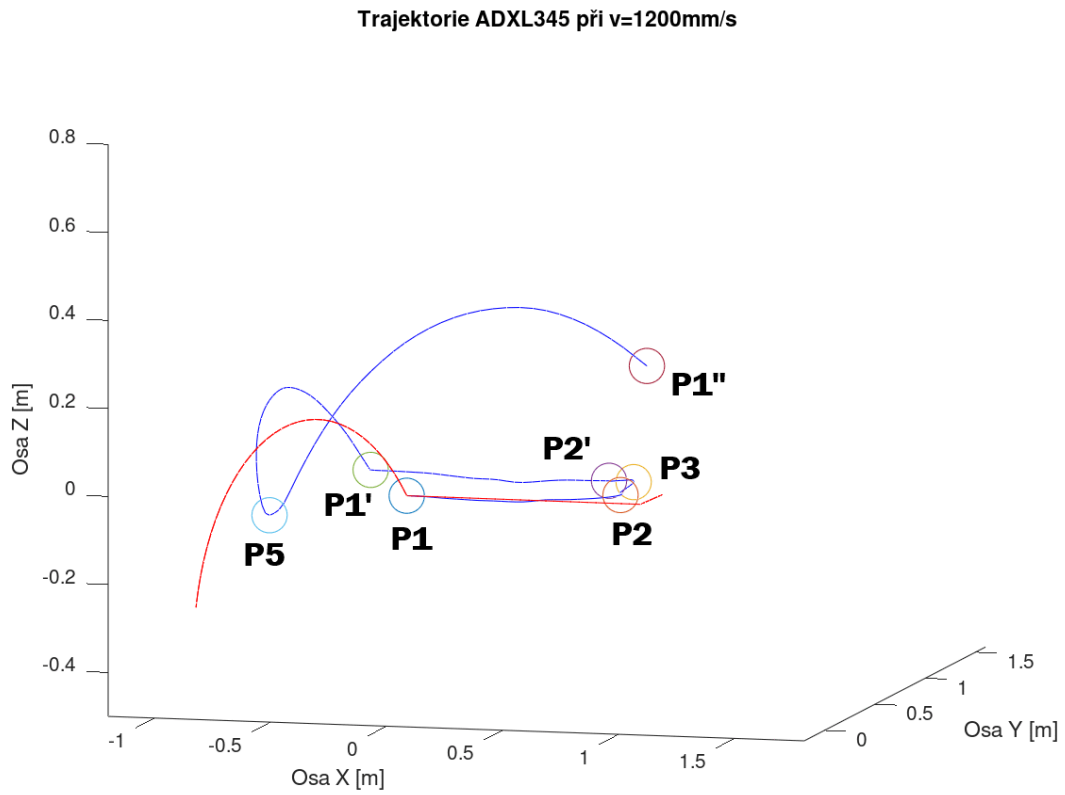
Graf 4:Trajektorie Reálného robotu při rychlosti 1400 mm/s (modrá) a simulace při rychlosti 1400 mm/s (červená). Použitý akcelerometr ADXL 345.

Souřadnice z reálného robotu pro v= 1400 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	-0,04216	0,95797	0,07309
P3	0,08452	0,95620	0,06067
P2'	-0,11091	0,91670	0,00914
P1'	0,00441	-0,15068	-0,13449
P5	0,22698	-1,27762	-0,40466
P1"	0,57569	-0,48099	-0,03358

Tabulka 12: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 1400 mm/s. Použitý akcelerometr ADXL345.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,29653	0,01989	-0,07310	0,30605
0,35705	-0,01515	-0,06069	0,36249
0,36528	0,06116	-0,00915	0,37048
-0,00441	0,15068	0,13449	0,20202
-0,44246	0,38477	0,13050	0,60071
-0,57569	0,48099	0,03358	0,75093
		průměr	0,43211

Tabulka 13: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 1400 mm/s. Použitý akcelerometr ADXL 345.



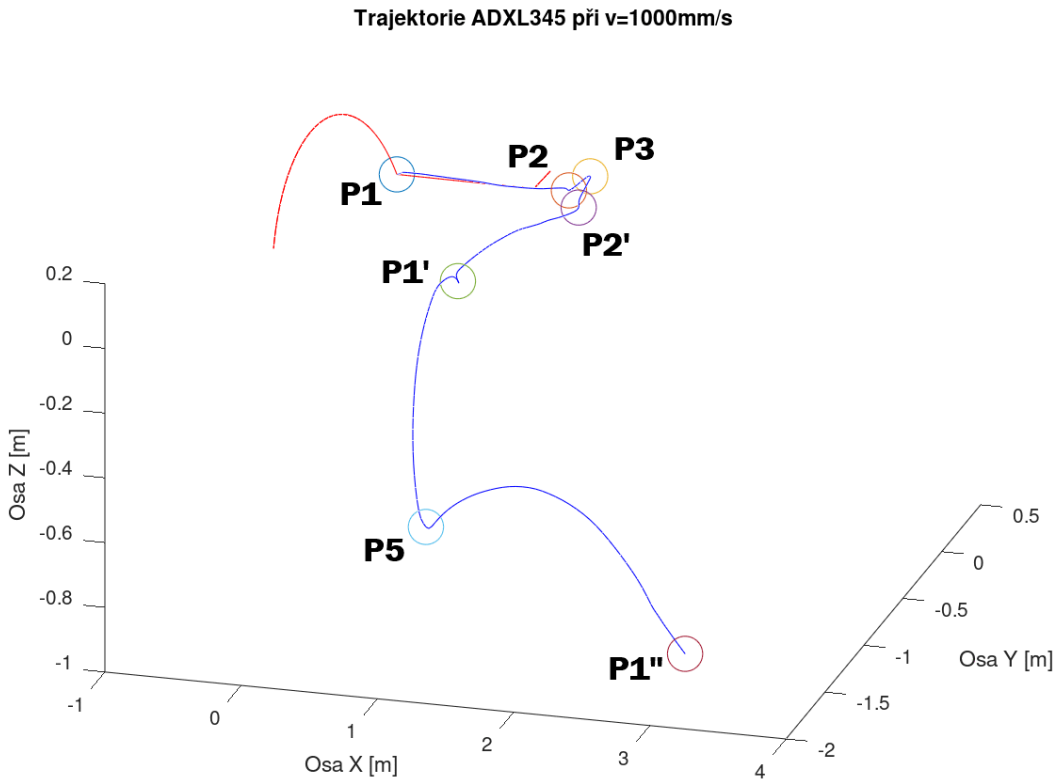
Graf 5: Trajektorie Reálného robotu při rychlosti 1200 mm/s (modrá) a simulace při rychlosti 1200 mm/s (červená). Použitý akcelerometr ADXL 345.

Souřadnice z reálného robotu pro v= 1200 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	0,03136	0,90774	0,01456
P3	0,22991	0,87817	0,01993
P2'	0,05446	0,84811	0,04323
P1'	0,07024	-0,18586	0,04696
P5	0,67393	-0,88362	-0,14048
P1"	1,46361	0,40017	0,12899

Tabulka 14: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 1200 mm/s. Použitý akcelerometr ADXL345.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,22301	0,07012	-0,01457	0,23423
0,21166	0,06288	-0,01995	0,22170
0,19991	0,12975	-0,04324	0,24222
-0,07024	0,18586	-0,04696	0,20416
-0,88941	-0,00923	-0,13368	0,89944
-1,46361	-0,40017	-0,12899	1,52280
		průměr	0,55409

Tabulka 15: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 1200 mm/s. Použitý akcelerometr ADXL345.



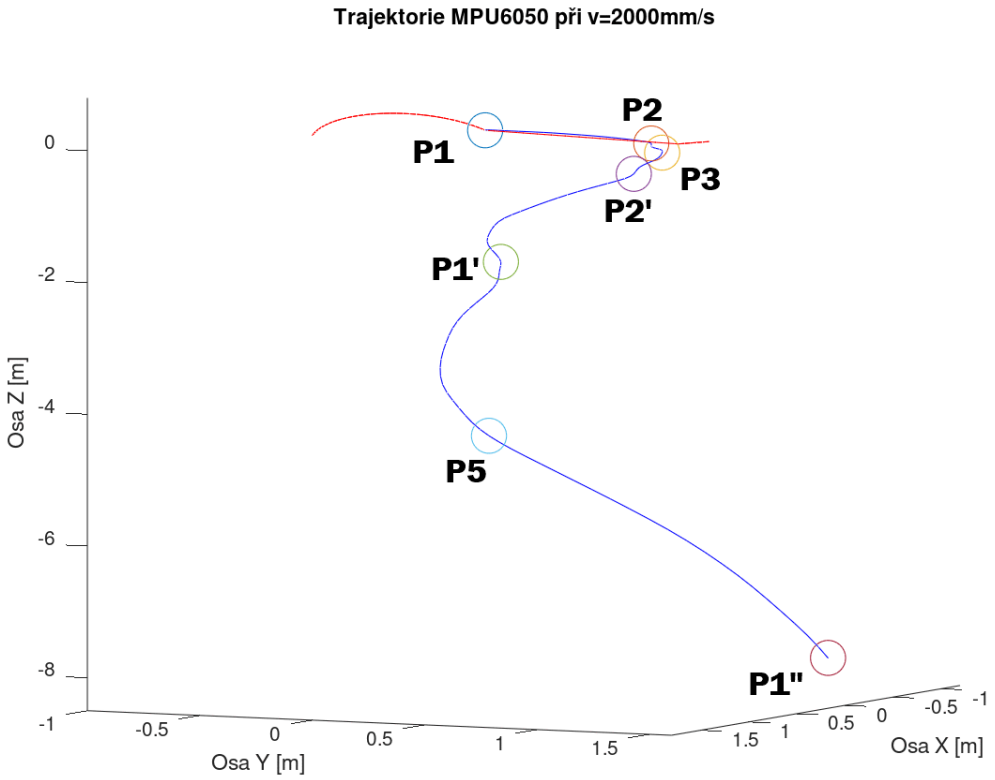
Graf 6: Trajektorie Reálného robotu při rychlosti 1000 mm/s (modrá) a simulace při rychlosti 1000 mm/s (červená). Použitý akcelerometr ADXL 345.

Souřadnice z reálného robotu pro v= 1000 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	-0,16515	1,35746	0,05444
P3	-0,05978	1,45357	0,07144
P2'	-0,36778	1,54518	0,06791
P1'	-0,71053	0,85457	-0,08885
P5	-1,11891	0,85313	-0,73005
P1"	-1,64217	3,05472	-0,87966

Tabulka 16: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 1000 mm/s. Použitý akcelerometr ADXL345.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,41952	-0,37960	-0,05445	0,56839
0,50135	-0,51252	-0,07146	0,72051
0,62215	-0,56732	-0,06792	0,84471
0,71053	-0,85457	0,08885	1,11491
0,90343	-1,74598	0,45589	2,01803
1,64217	-3,05472	0,87966	3,57796
		průměr	1,47409

Tabulka 17: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 1000 mm/s. Použitý akcelerometr ADXL345.



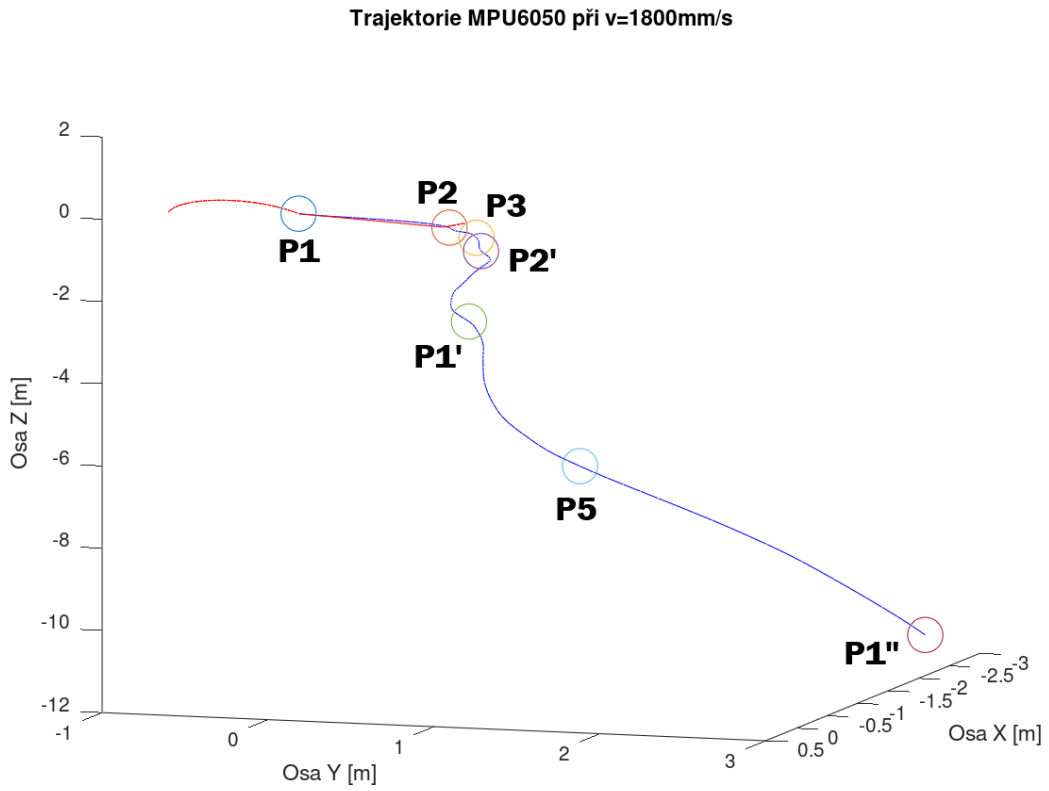
Graf 7: Trajektorie Reálného robotu při rychlosti 2000 mm/s (modrá) a simulace při rychlosti 2000 mm/s (červená). Použitý akcelerometr MPU6050.

Souřadnice z reálného robotu pro v= 2000 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	0,07275	0,77060	-0,07106
P3	-0,01692	0,78061	-0,23153
P2'	0,20735	0,75047	-0,50109
P1'	0,19886	0,15527	-1,92493
P5	-0,14452	-0,04486	-4,67989
P1"	-0,47958	1,32095	-7,93969

Tabulka 18: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 2000 mm/s. Použitý akcelerometr MPU6050.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0,00000	0,00000	0,00000	0
0,18162	0,20726	0,07105	0,28459
0,45849	0,16044	0,23151	0,53810
0,04702	0,22739	0,50108	0,55227
-0,19886	-0,15527	1,92493	1,94140
-0,07096	-0,84799	4,40573	4,48716
0,47958	-1,32095	7,93969	8,06310
		průměr	2,64444

Tabulka 19: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 2000 mm/s. Použitý akcelerometr MPU6050.



Graf 8: Trajektorie Reálného robotu při rychlosti 1800 mm/s (modrá) a simulace při rychlosti 1800 mm/s (červená). Použitý akcelerometr MPU6050.

Souřadnice z reálného robotu pro v= 1800 [mm/s]			
Název bodu	x [m]	y [m]	z [m]
P1	0	0	0
P2	-0,27207	0,80941	-0,36597
P3	-0,54522	0,87154	-0,76775
P2'	-0,51698	0,90935	-1,06769
P1'	-1,21028	0,58147	-3,25712
P5	-2,24513	0,87026	-7,35309
P1"	-2,75453	2,76366	-11,44179

Tabulka 20: Souřadnice významných bodů trajektorie získané z měření reálného robotu pro rychlost 1800 mm/s. Použitý akcelerometr MPU6050.

Rozdíl souřadnic bodů simulace a reálného robotu			
rozdíl x [m]	rozdíl y [m]	rozdíl z [m]	Vzdálenost bodů [m]
0	0	0	0
0,52644	0,16845	0,36596	0,66291
0,98679	0,06951	0,76773	1,25219
0,77135	0,06851	1,06768	1,31894
1,21028	-0,58147	3,25712	3,52302
2,02965	-1,76311	7,07893	7,57227
2,75453	-2,76366	11,44179	12,08883
		průměr	4,40303

Tabulka 21: Rozdíly souřadnic bodů trajektorie a jejich vzdálenost pro simulaci a reálný pohyb pro rychlost 1800 mm/s. Použitý akcelerometr MPU6050.

5.8.2 Diskuse dosažených výsledků

Zpracování dat bylo přerušeno po konzultaci s vedoucím práce. Vyhodnotili jsme, že již nemá smysl dále zpracovávat data, která jsou od počátku zatížena velkou chybou, která činí v průměru 1025 mm. Tato chyba je způsobená nepřesností použitého měřícího zařízení. Dá se vypozorovat, že akcelerometr MPU6050 poskytnul řádově horší výsledky, než 10x dražší akcelerometr ADXL345, konkrétně 3524 mm oproti 575 mm.

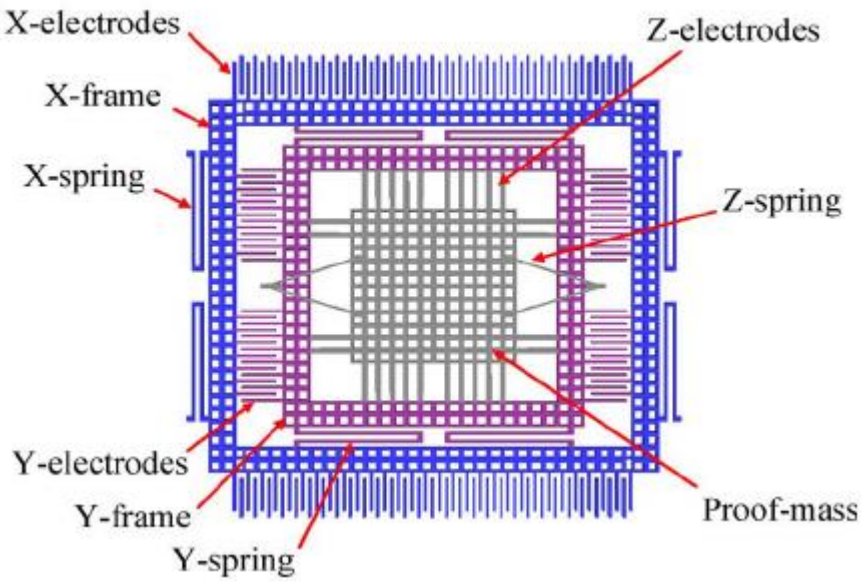
Přesnost použitých akcelerometrů se tedy ukázala pro naše účely jako zcela nedostatečná. Původní očekávání, založené na zkušenostech s profesionálními senzory

polohy, bylo maximálně 5 mm pro celou trajektorii. Dosažená chyba se však pohybuje průměrně o dva řády výše.

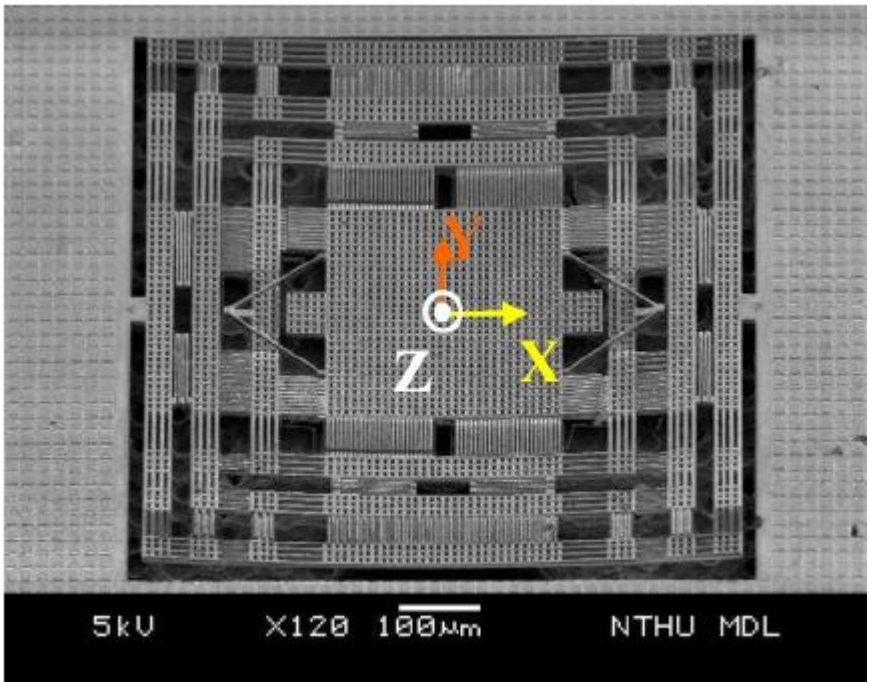
Dále jsem vypozoroval, že chyba pozice z měření reálného pohybu je nepřímo úměrná maximální nastavené rychlosti. Z pozorování pohybu mezi body P1 a P2 lze odhadnout, že akcelerometr pracuje mnohem lépe, když se pohybuje jen v jedné ose. Pokud se ovšem přidá pohyb v dalším směru, dojde ke projevení tzv. *crosstalk.* Crosstalk znamená, že akcelerometr může detekovat zrychlení v jiném směru v důsledku spojení mezi různými snímacími osami

Pro ilustraci uvádím obrázky 30 a 31, na kterých je schéma konstrukce a reálné zařízení pod elektronovým mikroskopem. Je zde patrné, že u 3-osého akcelerometru je měření jednotlivých os mechanicky spojené pružinami, proto se projevuje chyba *crosstalk.*

Tomuto nasvědčuje skutečnost, že se velké chyby objeví až po projetí bodem P2, kdy začne robot měnit směry



Obrázek 30: schéma reálné konstrukce 3-osého akcelerometru, (Weileun Fang, 2010)



Obrázek 31: 3-osý akcelerometr pod elektronovým mikroskopem SEM, (Weileun Fang, 2010)

Po diskusi s vedoucím práce jsem dohledával k uvedeným chybám další informace. Zjistil jsem, že u levných akcelerometrů dochází i k tzv. „*driftu*“, Při tomto jevu dojde k neúplnému navrácení mečící hmoty do původní pozice a akcelerometr detekuje zrychlení, i když se již nepohybuje. *Drift* je převážně způsoben levnou a méně přesnou technologií výroby a také vysokou teplotní závislostí. Konstrukce levných snímačů je tvořena s hlavním požadavkem nízké ceny, nikoliv co nejpřesnějšího měření. Chyba je způsobena nedostatečně odolnou konstrukcí proti deformaci snímače.

Za další možné vnesené chyby považuji:

- Špatná kalibrace.
- Nedostatečná kompenzace teploty.
- Okolní signálový šum (zdroj nejspíše rozvaděč robotu).

5.9 Časová analýza

Porovnání časů u průjezdů bodů reálného robotu a simulace poskytuje důležité informace o tom, jak odpovídá čas cyklu v simulaci reálnému pohybu. Více informaci o tom, proč je čas pohybu tak důležitý uvádím v kapitole **4.1.1.3 Čas cyklu**.

Data byla získána na základě analýzy signálu *trigger* a času. Vždy, když dojde ke změně *triggeru* z hodnoty 0 na hodnotu 1, jedná se o indikaci, že robot právě dorazil do požadovaného bodu. Délka pulzu musela být zvolena tak, aby nedocházelo k „překryvu" pulzů, ale ani ztrátě informace kvůli použité frekvenci vyčítání.

V tabulce níže je příklad zpracování dat, kde je označen jeden ze tří paternů hodnot *triggeru* 0→1. Dle pořadí nalezených paternů byl přiřazen příslušný bod trajektorie.

22	5715	-1,4721	2,0449	-1,3405	0	
23	5745	-0,4334	5,2764	0,3137	0	
24	5776	0,99	8,5464	-0,1094	0	
25	5806	-0,1256	9,2389	0,9677	0	
26	5836	0,2591	5,2764	-0,3787	1	P2
27	5866	-0,4718	-0,1864	1,3909	1	
28	5897	-5,1653	-0,0325	-0,5326	1	
29	5926	-13,5903	-1,1866	0,8523	1	
30	5957	-11,5129	-0,3018	-1,6867	1	
31	5987	-0,241	-0,9173	0,7369	1	
32	6018	10,9539	0,66	-0,9173	1	
33	6047	10,6077	-0,071	1,6602	0	
34	6078	8,4918	0,6215	-0,9943	1	P3
35	6108	-0,5103	-0,2633	0,7754	1	
36	6138	2,375	0,006	-0,725	1	
37	6168	11,7233	0,3907	1,0062	1	
38	6199	10,8385	0,8523	-0,2249	1	
39	6228	8,4918	-0,3403	0,0444	1	
40	6259	-9,4355	0,5446	-1,0712	1	
41	6289	-12,9363	-0,9173	1,3909	0	
42	6320	-11,2821	-0,2633	-0,4557	0	
43	6349	-1,2797	0,3137	1,1986	1	P2
44	6380	0,4899	-0,071	-0,1479	1	

Obrázek 32: Příklad analýzy dat pro srovnání časů průjezdů body

a) **Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 200 [mm/s].**

Jedná se o nejnižší testovanou rychlost, která se se používá například při velmi přesné manipulaci.

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	5184	5056	128
P3	6240	6088	152
P2‘	7266	7120	146
P1‘	12394	12212	182
P5	18366	18133	233
P1‘‘	24398	24052	346

Tabulka 22: Porovnání časů při rychlosti 200 [mm/s].

b) **Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 400 [mm/s].**

Rychlost 400 [mm/s] se používá například při odjezdu od formy po uchopení dílů, kde je zapotřebí zvýšená opatrnost: například, když je díl náchylný na prasknutí.

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	2655	2548	107
P3	3228	3100	128
P2‘	3770	3652	118
P1‘	6393	6244	149
P5	9439	9249	190
P1‘‘	12486	12256	230

Tabulka 23: Porovnání časů při rychlosti 400 [mm/s].

c) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 600 [mm/s].

Tato rychlost se používá například pro transfer větších dílů. Ty by při větších rychlostech měly tendenci se ohýbat při pohybu.

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	1812	1724	88
P3	2233	2128	105
P2‘	2655	2532	123
P1‘	4433	4304	129
P5	6485	6341	144
P1‘‘	8539	8376	163

Tabulka 24: Porovnání časů při rychlosti 600 [mm/s].

d) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 800 [mm/s].

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	1418	1316	102
P3	1780	1648	132
P2‘	2112	1984	128
P1‘	3502	3348	154
P5	5071	4905	166
P1‘‘	6702	6460	242

Tabulka 25: Porovnání časů při rychlosti 800 [mm/s].

e) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 1000 [mm/s]

Univerzální, ve většině nenáročných automatizací dostačující rychlost pro celý cyklus.

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	1177	1076	101
P3	1479	1368	111
P2‘	1781	1664	117
P1‘	2927	2788	139
P5	4194	4061	133
P1‘‘	5521	5332	189

Tabulka 26: Porovnání časů při rychlosti 1000 [mm/s].

f) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 1200 [mm/s]

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	997	916	81
P3	1269	1184	85
P2‘	1571	1456	115
P1‘	2567	2428	139
P5	3655	3513	142
P1‘‘	4770	4596	174

Tabulka 27: Porovnání časů při rychlosti 1200 [mm/s].

g) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 1400 [mm/s]

Jedná se o již poměrně vysokou rychlost, například pro převoz menších dílů, kde je rychlý takt IMM.

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	905	820	85
P3	1178	1076	102
P2‘	1449	1336	113
P1‘	2325	2208	117
P5	3291	3169	122
P1‘‘	4288	4128	160

Tabulka 28: Porovnání časů při rychlosti 1400 [mm/s].

h) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 1600 [mm/s]

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	845	752	93
P3	1087	1000	87
P2‘	1329	1248	81
P1‘	2175	2052	123
P5	3050	2933	117
P1‘‘	3956	3816	140

Tabulka 29: Porovnání časů při rychlosti 1600 [mm/s].

ch) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 1800 [mm/s]

Vysoká rychlost, vhodná například při přejezdech s díly, když je zapotřebí vykonat více operací, než jen odložení dílu na dopravníkový pás. Například jízda před kamerovou kontrolu, či před štítkovací automat.

Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
P1	0	0	0
P2	815	708	107
P3	1057	952	105
P2‘	1297	1196	101
P1‘	2082	1956	126
P5	2925	2781	144
P1‘‘	3801	3604	197

Tabulka 30: Porovnání časů při rychlosti 1800 [mm/s].

i) Srovnání časů průjezdů body reálného robotu a simulace při maximální rychlosti 2000 [mm/s]

Velmi vysoká rychlost, které robot dosáhne jen při jízdě na delší dráze.

Pořadí bodu trajektorie	Název Bodu	Čas průjezdu bodem (Měřící zařízení) [ms]	Čas průjezdu bodem (Simulace) [ms]	Rozdíl [ms]
0	P1	0	0	0
1	P2	754	628	126
2	P3	996	872	124
3	P2‘	1267	1116	151
4	P1‘	1992	1848	144
5	P5	2807	2633	174
6	P1‘‘	3592	3416	176

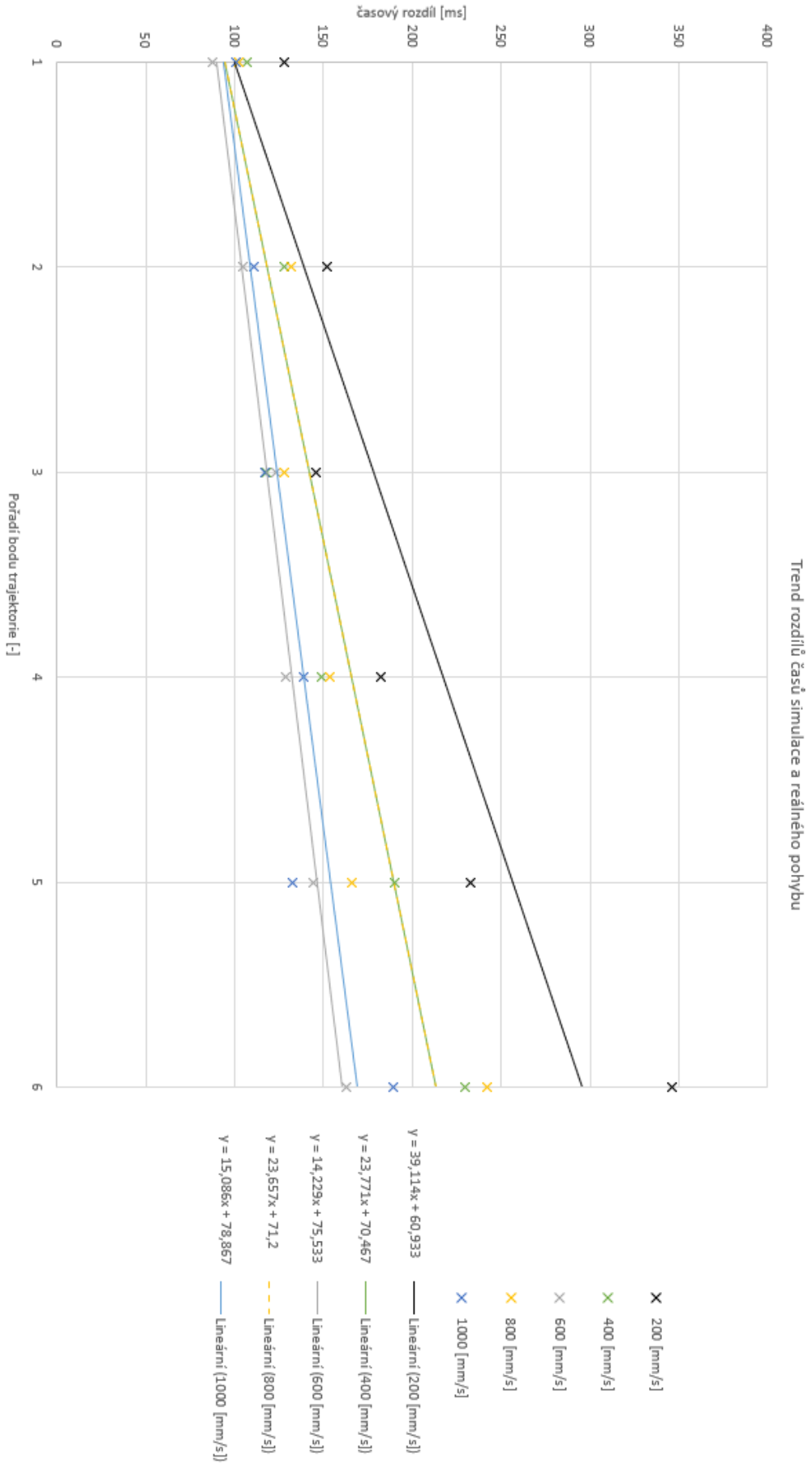
Tabulka 31: Porovnání časů při rychlosti 2000 [mm/s].

Hodnoty rozdílů časů průjezdů byly vyneseny do dvou grafů:

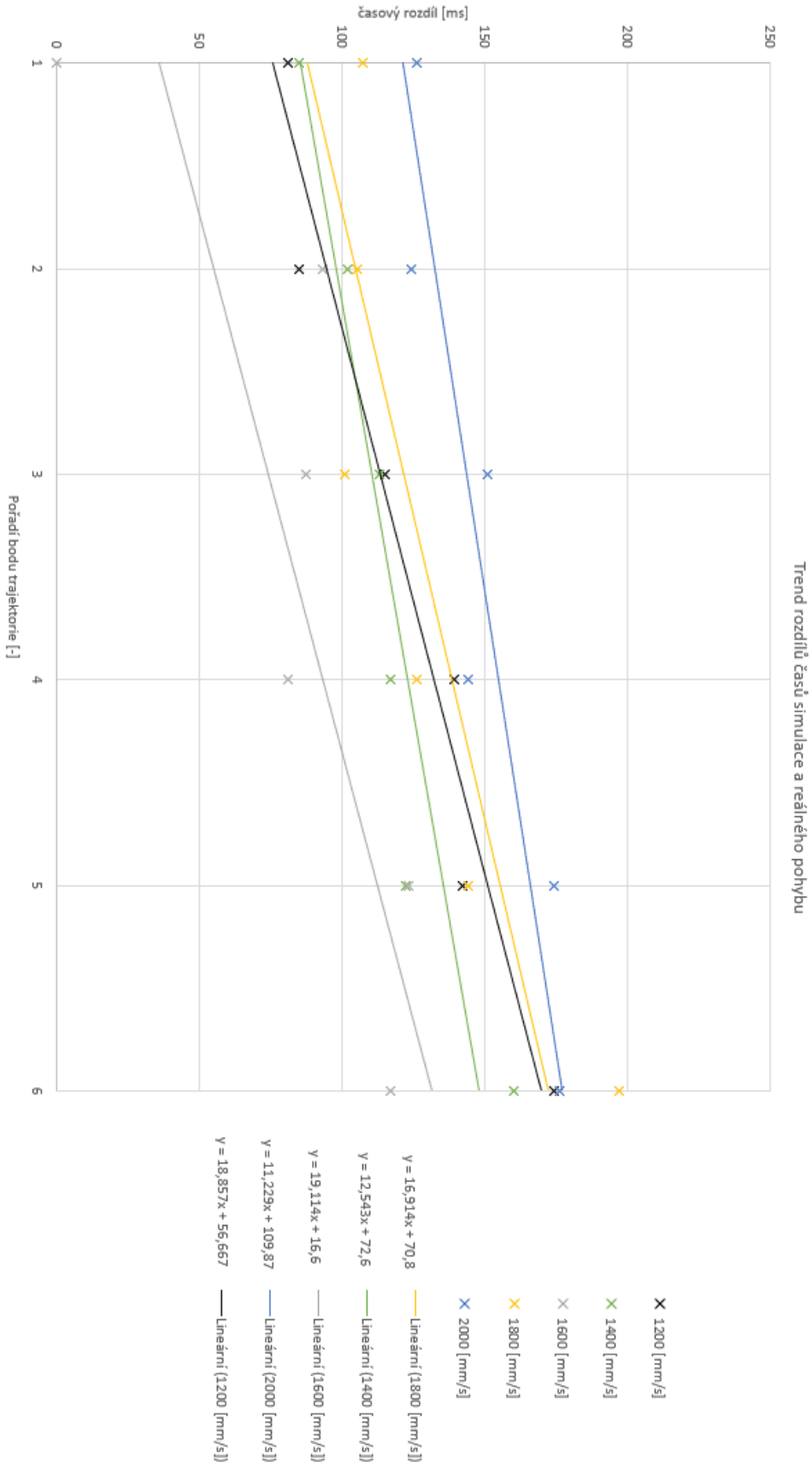
a) Graf 9: Trend rozdílů časů simulace a reálného pohybu pro rychlosti 200, 400,1600, 800 a 1000 [mm/s].
b) Graf 10: Trend rozdílů časů simulace a reálného pohybu ro rychlosti 1200, 1400, 1600, 1800 a 2000 [mm/s],

kde na ose x jsou vyneseny body bez nultého bodu P1, protože v tomto bodě trajektorie začíná, proto je zde časový rozdíl nulový, a tedy tato hodnota nenese žádnou přínosnou informaci pro další zpracování.

Dále byly hodnoty pro jednotlivé experimenty proloženy lineární funkcí. Rovnice těchto přímek jsou také vyobrazeny v grafu.



Graf 9: Trend rozdílů časů simulace a reálného pohybu pro rychlosti 200, 400,1600, 800 a 1000 [mm/s].



Graf 10: Trend rozdílů časů simulace a reálného pohybu ro rychlosti 1200, 1400, 1600, 1800 a 2000 [mm/s].

Koeficienty lineárních aproximací byly zaznamenány do tabulky.

Maximální rychlost [mm/s]	Lineární člen aproximace	Absolutní člen aproximace
200	39,114	60,933
400	23,771	70,457
600	14,229	75,533
800	23,657	71,200
1000	15,086	78,867
1200	18,857	56,667
1400	12,543	72,600
1600	19,114	16,600
1800	16,914	70,800
2000	11,229	109,870

Tabulka 32: Koeficienty lineárních aproximací rozdílů časů průjezdů body trajektorie v simulaci a reálného robotu.

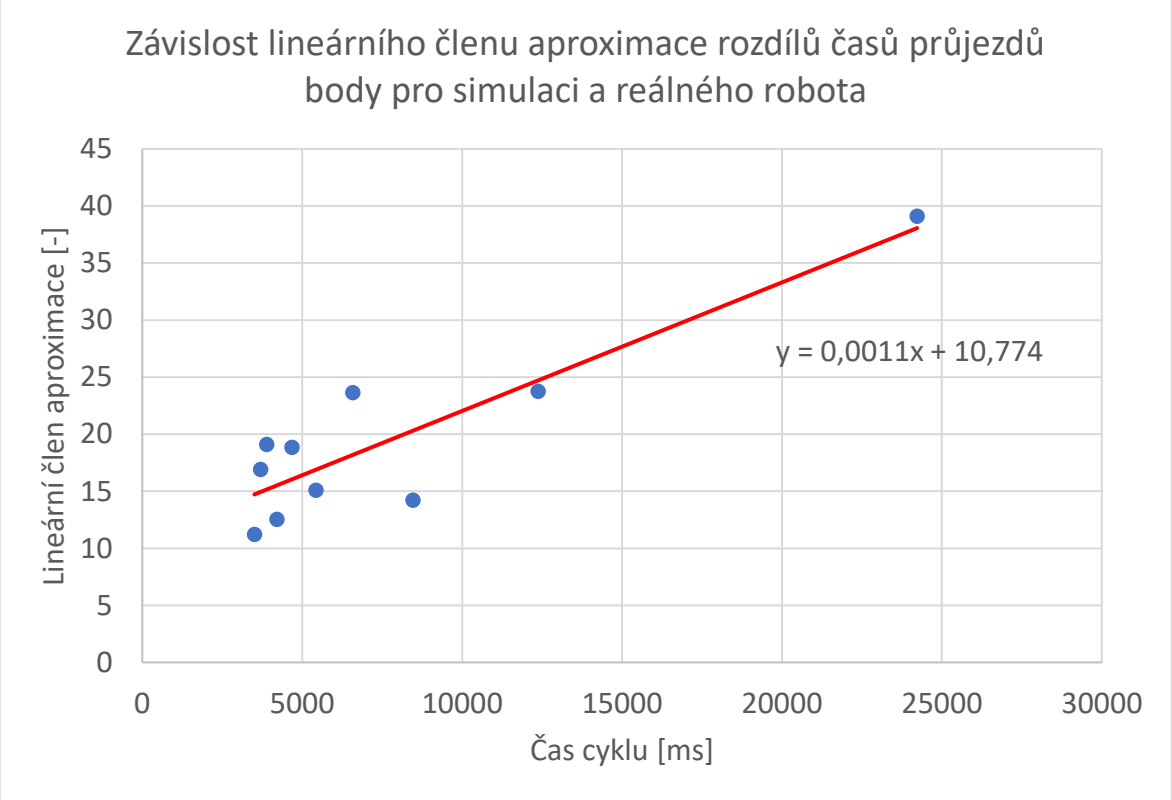
Všechny absolutní členy jsou kladné, což jasně indikuje, že se provoz reálného robotu se opožďuje oproti simulaci. Absolutní členy jsou ve většině případů podobné a jejich průměrná hodnota z deseti měření je 68,353. Lze tedy konstatovat, že opoždění provozu reálného robotu není závislé na maximální provozované rychlosti.

Lineární členy aproximací jsou též kladné, to udává, že se robot bude opožďovat oproti simulaci v čase čím dál více. Toto tvrzení lze podpořit i zjištěním, že pro velmi nízkou rychlost [200 mm/s] dosahuje lineární člen velmi vysokých hodnot a to 39,114. Toto je způsobeno tím, že experiment trval nejdéle, a to téměř 25 s oproti většině ostatních cyklů, které trvaly řádově jednotky sekund.

Pro zkoumání dalších závislostí vyneseme hodnoty lineárních členů aproximací v závislosti na délce cyklu, viz graf 11. Čas cyklu byl spočítán, jako aritmetický průměr času cyklu naměřeného pro simulaci a času cyklu reálného robota.

Maximální rychlost [mm/s]	Lineární člen aproximace [-]	Čas cyklu [ms]
200	39,114	24225
400	23,771	12371
600	14,229	8458
800	23,657	6581
1000	15,086	5427
1200	18,857	4683
1400	12,543	4208
1600	19,114	3886
1800	16,914	3702
2000	11,229	3504

Tabulka 33: Hodnoty lineárních členů aproximace a časy cyklu.



Graf 11: Závislost lineárního členu aproximace rozdílů časů průjezdů body pro simulaci a reálného robota.

Z trendu grafu 11 lze usoudit, že opoždění reálného robotu oproti simulaci bude opravdu růst jen lineárně.

Maximální rychlost [mm/s]	čas cyklu [ms]	zpoždění [ms]	Zpoždění za 1 s cyklu [ms]
200	24225,0	346	14,3
400	12371,0	230	18,6
600	8457,5	163	19,3
800	6581,0	242	36,8
1000	5426,5	189	34,8
1200	4683,0	174	37,2
1400	4208,0	160	38,0
1600	3886,0	140	36,0
1800	3702,5	197	53,2
2000	3504,0	176	50,2
		Průměr:	33,8

Tabulka 34: Tabulka hodnot a výpočet zpoždění za 1 s cyklu.

V tabulce č.34 jsou časy cyklů pro jednotlivé experimenty, kde čas cyklu byl vypočten jako aritmetický průměr času cyklu reálného robotu a času cyklu simulace. Tyto časy byly získány na základě detekce signálu Trigger při průjezdu posledním bodem trajektorie. Vypočtené hodnoty Zpoždění za 1 s cyklu [ms] jsou výsledkem poměru zpoždění ku času cyklu násobené 1000, aby byl výsledek taktéž v milisekundách. Z těchto hodnot byl opět vypočten aritmetický průměr. Tato tabulka slouží ke zjištění průměrného zpoždění reálného robotu oproti simulaci za 1 s cyklu.

6 Diskuse

6.1 Výsledky polohové analýzy

Polohová analýza dosahuje nejlepších výsledků při průjezdu trajektorie při rychlostech 1800 a 2000 [mm/s]. Tyto dvě maximální nastavené rychlosti jsou velmi podobné, protože robot při nastavení 2000 [mm/s] nedokázal této rychlosti ani docílit, nebo jí docílil na velmi krátkou dobu (řádově milisekundy). Proto jsou tyto trajektorie přibližně shodné.

I přes to, že se jedná o nejpřesnější rekonstrukci původní dráhy, chyba přesahuje vice jak 10^4 deklarovanou přesnost robotu. Dá se ovšem vypozorovat, že použité sensory poskytují lepší výsledky při vysokých rychlostech a s klesající rychlostí přesnost rekonstrukce klesá.

Dále se dá vypozorovat, že záleží na cenové kategorii použitých měřidel. Data z nejlevnějšího MPU6050 nebyla vůbec použitelná (chyba v průměru činila 3524 mm), zato data získaná z ADXL345 byla znatelně lepší (chyba v průměru činila 575 mm). I přesto je lze stále považovat za nedostatečná pro tento typ analýzy, kde bychom očekávali přesnost alespoň 10 mm.

Pro polohovou analýzu by byl zapotřebí buď profesionální akcelerometr s přesností alespoň 10 μg, popřípadě tři jednoosé akcelerometry, aby byly všechny tři osy separované a nebylo možné jejich ovlivnění. Takové akcelerometry se pohybují řádově v desítkách až stovkách tisíc českých korun, například MEMS akcelerometr MTI-670 lze pořídit za 40.000Kč.

Z grafů je a dat je možné vypozorovat, kde vzniká největší chyba rekonstrukce. Při pohybu akcelerometru jen jedním směrem (v trajektorii první pohyb z bodu P1 do P2) je vidět, že je rekonstrukce relativně přesná. Ovšem, když se přidají pohyby v dalších osách dojde k obrovským chybám. Toto je nejspíše způsobeno levnou konstrukcí MEMS viz Obrázek 30: schéma reálné konstrukce 3-osého akcelerometru

Dalším z důvodů, který vedl k získání naměřených dat, je skutečnost, že akcelerometr při pohybu tam a zpět ještě v „setrvačnosti“ ukazuje hodnoty zrychlení, i když je již v klidu. Tato chyba se samozřejmě projeví při integrování, kde je velmi znát, jestli zůstane na akcelerometru nějaké „zbytkové“ zrychlení a je s ním počítáno v každém kroku integrace, místo toho, aby byl robot v klidu.

6.2 Výsledky časové analýzy

Časová analýza měla být dle zadávacího protokolu hlavním bodem této práce. Získané výsledky byly více než zajímavé. Vyřknutá hypotéza na začátku této práce, pojednávající o skutečnosti, že se bude provoz reálného robotu zpožďovat oproti robotu v simulaci při průjezdu trajektorií, se potvrdila. Robot opravdu v reálném provozu projíždí body trajektorie s časovým zpožděním. Ovšem toto zpoždění je řádově desítky, až stovky *[ms]*. To poukazuje na kvalitu simulačního software, kde modely robotů budou nejspíše velmi věrohodně zpracované bez výrazných zjednodušení, či vhodně použité kompenzace.

I přes velmi kvalitní simulační software, lze tvrdit, že se časový posun průjezdu body trajektorie oproti simulaci "zhoršuje" lineárně. Dále lze vypozorovat, že není vidět žádná jasná indikace, která by poukazovala na souvislost zpožďování s použitou maximální rychlostí. I když jsou hodnoty různé pro různé rychlosti, nenaznačuje to, že by se projevovala sama maximální rychlost, ale délka celého cyklu.

Z měření lze vypozorovat následující závěr:

Vysoká rychlost ⇒ krátký čas projetí celé trajektorie ⇒ malé opoždění

Nízká rychlost ⇒ vysoký čas projetí celé trajektorie ⇒ velké opoždění

Je tedy zřejmé, že s delším cyklem robotu bude hodnota opoždění lineárně stoupat. Protože všechny vypočtené křivky mají kladný lineární koeficient (viz. Graf 9 a 10). Můžeme tvrdit, že pro krátké cykly čas simulace odpovídá časům reálného provozu. Ovšem u delších cyklů delších než cca 15 s je zapotřebí s touto chybou počítat a odpovídajícím způsobem ji kompenzovat. To lze učinit např. nastavením vyšší maximální rychlosti, nastavením vyšší akcelerace a decelerace robotu, nebo se může upravit trajektorie tak, aby se kompenzoval rozdíl časů. Takováto optimalizace se nejčastěji provádí úpravou trajektorie ve smyslu zkrácení dráhy, nebo snížení přesnosti průjezdů body.

6.3 Řešené problémy

V průběhu mé práce jsem narazil na několik zajímavých problémů, které bylo třeba vyřešit, aby bylo možné se dobrat výsledku. Zde uvedeme ty hlavní:

a) Data z gyroskopu

Data o aktuálním sklonu nebylo možné použít, protože gyroskop, který je součástí MPU6050 sice dosahuje přesností deklarovaných v *datasheetu*, ovšem jen při velmi pomalých změnách orientace.

Tuto skutečnost jsem zjistil během experimentu a následně jsem si í potvrdil krátkými, nezávislými testy. Opakovaně jsem prováděl rychlé nakloněním senzoru o úhel 45° kolem osy X a vrácení do původní polohy. Trvalo několik vteřin, než se hodnota ustálila, a to ještě nikoliv na původní hodnotě. Dále jsem prováděl test naklonění senzoru o úhel 45° kolem osy X a provedení translačního pohybu vpřed a vzad ve směru osy Y pod stejným úhlem. V tomto případě sensor detekoval až o 20 stupňů více než byla reálná hodnota.

Data z gyroskopu, byla vyhodnocena jako nedostačující pro naší aplikaci. Byla navržena trajektorie, kde nebude docházet k náklonu měřícího zařízení a bude tedy možné odečíst gravitační zrychlení jen od osy Z.

b) I2C

Při konstrukci měřícího zařízení se naskytl problém se zapojením. Oba senzory ADXL345 i MPU6050 lze provozovat na stejné sběrnici a také mají obě zařízení rozdílnou adresu 0x68 a 0x53. Ovšem při zapojení obou senzorů na stejnou linku došlo k problému, že ani jeden ze senzorů nebyl detekován. Pro detekci zařízení na sběrnici byl použit vzorový program i2c_scanner z knihovny Wire.h.

c) Jiný robot během vypracování práce

Práci dále komplikovala skutečnost, že během vypracování došlo z důvodu rozhodnutí managementu firmy k výměně původně použitého typu robotu ABB IRB 4800 za nový ABB IRB 1600. To vneslo do práce jisté zdržení a nutnost opravit, předělat a nově naprogramovat.

7 Závěr

Podařilo se navrhnout a sestrojit měřící zařízení pro měření zrychlení a průjezdů body trajektorie reálného robotu. S daty z provedených experimentů na reálném robotu a ze simulace, byla provedena dvojí analýza.

Polohovou analýzou se podařilo zjistit, z dat zaznamenaných běžnými akcelerometry, že není možné zrekonstruovat dráhu pohybu robotu s požadovanou přesností. Tato analýza také ukázala, že data získaná z kvalitnějšího akcelerometru (ADXL345), jsou podstatně přesnější a lépe zpracovatelná, což poukazuje na potřebu zakoupení vhodnějších průmyslových, či laboratorních akcelerometrů, aby byla možná rekonstrukce pohybu s požadovanou přesností.

Z časové analýzy, která byla stěžejní pro tuto práci, se dostalo těchto závěrů:

a) Byla potvrzena hypotéza, že pohyb reálného robotu se bude opožďovat oproti pohybu robotu v simulaci.
b) Dále bylo zjištěno, že toto zpoždění je přibližně 34 ms za 1 s cyklu. To může být v reálné praxi zanedbatelná hodnota pro krátké cykly (např. <15 s). Pro cyklus, který bude delší, než 15 s, by zpoždění reálného robotu oproti simulaci bylo větší, než 0,5 s. Proto je důležité tuto nově objevenou skutečnost brát v potaz a počítat s onou nepřesností simulace hlavně u delších cyklů.
c) Dalším zjištěním plynoucím z analýzy je to, že zpoždění roste lineárně a není závislé na použité maximální rychlosti robotu.

Jelikož se profesně věnuji instalaci, programování a školení robotů, výstupy této práce mi umožní v zaměstnání lépe využívat softwarovou simulaci jejich činnosti. Většina zákazníků, kteří roboty ve výrobě využívají, si před zavedením nové výroby či podstatnější změnou výrobního procesu, nechávají zpracovat analýzu přínosů těchto změn. Podstatnou součástí této analýzy může být právě provedení simulace činnosti robota: časová i polohová. Mezi očekávané přínosy patří optimalizace času výroby, prostorová náročnost robota, nutnost investic do obslužných zařízení (pásové dopravníky, odkládací plochy, ochranné zóny), energetické náklady. Toto vše simulace dokáže s určitou přesností určit.

Díky zjištěným poznatkům v této práci budu schopen při simulacích lépe odhadnout jejich nepřesnost, resp. věrohodnost s ohledem na konkrétní situace a

podmínky nového provozu. Ve výsledku tak bude možné lépe plánovat využití robotů pro reálný provoz.

8 Seznam obrázků

9 Seznam tabulek

10 Seznam příloh

Naměřená data ze simulace a reálného robotu jsou obsažena v příloze *Kocar_Vaclav_2023_DP_prilohy.zip*

11 Seznam literatury

1) ABB Robotics (2022) ‘Product specification IRB 1600/1660’. Available at: https://search.abb.com/library/Download.aspx?DocumentID=3HAC023604-001&LanguageCode=en&DocumentPartId=&Action=Launch.

2) Abdullah Al Mamun and Mr. Fakir Mashuque Alamgir (2017) ‘Flex Sensor Based Hand Glove for Deaf and Mute People’. nternational Journal of Computer Networks and Communications Security. Available at: https://www.researchgate.net/publication/325485280_Flex_Sensor_Based_Hand_Glove_for_Deaf_and_Mute_People.

3) Ahmed Jasim (2020) ‘Design and Implementation a New Real Time Overcurrent Relay Based on Arduino’. IOP Publishing.

4) ‘Akcelerometry MPU6050’ (2017). Wikipedia. Available at: http://wiki.sunfounder.cc/index.php?title=MPU6050_Module.

5) ‘Arduino’ (2017) *Wikipedia*. Wikipedia. Available at: https://cs.wikipedia.org/wiki/Arduino.

6) Arduino Official Store (2017) ‘Arduino’. Available at: https://store.arduino.cc/products/arduino-nano?queryID=undefined.

7) Havel, I.M. (1980) *Robotics: Introduction to the Theory of Cognitive Robots*. Praha: SNTL.

8) InvenSense Inc (2013) ‘MPU-6000 and MPU-6050 Product Specification Revision 3.4’. Available at: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

9) ITNetwork.cz and Čápka, D. (2020) ‘MPU6050 akcelerometr a gyroskop pro Arduino Zdroj: https://www.itnetwork.cz/hardware-pc/arduino/hardware/mpu6050-akcelerometr-a-gyroskop-pro-arduino’. ITNetwork.cz. Available at: https://www.itnetwork.cz/hardware-pc/arduino/hardware/mpu6050-akcelerometr-a-gyroskop-pro-arduino.

10) Jean-Louis Boimond (2020) ‘Calcul de la position du repère Outil par la méthode « XYZ 4 points »’. Université Angers. Available at: http://perso-laris.univ-angers.fr/~boimond/TP%202_Kuka_position_outil_XYZ_4_points.pdf.

11) Jeff Watson (2016) ‘MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments’. Analog devices.

12) Johan Gustafsson (2014) *Orientation estimation of a rigid multi body system using accelerometers, gyroscopes and the geometry*. Chalmers University of Technology. Available at: https://publications.lib.chalmers.se/records/fulltext/200469/200469.pdf.

13) Kocar, V. (2019) *Vytvoření robotické soustavy vykonávající motorickou činnost nohou a řízenou senzory umístěnými na lidském vodiči.* Jihočeská univerzita v Českých Budějovicích Přírodovědecká fakulta. Available at: https://dspace.jcu.cz/bitstream/handle/123456789/40767/BP_Kocar.pdf?sequence=1.

14) Krejsa, M. (2023) ‘Numerická integrace určitého integrálu’. Vysoká škola báňská – Technická univerzita Ostrava. Available at: http://fast10.vsb.cz/krejsa/studium/algoritmy_06.pdf.

15) Kučera, R. and Morávková, Z. (2015) ‘Numerická matematika’. Katedra matematiky a deskriptivní geometrie Vysoká škola bá ˇnská – Technická Univerzita Ostrava. Available at: http://homel.vsb.cz/~kuc14/textyNM/FINALNI_VERZE_CD.pdf.

16) Maik Schmidt (2015) *Arduino*. Germany: dpunkt.

17) ‘Manipulační průmyslové roboty - Slovník’ (1998). Česká technická norma. Available at: https://www.technicke-normy-csn.cz/csn-en-iso-8373-186501-169340.html.

18) Nof, S.Y. (1999) *Handbook of Industrial Robotics*. John Wiley & Sons,.

19) Pospíšil, J. (2020) ‘Elektrotechnika I.’ ČVUT v Praze, Fakulta elektrotechnická.

20) Ratiu, M., Alexandru Rus and Balas, M.L. (2018) ‘Modeling in ADAMS of 6R industrial robot’. MATEC Web of Conferences: Annual Session of Scientic Papers IMT ORADEA.

21) Skařupa, J. (2007) ‘Průmyslové roboty a manipulátory, učební text’. Available at: http://www.elearn.vsb.cz/archivcd/FS/PRM/Text/Skripta_PRaM.pdf.

22) Smutný, V. (2004) ‘Robotika’. České vysoké učení technické v Pra.

23) SparkFun Electronics (2023) ‘Akcelerometry MPU6050, ADXL345 Sparkfun’. Available at: https://www.sparkfun.com/.

24) Volf, L., Beránek, L. and Mikeš, P. (2010) ‘POČÍTAČOVÁ SIMULACE VE STROJÍRENSKÉ VÝROBĚ’. Ústav technologie obrábění, projektování a metrologie, Fakulta strojní, ČVUT v Praze. Available at: https://dspace5.zcu.cz/bitstream/11025/16403/1/Volf.pdf.

25) Vondrák, V. and Pospíšil, L. (2011) *Numerické metody I.* Západočeská Univerzita v Plzni.

26) VŠB, Technická Univerzita Ostrava (2023) ‘Gravitační pole’. Available at: https://www.studopory.vsb.cz/studijnimaterialy/Fyzikaprobakalare/PDF/1_5_Gravitacni_pole.pdf.

27) VTI Technologies (2005) ‘Cross-axis Compensation’. Available at: https://www.mouser.com/pdfdocs/an32_crossaxis_compensation.PDF.

28) Weileun Fang (2010) ‘Implementation of a Monolithic Single Proof-Mass Tri-Axis Accelerometer Using CMOS-MEMS Technique’. National Tsing Hua University. Available at: https://www.researchgate.net/publication/224139107_Implementation_of_a_Monolithic_Single_Proof-Mass_Tri-Axis_Accelerometer_Using_CMOS-MEMS_Technique.