VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Tadeáš Navrátil

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYSTÉM PRO SNÍMÁNÍ ÚHLU NATOČENÍ VOLANTU S VYUŽITÍM AKCELEROMETRŮ
ACCELEROMETER-BASED MONITORING OF STEERING WHEEL ANGLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tadeáš Navrátil**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ondrej Mihálik**
ADVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Tadeáš Navrátil ***ID:*** 211164
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Systém pro snímání úhlu natočení volantu s využitím akcelerometrů

POKYNY PRO VYPRACOVÁNÍ:

1. Navrhněte a zrealizujte zapojení, které umožní sbírat a ukládat data z vybraných snímačů ve formě časových řad.
2. Ověřte možnost rekonstruovat úhel natočení ze získaných dat – v prostředí MATLAB vytvořte algoritmus pro off-line výpočet úhlu natočení.
3. Optimalizujte umístnění snímačů i algoritmus zpracování signálů s cílem dosáhnout co nejmenší chyby rekonstrukce.
4. Navržené metody implementujte do mikrokontroléru pro účely rekonstrukce signálu v reálném čase. Zařízení otestujte v laboratorních podmínkách.
5. Vytvořte dokumentaci.

DOPORUČENÁ LITERATURA:

1. DROSESCU, Radu a Silviu ZAMFIR. MEMS based device for steering wheel angle experimental measuring. ICAEMM. 2016, s. 304–313. DOI: 10.1142/9789813146587_0045

2. MOUSSA, Mohamed et al. Steering Angle Assisted Vehicular Navigation Using Portable Devices in GNSS-Denied Environments. Sensors. 2019, roč. 19, č. 7. DOI: 10.3390/s19071618

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Ondrej Mihálik
Konzultant: Ing. Vlastimil Mancl

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zabývá návrhem zařízení pro snímání úhlu natočení volantu. Cílem bylo navrhnout základní koncept zařízení využívající data ze tří akcelerometrů a k tomuto zařízení navrhnout základní rekonstrukční algoritmus. V práci jsou postupně rozebrané různé verze rekonstrukčního algoritmu, které v sobě zahrnují i simulaci šumu reálných akcelerometrů. Rekonstrukční algoritmus byl realizován v prostředí MATLAB Simulink.

Dále se práce zabývá výběrem vhodných komponent. Jedná se především o výběr akcelerometrů, mikrokontroléru a volbu vhodné komunikační sběrnice pro vzájemnou komunikaci mezi nimi. Následně se práce věnuje vylepšením rekonstrukčního algoritmu a praktické realizace celého zařízení. V závěru porovnáváme výsledky výsledného zařízení se snímačem vestavěným v testovacím volantu.

KLÍČOVÁ SLOVA

Snímače, MEMS technologie, Akcelerometr, Gyroskop, Arduino, Datová sběrnice, I2C, SPI, MATLAB, Simulink, Python, Matematický model, Šum akcelerometrů, Měření, Úhel natočení, Mikrokontrolér

ABSTRACT

This thesis deals with the design of a device which will be used for measuring the steering wheel angle. Main goal was to design basic concept of device which is using data gathered from three accelerometers in the reconstruction algorithm to get steering wheel angle. Discussion of various reconstruction algorithms (with regards to simulated output noise levels from acceleromater) and final Implementation of reconstruction algorithm (in MATLAB simulink enviroment) is also part of this thesis.

Furthermore, thesis deals with the selection of suitable components. Mainly selection of accelerometers, microcontroller and the selection of a suitable communication bus for mutual communication between them. Thesis also deals with design improvements of algorithm and realization of the whole device.

In the end, we compare results of measuring device and inner senzor of steering wheel.

KEYWORDS

Sensors, MEMS technology, Accelerometer, Gyroscope, Arduino, Data bus, I2C, SPI, MATLAB, Simulink, Python, Mathematical model, Accelerometer noise, Measurement, Rotation angle, Microcontroller

NAVRÁTIL, Tadeáš. *Systém pro snímání úhlu natočení volantu s využitím akcelerometrů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2021, 82 s. Bakalářská práce. Vedoucí práce: Ing. Ondrej Mihálik

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Tadeáš Navrátil
VUT ID autora:	211164
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Systém pro snímání úhlu natočení volantu s využitím akcelerometrů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondreji Mihálikovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Cíl práce

Pokud chceme sledovat chování řidiče během řízení dopravního prostředku, potřebujeme mimo jiné sledovat úhel natočení volantu v závislosti na čase.

K snímání úhlu natočení volantu můžeme použít interní snímač zabudovaný přímo ve voze nebo externí snímač připevněný na volant. Používání externích snímačů má oproti interním mnoho výhod. Mimo jiné můžeme používat stejný systém nezávislý na automobilu, ve kterém budeme data snímat, takže nemusíme řešit komunikaci s ovládacími jednotkami jednotlivých automobilů, které se navzájem mohou velmi lišit. Výsledné zařízení bude univerzální a použitelné v libovolném automobilu.

Cílem této práce je tedy vytvořit zařízení schopné snímat natočení volantu v reálném čase. Řešení musí být nezávislé na platformě. To znamená, že zařízení lze použít na snímání libovolného volantu a nekomunikuje s žádným senzorem umístěným v automobilu. Úhel natočení volantu budeme snímat pomocí soustavy akcelerometrů, které doplníme o gyroskopy. K ovládání celého zařízení budeme používat mikrokontrolér eses klon Arduino UNO R3 CH340.

Tato bakalářská práce je součástí projektu datového koncentrátoru, který má za úkol sledovat chování řidiče při jízdě na dálnici. V něm se budou ukládat a analyzovat mimo jiné i data o natočení volantu v průběhu jízdy.

Popis kapitol

Práce se skládá z teoretické a praktické části.

V teoretické části se budeme věnovat snímačům polohy a natočení. Zaměříme se na jednotlivé typy snímačů, jejich vlastnosti, použití a možnosti pro připojení k mikrokontroléru pomocí datových sběrnic. Poté si v systému MATLAB Simulink vytvoříme matematický model systému rekonstrukce úhlu natočení na základě simulovaných vstupů akcelerometrů. Následně do něho implementujeme šum změřený na reálném akcelerometru a zhodnotíme, jestli šum neovlivňuje příliš naše výsledky a tudíž není potřeba změnit metodu nebo použít jiný akcelerometr.

V praktické části se pokusíme zvolenou metodu implementovat do mikrokontroléru pro účel rekonstrukce signálu v reálném čase. Metodu se budeme snažit optimalizovat pro co nejpřesnější rekonstrukci úhlu natočení. Na závěr celé zařízení sestavíme, a porovnáme výsledky měření s vestavěným snímačem testovacího volantu.

1 Snímače polohy a natočení

Pro snímání polohy a natočení se se nejčastěji používají akcelerometry, gyroskopy a magnetometry. Ke snímání úhlu natočení volantu budeme používat hlavně akcelerometry a pro dodatečné zpřesnění i gyroskopy. V této kapitole se zaměříme na principy, vlastnosti a použití těchto dvou snímačů, resp. jejich MEMS variant, které mají oproti klasickým snímačům mnoho výhod.

1.1 MEMS technologie

MEMS (micro electro-mechanical systems) jsou miniaturizované elektromechanické systémy, založené na křemíkových technologiích, nejčastěji na fotolitografii. Skládají se z elektrických a mechanických komponent o velikosti 1 – 1000 µm.

Celý MEMS systém se skládá z mechanických součástí, senzorů, akčních členů a operační elektroniky. Velikost jednotlivých komponent umožňuje integraci celého systému na jeden čip. Díky pokročilosti technologie jejich výroby tak vznikají nejen velice levná, ale i velice přesná zařízení.

Mezi typické MEMS systémy patří mikropřevodovky, mikromotory, mikroturbíny, mikro-optické komponenty a mikrosenzory. [1] [2] [3]

1.2 MEMS akcelerometr

Akcelerometr je zařízení, které měří zrychlení způsobené silou působící na snímač podle druhého Newtonova pohybového zákona $F = ma$. Tento senzor může například měřit gravitační zrychlení, dostředivé zrychlení u rotujícího tělesa nebo zrychlení způsobené nerovnoměrným pohybem.

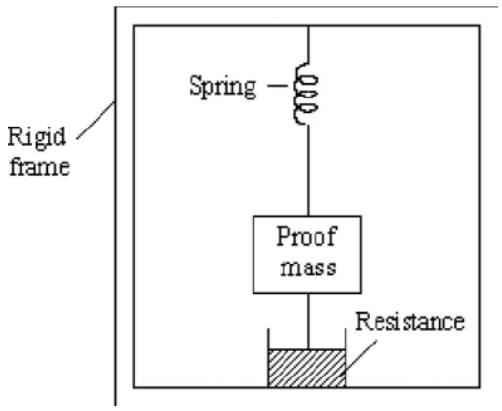
MEMS akcelerometry se hojně používají v automobilovém průmyslu například pro spouštění airbagového systému.[4]

1.2.1 Princip akcelerometru

Obecný princip akcelerometru spočívá v připojení hmotného objektu *(angl. proof mass)* k pružnému členu *(angl. spring)*. Síla působící zrychlení akcelerometru ovlivňuje i výchylku hmotného objektu v čase, ze které můžeme určit měřené zrychlení. Součástí celého systému je i tlumící člen *(angl. resistance)*. [1]

Principiální model akcelerometru můžeme viděl na obrázku 1.1.

Existuje velké množství principů snímání polohy hmotného objektu. My se zaměříme pouze na kapacitní princip, který využívá senzor použitý v této práci.[5]

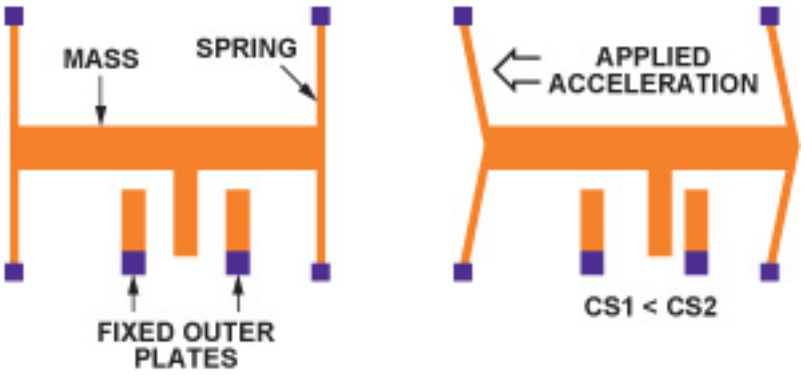


Obr. 1.1: Základní princip akcelerometru. Převzato z [17]

1.2.2 Kapacitní MEMS akcelerometr

Princip kapacitního MEMS akcelerometru je založen na pohyblivé elektrodě, která tvoří hmotný prvek. Ta se vysune vždy proti směru zrychlení. Velikost výchylky závisí na velikosti zrychlení a tuhosti pružného členu.

Kapacita mezi prostřední pohyblivou elektrodou a fixními krajními elektrodami je dána vzorci: $C_{s1} = \frac{\varepsilon S}{d+\Delta d}$ a $C_{s2} = \frac{\varepsilon S}{d-\Delta d}$, kde C_{s1} a C_{s2} jsou kapacity mezi pohyblivou elektrodou a statickými elektrodami podle obrázku 1.2, ε je permitivita prostředí a $d \pm \Delta d$ je vzdálenost pohyblivé elektrody od statických elektrod. Při vychýlení prostřední elektrody z rovnovážné polohy se tedy změní kapacity mezi ní a krajními elektrodami a akcelerometr měří zrychlení. Schéma můžeme vidět na obrázku1.2.



Obr. 1.2: Princip kapacitního akcelerometru. Převzato z [18]

Tato měřící buňka umožňuje snímání zrychlení pouze v jednom směru. Akcelerometr, který používáme se skládá z těchto tří, na sobě nezávislých, na sebe kolmých buněk.[1]

1.3 MEMS gyroskop

Gyroskopy jsou zařízení snímající úhlovou rychlost otáčení a patří mezi nejběžnější navigační zařízení. Používají se například k navádění letadel, lodí a torpéd. Jejich hlavní výhodou je jeho nezávislost na zemské gravitaci, takže se hodí i k navádění vesmírných modulů.[2]

1.3.1 Typy MEMS gyroskopů

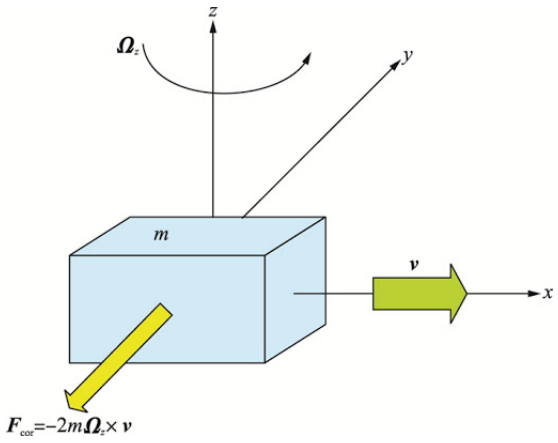
Existují tři typy gyroskopů: mechanický, optický a vibrační. Mechanický gyroskop je založen na rotoru, který zachovává osu své rotace. Tento typ gyroskopu je ale velmi náročný na výrobu, protože ukotvení rotoru musí být téměř bez tření. Proto jsou velmi drahé. Optický gyroskop je založen na Sagnacově jevu a jedná se o nejpřesnější zařízení pro snímání úhlové rychlosti. S tím ovšem souvisí jeho vysoká cena, takže ho používat nebudeme. V této práci budeme pracovat s vibračním gyroskopem, a proto si ho popíšeme blíže.[1]

1.3.2 Vibrační MEMS gyroskop

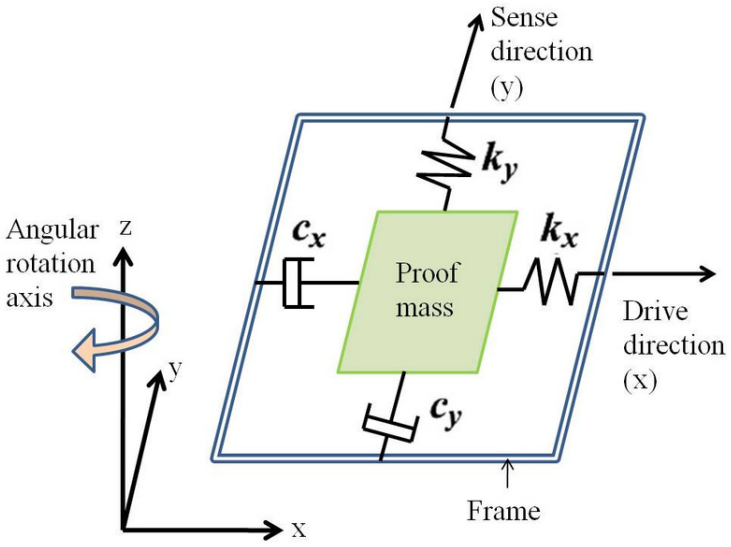
Vibrační gyroskop pracuje na principu Coriolisovy síly, jejíž princip můžeme vidět na obrázku 1.3. Pro Coriolisovu sílu platí, že pokud se hmotné těleso pohybuje podél osy x rychlostí v a rotuje přitom kolem své osy úhlovou rychlostí ω, působí na něj Coriolisova síla vyjádřena vzorcem (1.1), kde F_c je Coriolisova síla, m je hmotnost tělesa, $\boldsymbol{\omega}$ je vektor úhlové rychlosti otáčení soustavy a $\boldsymbol{v}$ je vektor rychlosti tělesa v neinerciální vztažné soustavě.

$$F_c = -2m\boldsymbol{\omega} \times \boldsymbol{v} \tag{1.1}$$

Na obrázku 1.4 můžeme vidět, jak tohoto jevu využívá vibrační gyroskop. Hmotný objekt *(angl. proof mass)* je pomocí statické elektřiny rozkmitán podél osy x. Pokud dojde k rotaci gyroskopu, začne se hmotný objekt vychylovat podél osy y. Tato výchylka je změřena pomocí podobného principu jako u kapacitního akcelerometru, tedy pomocí změny kapacity.[1]



Obr. 1.3: Princip vzniku působení Corilisovy síly. Dostupné na: [19]



Obr. 1.4: Princip vibračního akcelerometru. Dostupné na: [20]

2 Použitý hardware

Pro snímač natočení budeme používat tři zařízení s integrovaným akcelerometrem. Pro první testy budeme používat různé desky a potom se rozhodneme, která nám nejvíce vyhovuje a tu poté objednáme ve více kusech. Pro první testování budeme používat inerciální senzory GY-91 a GY-521. Pro snímání dat budeme používat mikrokontrolér eses klon Arduino UNO R3 CH340. Ten bude data číst, zpracovávat a odesílat po sériové lince do počítače, či jiného datového koncentrátoru. Celé zařízení budeme v rámci bakalářské práce testovat na nepájivém poli.

2.1 GY-91

GY-91 je inerciální snímač s integrovaným senzorem tlaku BMP-280. Akcelerometry a gyroskopy jsou integrovány v čipu MPU-9255. Pro interakci desky do systému slouží celkem 8 pinů. Jejich využití shrnuje tabulka 2.1.

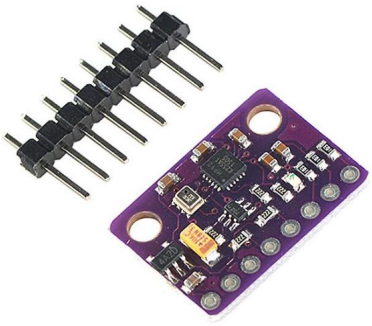
Tab. 2.1: Piny inerciálního senzoru GY-91

Název pinu	Použití
VIN	Napájení 5V DC
3V3	Napájení 3,3V DC
GND	Zemnící vodič pro napájení senzoru
SCL	Sériový clock pro I2C i SPI
SDA	Sériová data pro I2C, MOSI data pro SPI
SD0/SA0	Změna I2C adresy, MISO data pro SPI
NCS	SS signál pro SPI
CSB	Určení aktivity senzoru BMP280

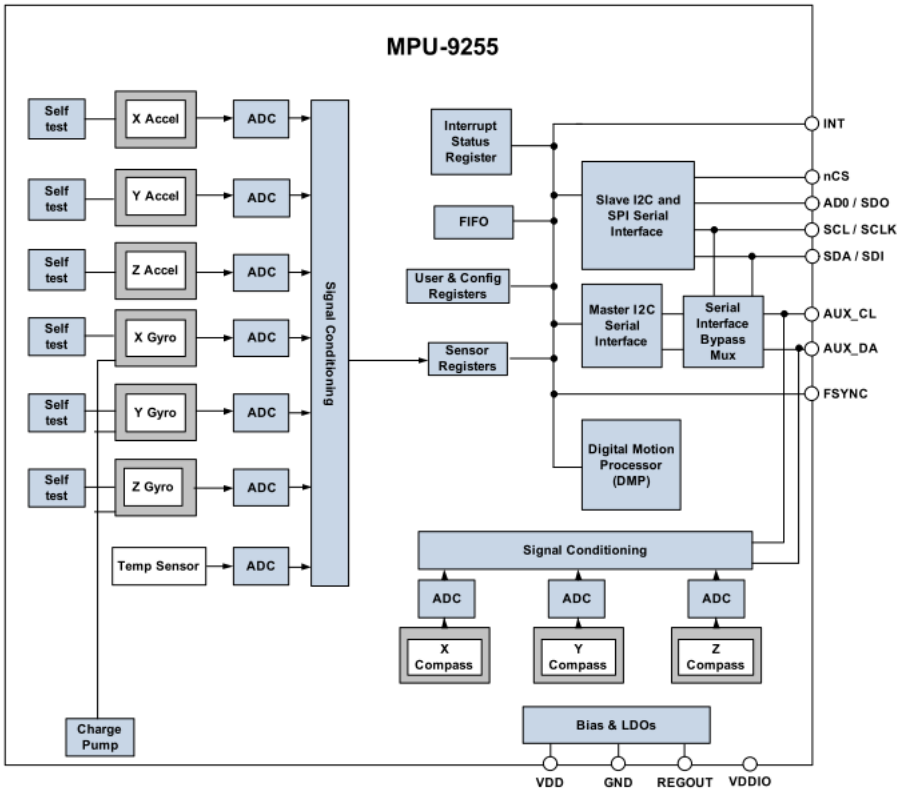
Deska GY-91 je zobrazena na obrázku 2.1.

2.1.1 MPU-9255

Modul MPU-9255 se skládá ze dvou čipů. Čip AK8963 má v sobě integrovaný tří-osý magnetometr. Ten nebudeme v této práci potřebovat a nebudeme se mu tedy proto dále věnovat. Čip MPU-6500 obsahuje tří-osý akcelerometr, tří-osý gyroskop, teplotní senzor a DMP (Digital motion procesor). Na obrázku 2.2 můžete vidět blokový diagram MPU-9255.



Obr. 2.1: Deska GY-91. Dostupné na [7]



Obr. 2.2: Schéma MPU-9255. Dostupné na [5]

Tří-osý kapacitní akcelerometr používá k digitalizaci svého signálu pro každou osu samostatný 16-bitový AD převodník. Měřící rozsah akcelerometru je možné nastavit na ±2g, ±4g, ±8g, nebo ±16g, kde g je konstanta velikosti gravitačního zrychlení. Tabulková hodnota g je cca 9,81 ms^{-2}. Ve skutečnosti se její hodnota liší podle geografické polohy. Pro naše měření by měl stačit výchozí rozsah ±2g, ale pokud by se při testech objevil problém s rozsahem, není problém ho upravit zápisem

do registru 0x1C. Optimální provozní teplota akcelerometru je 25 °C. Pro rozsah od −40 do 85 °C je definován drift nuly 1,5 mg/°C.

Tří-osý vibrační gyroskop také používá k digitalizaci výstupního signálu tři 16-bitové AD převodníky. Měřící rozsah lze podobně jako u akcelerometru nastavit zápisem do registru 0x1B. Měřící rozsahy lze u gyroskopu přepínat mezi ±250 dps, ±500 dps, ±1000 dps, ±2000 dps. Optimální provozní teplota gyroskopu je 25 °C. Pro rozsah od −40 do 85 °C je definována relativní odchylka 4 %.

Modul MPU-9255 lze v rámci desky GY-91 připojit ke sběrnicím I2C a SPI. Pro I2C sběrnici je definována komunikační frekvence maximálně 400 kHz pro fast mode. Pro normal mode je definovaná frekvence 100 kHz. SPI sběrnice má definovanou frekvenci komunikace 1 MHz. Pro I2C sběrnici má modul pevně danou adresu 0x68, která se může změnit na 0x69 při přivedení napětí na pin SA0. Při I2C i SPI komunikaci se modul MPU-9255 chová vždy jako slave.[5]

Modul MPU-9255 obsahuje 126 registrů, které slouží k ukládání posledních naměřených dat, nastavení jednotlivých senzorů nebo umožňují sebetestování (self-test) zařízení. Uvedeme zde jen ty, které považujeme za důležité pro tuto práci.[6]

- Registr 0x1B slouží především k nastavení měřícího rozsahu gyroskopu.
- Registr 0x1C slouží především k nastavení měřícího rozsahu akcelerometru.
- Registr 0x1D umožňuje nastavit vzorkovací frekvenci a šířku pásma (band width) akcelerometru.
- Registry 0x3B až 0x40 slouží pro ukládání naměřených dat akcelerometru. Pro každou osu je vyhrazena paměť 2 byty.
- Registry 0x43 až 0x48 slouží pro ukládání naměřených dat akcelerometru. Stejně jako u akcelerometru je pro každou osu vyhrazena paměť 2 byty.
- Registr 0x68 při nastavení na hodnotu 0x0 resetuje signálové cesty pro akcelerometr, gyroskop a teploměr.
- Registr 0x6C umožňuje vypnout/zapnout jednotlivé senzory.

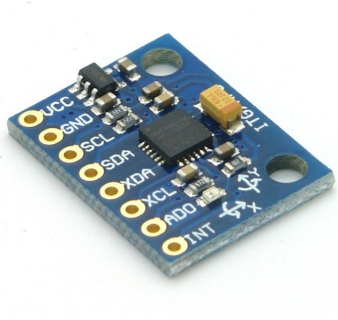
2.2 GY-521

GY-521 na rozdíl od GY-91 obsahuje pouze modul MPU-6050 s akcelerometrem, gyroskopem a teploměrem. Neumožňuje tedy orientaci v prostoru pomocí magnetometru nebo měření atmosférického tlaku. To ovšem pro náš projekt nepředstavuje žádné omezení. Pro interakci desky do systému slouží celkem 8 pinů. Jejich využití shrnuje tabulka 2.2. Desku lze propojit s mikrokontrolérem pouze pomocí I2C sběrnice. SPI komunikaci bohužel nepodporuje.[9]

Deska GY-521 je zobrazena na obrázku 2.3.

Tab. 2.2: Piny inerciálního senzoru GY-521

Název pinu	Použití
VCC	Napájení 5V DC
GND	Zemnící vodič pro napájení senzoru
SCL	Sériový clock pro I2C
SDA	Sériová data pro I2C
XDA	Sériová data pro externí zařízení
XCL	Sériový clock pro externí zařízení
AD0	Změna I2C adresy
INT	Signál dávající informaci o přerušení



Obr. 2.3: Deska GY-521. Dostupné na [9]

2.2.1 MPU-6050

Modul MPU-6050 se ve funkcích, které budeme požívat, úplně shoduje s modulem MPU-9255. Proto zde nebudeme jeho jednotlivé vlastnosti zvlášť uvádět. Modul MPU-6050 se pouze zásadně odlišuje piny XDA, XCL a INT. Pin XDA slouží jako datový pin pro připojení modulu třetích stran. Pin XCL se používá jako zdroj hodinového signálu pro externí modul. Pin INT se používá jako výstup pro vyvolání přerušení. Tento výstup je programovatelný a může reagovat na přetečení FIFO (fronty), na data senzorů připravených v registrech ke čtení a chyby v I2C komunikaci.[10]

2.3 Eses klon Arduino UNO R3 CH340

Eses klon Arduino UNO R3 CH340 je kopie open-sourcového mikrokontroléru Arduino UNO Rev.3. Hlavní výhoda klonů Arduina je jejich nesrovnatelně nižší cena oproti originálu. Originál je cca 5x dražší než kopie. Největší nevýhoda Arduino

klonů je jejich snížená spolehlivost, některé odlišné vlastnosti a téměř neexistující dokumentace.

Eses klon Arduino UNO R3 CH340 je téměř totožný s originálem. Za celou dobu používání se nevyskytla žádná situace, kdy by se tento klon choval jinak než originál. Díky tomu můžeme pro klon používat dokumentaci Arduina UNO, která je na rozdíl od dokumentace klonu na velmi dobré úrovni. Proto kdykoliv budeme psát Arduino UNO, budeme mít na mysli jeho klon.

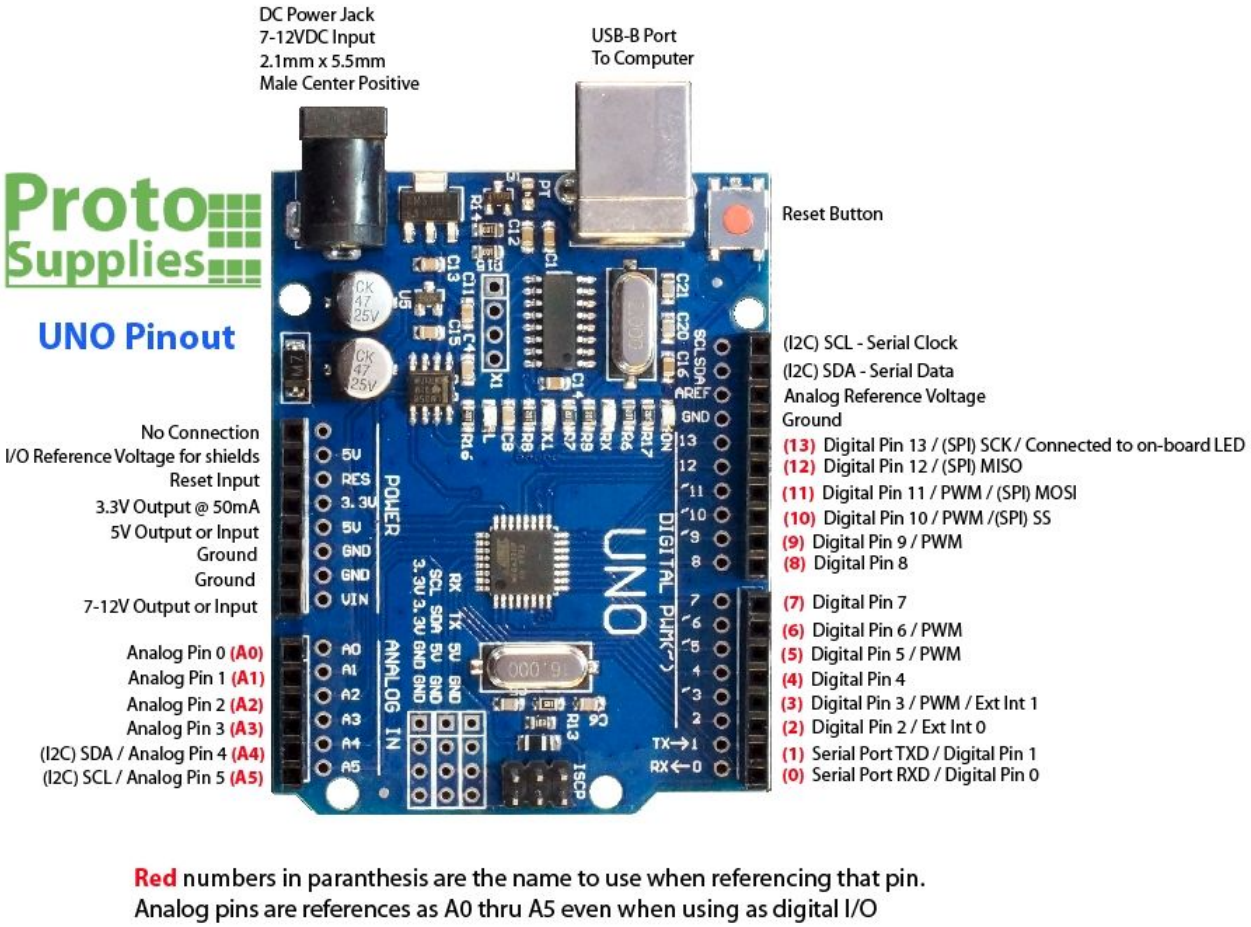
Architektura mikrokontroléru eses klon Arduino UNO R3 CH340 je založena na mikroprocesoru Atmega328P. Jedná se o 8-bitový procesor architektury AVR RISC. Jeho základní takt je 20 MHz.

Součástí čipu Atmega328P jsou vnitřní paměti. Mezi ně patří programovatelná 32-kilobytová Flash paměť. Ta je z bezpečnostních důvodů rozdělena na část pro bootování systému a na část, kam může programátor nahrát svůj program. Bootovací část je oproti části pro aplikace naprosto minimální. Pro tuto paměť je výrobcem garantováno 10 000 přepisovacích cyklů. Pro ukládání dat do I/O registrů slouží 2-kilobytová SRAM. Paměť EEPROM má velikost 1-kilobyte a slouží jako kontrolní a datový registr. U paměti EEPROM je výrobcem garantováno alespoň 100 000 přepisovacích cyklů.[12]

Na desce eses klonu Arduina UNO R3 CH340 se nachází 14 digitálních I/O pinů. Šest z nich má zabudovanou PWM. Na desce se nachází 6 analogových vstupních pinů. Deska dále obsahuje piny pro napájení periférií, které k ní připojujeme. Dále obsahuje připojení na USB typu A, přes které lze zařízení programovat z počítače a vstup pro externí napájení.[11]

Popis jednotlivých částí eses klonu Arduina UNO R3 CH340 můžeme vidět na obrázku 2.4.

Celé Arduino se programuje pomocí jazyka, který je velmi podobný jazyku C a C++. Pro tvorbu programu se často využívají knihovny, které jsou přímo zabudované ve vývojovém prostředí Arduino IDE nebo jsou volně dostupné na internetu.



Obr. 2.4: Arduino UNO Rev. 3. Dostupné na [21]

3 Výběr komunikační sběrnice

Cílem této kapitoly je vybrat nejvhodnější sériovou sběrnici, kterou použijeme pro komunikaci mezi mikrokontrolérem a periferiemi (Desky s integrovanými akcelerometry a gyroskopy). Pro periférie mikrokontroléru Arduino UNO se používají sběrnice I2C a SPI. Mezi těmito dvěma sběrnicemi jsme se rozhodovali. Nakonec jsme se rozhodli pro SPI sběrnici. V této kapitole zdůvodníme proč.

3.1 Vlastnosti datových sběrnic

Než přistoupíme k popisu sběrnic, které budeme používat v tomto projektu, tak se musíme seznámit se základními pojmy, které se datových sběrnic a datové komunikace obecně týkají.

3.1.1 Sériová a paralelní komunikace

Pro vzájemnou komunikaci mezi inerciálními senzory a Arduinem budeme používat sériovou komunikaci. Posílání dat se v tomto případě realizuje jako sekvence bitů, které jsou postupně odesílány po sériové datové sběrnici. Tím se liší od paralelní komunikace, kde jsou skupiny bitů odesílány najednou po paralelních vodičích. Paralelní komunikace je přímá, jednoduchá na implementaci, ale vyžaduje veliké množství vodičů na propojení komponent. Proto budeme v našem projektu využívat sériovou komunikaci, která výrazně šetří počet propojovacích vodičů, vstupních a výstupních vývodů na mikrokontroléru a jeho rychlost pro náš účel naprosto stačí.[15]

3.1.2 Rozdělení zařízení podle práv

Zařízení na datové sběrnici se dělí na dvě kategorie. Master zařízení zpravidla ovládají hodinový signál, zahajují komunikaci a obecně mají na sběrnici vyšší pravomoce než zařízení typu slave. Ty většinou čekají až master zahájí komunikaci, zažádá si o jejich data nebo jim nějaká data pošle. Některé sběrnice umožňují připojení více masterů (tzv. multimaster sběrnice), některé pouze jednoho. [15]

3.1.3 Synchronní a asynchronní sběrnice

Sériové datové sběrnice se dělí na synchronní a asynchronní. Synchronní datové sběrnice se vyznačují hodinovým signálem clock, který je společný všem zařízením a podle kterého jsou synchronizované toky dat mezi nimi. Díky signálu clock je tato

komunikace lépe implementovatelná, ale sběrnice potřebuje o jeden vodič víc. Mezi významné synchronní datové sběrnice patří SPI a I2C.[13]

Asynchronní systémy nemají společný clock, což umožňuje minimalizaci počtu použitých vodičů. Vzhledem k nepřítomnosti hodinového signálu je třeba ověřovat integritu dat posílaných po sběrnici. Pro její ověření musí mít data několik základních vlastností. Musí se posílat předem dohodnutou rychlostí. K tomu se používá tzv. baudrate, který určuje, kolik bitů za sekundu se pošle po asynchronní sběrnici. Baudrate může být téměř jakákoliv hodnota větší než 0. Stačí, když mají vysílač i přijímač nastavenou stejnou hodnotu. Přesto existují standardizované hodnoty: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 75880, 115200 bps a další. Pro ověření integrity dat se dále používají synchronizační bity a paritní bity. Asynchronní komunikaci v našem projektu používáme při posílání dat z mikrokontroléru do počítače.[13]

3.2 I2C sběrnice

Sběrnice I2C je především známá svou jednoduchostí. Používá se, pokud chceme například k mikrokontroléru připojit na krátkou vzdálenost velké množství periferií a naší prioritou není vysoká rychlost, ale spíše snadná implementace. Jedná se o 2-vodičovou multimaster sběrnici pracující v half-duplex režimu. To znamená, že se data mezi master a slave zařízeními vyměňují v obou směrech, ale ne zároveň.[14]

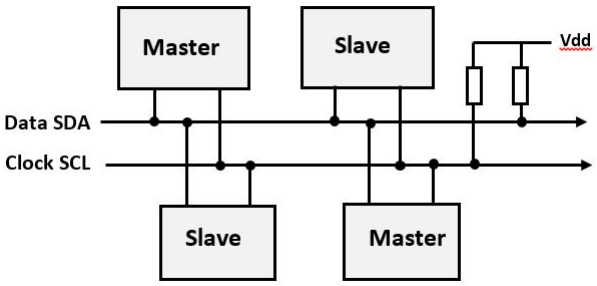
3.2.1 Hardwarové řešení

Vodiče I2C sběrnice se jmenují SDA a SCLK. SCLK vodič umožňuje master zařízení vysílat hodinový signál. Přes SDA vodič se posílají data. Oba vodiče jsou připojeny k tzv. pull-up rezistorům, jejichž účelem je držet oba vodiče na vysoké úrovni napětí. V Arduinu UNO jsou pull-up rezistory integrovány v pinech A4 a A5.

Každé zařízení, které je ke sběrnici připojeno, by mělo být schopné podržet vodiče v nízké úrovni. Proto se jednotlivá zařízení připojují na sběrnici pomocí tzv. openkolektoru. Zařízení se na sběrnici identifikují pomocí adresy, která je (alespoň částečně) pevně daná výrobcem.[14]

3.2.2 Průběh komunikace

Při popisu komunikace budeme používat symbol H pro vysokou úroveň a symbol L pro nízkou úroveň signálu. Zároveň s průběhem komunikace popíšeme i datový rámec.



Obr. 3.1: Topologie I2C sběrnice. Dostupné na [22]

Komunikace začíná, jakmile master vygeneruje start podmínku (S). Start podmínka nastane, pokud je volná sběrnice, SCL je v H a SDA přejde ze stavu H do L. Potom začne vysílat hodinový signál na SCLK a zároveň data na SDA.

První balík dat obsahuje 7-bitovou (platí pro MPU-5255) adresu slave zařízení, které je připojené na sběrnici. Osmý bit říká slave zařízení, jestli chce do něj zapisovat nebo z něho číst. Potom master čeká na odpověď slave zařízení. To potvrdí přijetí datového bloku tzv. acknowledge bitem. To znamená, že oslovené zařízení podrží SDA vodič na jeden takt hodinového signálu v L úrovni.

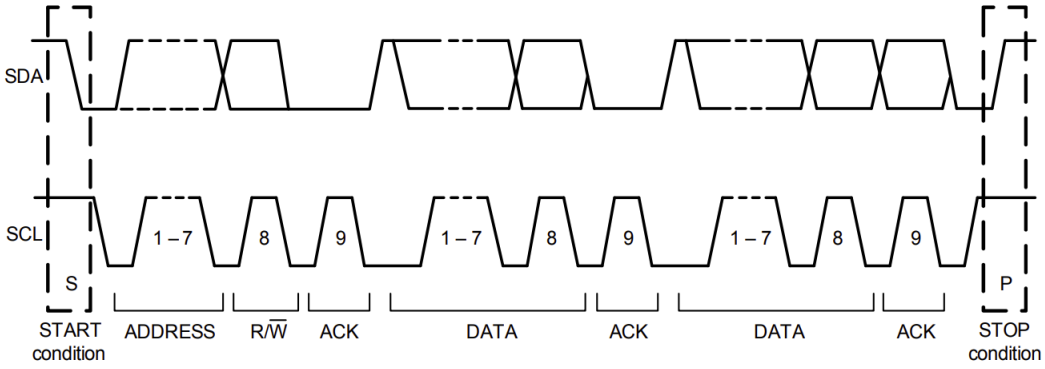
Potom následují typicky dva 8-bitové datové balíky vždy potvrzené acknowledge bitem, které master buďto odesílá nebo přijímá. V případě MPU-5255 obsahuje druhý datový balík adresu vnitřního registru, ze kterého chce master číst nebo do něj zapisovat a až třetí datový balík obsahuje data. Při čtení většího objemu dat než je jeden byte z jednoho zařízení lze využít tzv. burst read sequence a pokračovat v čtení dalšími datovými balíky bez nutnosti opakovat žádanou adresu.

Po ukončení čtení/zápisu může master pokračovat ve využívání sběrnice vygenerováním nové start podmínky, nebo ji může uvolnit. Master uvolní sběrnici vygenerováním stop podmínky (P), která nastane, pokud SCL je v H a SDA přejde z L na H. Potom může sběrnici začít používat jiný master. [5]

Celý průběh komunikace v grafické podobě můžeme vidět na obrázku 3.2.

3.3 SPI sběrnice

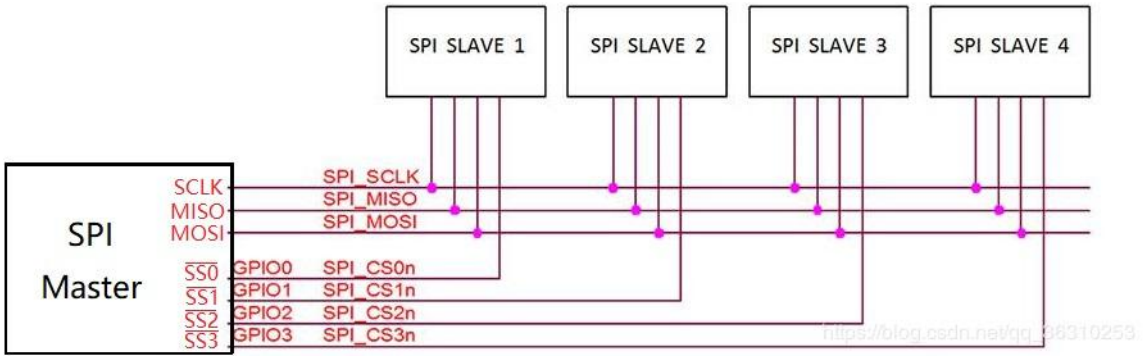
Sběrnice SPI (angl. Serial Peripheral Interface) byla vyvinuta pro embeded systémy na komunikaci mezi procesorem a několika periferiemi na krátkou vzdálenost. Vyznačuje se poměrně snadnou implementací (i když ne tak snadnou jako I2C). Jedná se o 4-vodičovou singlemaster sběrnici pracující ve full duplex režimu, což znamená, že se data mezi master a slave zařízením mohou vyměňovat v obou směrech zároveň.



Obr. 3.2: Komunikace na I2C sběrnici Dostupné na [5]

3.3.1 Hardwarové řešení

Na SPI sběrnici se může připojit pouze 1 master a určité množství slave zařízení. Ty jsou navzájem propojeny čtyřmi vodiči: MOSI, MISO, SCK a SS. MOSI vodič (master out slave in) zajišťuje přenos dat vycházející z mastera do všech slave zařízení. MISO vodič (master in slave out) je vodič, který propojuje všechna slave zařízení s masterem a slouží pro posílání dat z aktivního slave zařízení do mastera. SCK slouží jako zdroj hodinového signálu, který generuje master pro všechna slave zařízení. SS (slave select) je vodič, který rozhoduje o tom, který slave bude aktivní. Každý slave má svůj vodič, který je napojený na mastera a pokud je na SS vysoká úroveň napětí, tak se MISO vstup slave zařízení přepne do stavu vysoké impedance a neinteraguje se sběrnicí. Pokud je SS v nízké úrovni, tak je slave aktivní a komunikuje s masterem. V jedné chvíli může být aktivní pouze jeden slave.[5]



Obr. 3.3: Topologie SPI sběrnice. Dostupné na [23]

3.3.2 Průběh komunikace

Při popisu komunikace budeme budeme používat stejně jako u I2C sběrnice symbol H pro vysokou úroveň napětí a symbol L pro nízkou úroveň. Master zahájí komunikaci výběrem slave zařízení, se kterým chce komunikovat. To udělá změnou úrovně příslušného SS výstupu z H do L.

Potom master pošle data slave zařízení. První bit v prvním bytu obsahuje informaci, jestli chce ze slave číst, nebo do něho zapisovat. Zbylých 7-bitů obsahuje adresu vnitřního registru zařízení. Další byty obsahují data a mohou být odeslány z master zařízení přes MOSI do slave zařízení nebo ze slave zařízení přes MISO do master zařízení.

Celá komunikace probíhá podle taktu hodinového signálu. Např. v MPU-9255 probíhá rychlostí max. 1 MHz a data jsou posílána při sestupné hraně hodinového signálu. Komunikace je ukončena přechodem SS vodiče z L na H. Tím dojde k odpojení slave zařízení od sběrnice.[5]

3.4 Výběr vhodné sběrnice

V této kapitole jsme si krátce představili sběrnice, které podporují námi používané periferie připojené k Arduinu. Bohužel ne všechny periferie podporují obě komunikace. Například deska GY-521 s akcelerometrem a gyroskopem podporuje pouze I2C sběrnici. Výhodou desky GY-91 je podpora I2C a SPI.

Nyní si shrneme všechny výhody a nevýhody obou sběrnic a na závěr se rozhodneme kterou použijeme.

3.4.1 Výhody a nevýhody I2C

Výhody:

- Jednoduchá implementace
- Minimální počet propojovacích kabelů
- Minimální počet potřebných pinů na mikrokontroléru
- Podpora desky GY-91 i desky GY-521
- Arduino je připraveno pro implementaci I2C sběrnice (obsahuje piny s pull-up rezistory)

Nevýhody

- Problém s připojením většího počtu stejných zařízení (námi používané akcelerometry mají přednastavené pouze 2 adresy, kterými se mohou na I2C sběrnici připojit)
- Nižší přenosová rychlost než u SPI

3.4.2 Výhody a nevýhody SPI

Výhody:

- Jednoduchá implementace (I2C je jednodušší)
- Umožňuje propojit veliké množství stejných zařízení (na rozdíl od I2C)
- Vyšší maximální přenosová rychlost než u I2C.

Nevýhody

- Nepodporuje desku GY-521
- Větší množství kabelů, každý slave musí mít svůj SS vodič
- Je dovolen pouze jeden master na sběrnici

3.4.3 Konečné rozhodnutí

Po důkladném zvážení všech výhod i nevýhod obou sběrnic jsme se rozhodli pro použití SPI sběrnice. Hlavní důvod pro naše rozhodnutí byla skutečnost, že námi používané akcelerometry mají pouze dvě I2C adresy, mezi kterými sice lze rychle přepínat pomocí jednoho ze vstupních pinů, ale tím bychom přišli o hlavní výhodu I2C sběrnice, což je maximální jednoduchost zapojení. Byli bychom nuceni ji ovládat podobně jako SPI sběrnici přepínáním slave zařízení mezi aktivním a neaktivním stavem (tedy vyšší a nižší adresou).

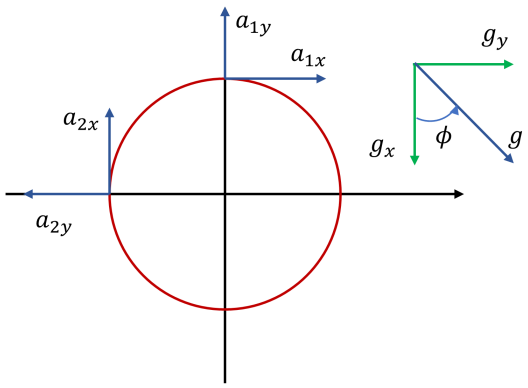
V další kapitole se budeme věnovat matematickému modelu celého zařízení. Celý model budeme následně simulovat v prostředí MATLAB Simulink. Do simulace následně implementujeme i šum, abychom se přesvědčili, zda je algoritmus dostatečně stabilní.

4 Matematický model zařízení

V této kapitole sestrojíme matematický model natáčejícího se volantu, jehož pohyb budou snímat dva akcelerometry. Druhou částí modelu bude zpětná rekonstrukce úhlu natočení. Celý tento systém budeme modelovat v prostředí MATLAB Simulink. Pokud se nám podaří vytvořit funkční model zpětné rekonstrukce úhlu natočení, zavedeme do modelu šum produkovaný reálným akcelerometrem a zjistíme, jestli neovlivňuje příliš zpětný výpočet úhlu natočení. Tento matematický model odvodíme pro jednoduchost nejprve ve 2D prostředí. Následně problém zobecníme na 3D prostor, tedy pro volant natočený v prostoru. Prozatím budeme uvažovat ideální umístění jednotlivých senzorů. Kompenzace nepřesnosti jejich umístění bude podrobně rozebrána v Kapitole 6.

4.1 Odvození výchozích rovnic ve 2D

Na obrázku 4.1 můžeme vidět model volantu, na kterém jsou kolmo na sebe umístěny 2 akcelerometry. Označíme si je jako akcelerometr 1 a akcelerometr 2. Vektory zrychlení, které měří označíme jako $\vec{a_1}$ a $\vec{a_2}$.



Obr. 4.1: 2D model volantu s akcelerometry

Na oba senzory bude působit vektor zrychlení $\vec{g}$, který vzniká vektorovým součtem vektorů $\vec{g_x}$ a $\vec{g_y}$. Tyto vektory vzniknou díky vlivům prostředí jako je gravitační síla, zrychlování vozu, zatáčení apod. Velikost měřeného zrychlení v jednotlivých osách obou akcelerometrů, v závislosti na natočení volantu, způsobenou vektorem $\vec{g}$, popisují rovnice (4.1) a (4.2),

$$\vec{a_{1g}} = (-g\sin(\theta-\phi), -g\cos(\theta-\phi)) \tag{4.1}$$

$$\vec{a_{2g}} = (-g\cos(\theta-\phi), \ g\sin(\theta-\phi)) \tag{4.2}$$

kde g je velikosti zrychlení způsobené vnějším světem, ϕ je úhel jeho natočení (viz. obrázek 4.1) a θ je úhel natočení volantu. Pokud je θ kladná, volant se otáčí v kladném smyslu otáčení.

Při otáčení volantu působí na akcelerometry v jejich osách y (viz. obrázek 4.1) odstředivé zrychlení, jehož příspěvek do celkového měřeného zrychlení lze vyjádřit pomocí vztahů (4.3) a (4.4),

$$\vec{a_{1d}} = (0, \omega^2 R) \tag{4.3}$$

$$\vec{a_{2d}} = (0, \omega^2 R) \tag{4.4}$$

kde ω je velikost úhlové rychlosti a R je vzdálenost akcelerometru od středu rotace volantu.

Při nerovnoměrném otáčení volantu na osy x obou akcelerometrů působí setrvačná síla způsobující zrychlení, které lze vyjádřit pomocí vztahů (4.5) a (4.6).

$$\vec{a_{1a}} = \left(-R\frac{d^2\theta}{d^2t}, 0\right) \tag{4.5}$$

$$\vec{a_{2a}} = \left(-R\frac{d^2\theta}{d^2t}, 0\right) \tag{4.6}$$

Celkové zrychlení $\vec{a_1}$ a $\vec{a_2}$, které budou akcelerometry měřit vyjádříme součtem příspěvků od jednotlivých zdrojů zrychlení.

$$\vec{a_1} = \vec{a_{1g}} + \vec{a_{1d}} + \vec{a_{1a}} \tag{4.7}$$

$$\vec{a_2} = \vec{a_{2g}} + \vec{a_{2d}} + \vec{a_{2a}} \tag{4.8}$$

$$\vec{a_1} = \left(-R\frac{d^2\theta}{d^2t} - g\sin(\theta - \phi), \omega^2 R - g\cos(\theta - \phi)\right) \tag{4.9}$$

$$\vec{a_2} = \left(-R\frac{d^2\theta}{d^2t} - g\cos(\theta - \phi), \omega^2 R + g\sin(\theta - \phi)\right) \tag{4.10}$$

Pro výpočet úhlu natočení volantu nám jsou užitečné pouze příspěvky od vektoru $\vec{g}$. Přitom si můžeme všimnout, že na oba akcelerometry působí příspěvky dostředivých $(\vec{a_{1d}}, \vec{a_{2d}})$ a setrvačných $(\vec{a_{1a}}, \vec{a_{2a}})$ zrychlení stejně. Abychom eliminovali jejich vliv, tak od sebe vektory $\vec{a_1}$ a $\vec{a_2}$ odečteme.

$$\vec{a_1} - \vec{a_2} = (-g\sin(\theta - \phi) + g\cos(\theta - \phi),\ -g\cos(\theta - \phi) - g\sin(\theta - \phi)) \tag{4.11}$$

Pro další postup si rozdělíme rozdíl $\vec{a_1} - \vec{a_2}$ na jeho složky v ose x a y.

$$a_{1x} - a_{2x} = g\left(-\sin(\theta - \phi) + \cos(\theta - \phi)\right) \tag{4.12}$$

$$a_{1y} - a_{2y} = g\left(-\cos(\theta - \phi) - \sin(\theta - \phi)\right) \tag{4.13}$$

Abychom mohli pokračovat ve vyjadřování úhlu θ, tak tyto složky k sobě přičteme a také od sebe odečteme. Dostaneme rovnice (4.14) a (4.15).

$$a_{1x} - a_{2x} + a_{1y} - a_{2y} = -g\,(2\sin(\theta - \phi)) \tag{4.14}$$
$$a_{1x} - a_{2x} - a_{1y} + a_{2y} = g\,(2\cos(\theta - \phi)) \tag{4.15}$$

Z rovnice (4.14) vyjádříme θ

$$\theta = \arcsin\left(\frac{a_{1x} - a_{2x} + a_{1y} - a_{2y}}{-2g}\right) + \phi \tag{4.16}$$

Z rovnice (4.15) vyjádříme θ

$$\theta = \arccos\left(\frac{a_{1x} - a_{2x} - a_{1y} + a_{2y}}{2g}\right) + \phi \tag{4.17}$$

Velikost vektoru $\vec{g}$, včetně jeho úhlu natočení ϕ můžeme vyjádřit pomocí vektorů $\vec{g_x}$ a $\vec{g_y}$ podle obrázku 4.2 pomocí vztahů (4.18) a (4.19).

$$g_x = (g\cos\phi) \tag{4.18}$$
$$g_y = (g\sin\phi) \tag{4.19}$$

Pro velikost vektoru $\vec{g_0}$ platí:

$$g_0 = \sqrt{g_x^2 + g_y^2} \tag{4.20}$$

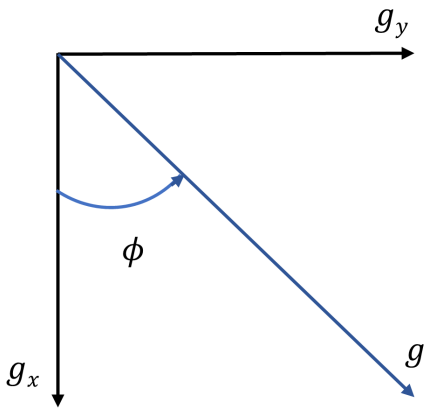
a pro úhel ϕ platí:

$$\frac{g_y}{g_x} = \tan\phi \tag{4.21}$$
$$\phi = \arctan\frac{g_y}{g_x} \tag{4.22}$$

Pomocí rovnic (4.20) a (4.22) můžeme z rovnic (4.16) a (4.17) vyjádřit finální rovnice rekonstrukčního algoritmu.

$$\theta = \arcsin\left(\frac{a_{1x} - a_{2x} + a_{1y} - a_{2y}}{-2\sqrt{g_x^2 + g_y^2}}\right) + \arctan\frac{g_y}{g_x} \tag{4.23}$$

$$\theta = \arccos\left(\frac{a_{1x} - a_{2x} - a_{1y} + a_{2y}}{2\sqrt{g_x^2 + g_y^2}}\right) + \arctan\frac{g_y}{g_x} \tag{4.24}$$



Obr. 4.2: Vyjádření úhlu ϕ pomocí poměru velikostí g_x a g_y

4.2 Simulace 2D modelu v prostředí MATLAB

MATLAB je programovací jazyk a vývojové prostředí vyvíjené společností Mathworks. Jazyk Matlab je založen především na práci s maticemi, ale podporuje i objektové programování. Využití Matlabu je obrovské. Používá se pro zpracování a analýzu dat, hluboké učení, počítačové vidění, zpracování signálů, ovládání robotů, řídící systémy a regulátory, finanční modely a další ...

Simulink je modul MATLABu, který umožňuje simulaci a modelování dynamických systémů. Tyto modely jsou tvořeny v grafickém prostředí pomocí blokových schémat toku signálu. Simulink obsahuje širokou paletu různých bloků jako například násobení konstantou (gain), derivátor, integrátor, různé funkce (např. sinus, cosinus), zdroje signálu (např. zdroj harmonického signálu, zdroj konstantního signálu), scope (pro zobrazení průběhu libovolného signálu v čase) a mnoho jiných. Modely systému v Simulinku jsou počítány pomocí aproximačních numerických metod, jejichž parametry můžeme před simulací nastavit. Výběr správné metody a kroku je velmi důležitý, protože při špatném nastavení nemusí Simulink dávat validní výsledky.

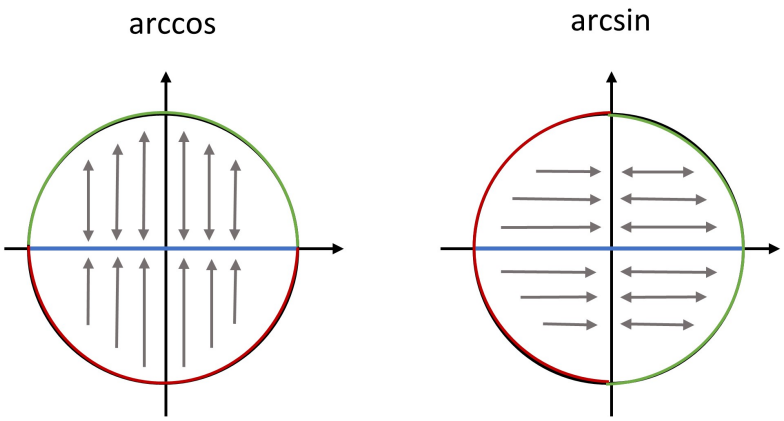
4.2.1 Matematický model rekonstrukčního algoritmu bez šumu

Pro první otestování rekonstrukčního algoritmu budeme předpokládat, že máme ideální akcelerometry, které nemají vlastní šum. Tím se zbavíme velkého množství problémů, které spolu s sebou šum potenciálně přináší. Pokud bude algoritmus funkční, tak se začneme věnovat odstranění problémů způsobených šumem. Celý model je vytvořen v prostředí MATLAB Simulink. Výsledné blokové schéma můžete vidět v

příloze B.1.

Model rekonstrukčního algoritmu se skládá z části, která simuluje natáčení volantu a sběr dat z akcelerometrů, které jsou na volantu umístěny podle obrázku 4.1. Tato část má signál, který symbolizuje natáčení v čase, dále jeho první a druhou derivaci, které symbolizují úhlovou rychlost a zrychlení. Derivace jsme odvodili analyticky a do Simulinku je umístili jako hotové signály, protože derivace jsou obecně u numerických řešení problematické. Pro základní testování jsme použili vstupní signál $y(t) = -\frac{A}{2}\sin(\omega t + \frac{\pi}{2}) + \frac{A}{2}$, kde $A = \frac{5\pi}{180}$ rad, $\omega = \frac{2\pi}{10}$ rad/s, který umožňuje snadnou derivaci.

Druhá část modelu provádí rekonstrukci signálu získaného ze simulovaných akcelerometrů. Ta je sestavena podle vztahů (4.23) a (4.24). To znamená, že rekonstrukci signálu provádí dvě nezávislé větve a máme tedy i dva nezávislé výsledky rekonstrukce. Nevýhodou je, že jedna větev obsahuje funkci arcsin a druhá arccos. Funkce arcsin má obor hodnot $\left\langle -\frac{\pi}{2}, \frac{\pi}{2} \right\rangle$ a funkce arccos má obor hodnot $\langle 0, \pi \rangle$. Pro naši aplikaci ale potřebujeme získat úhel v celém rozsahu, tedy $\langle -\pi, \pi \rangle$. Abychom z oboru hodnot funkcí arcsin a arccos dokázali sestavit celý interval $\langle -\pi, \pi \rangle$, podíváme se na vztahy mezi skutečným úhlem natočení a úhlem, který nám vypočítají funkce arcsin a arccos.



Obr. 4.3: Grafická reprezentace vztahů funkcí cos a arccos, sin a arcsin

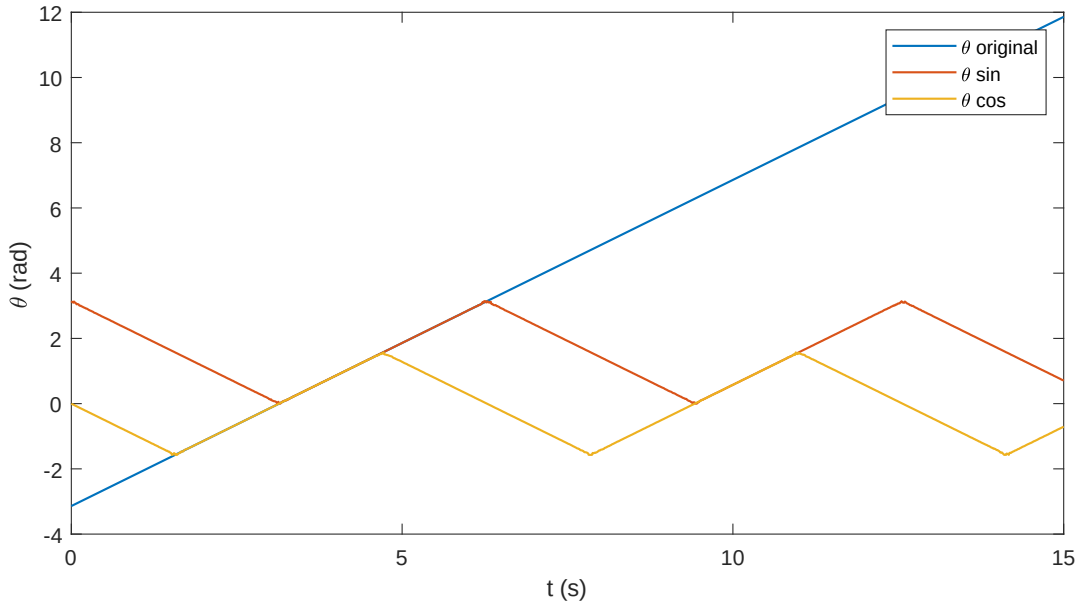
Vztahy mezi úhlem natočení a hodnotami goniometrických funkcí se dají snadno odvodit z jednotkové kružnice. Na obrázku 4.3 můžeme vidět, jak se hodnoty funkcí sinus a cosinus úhlů promítají na modrou osu. Zpátky se pomocí funkcí arcsin a arccos promítají pouze na zeleně zobrazené rozsahy úhlů. Pomocí tohoto obrázku jsme schopni sestavit tabulku 4.1, která systematičtěji popisuje vztahy mezi úhlem natočení a úhlem, který vypočítají funkce arcsin a arccos. Velikost úhlu odpovídá rozdílu θ a ϕ.

Tab. 4.1: Tabulka zobrazení úhlů pomocí funkcí arcsin a arccos

Skutečný úhel	Úhel z arccos	Úhel z arcsin
$\langle 0, \frac{\pi}{2} \rangle$	$\langle 0, \frac{\pi}{2} \rangle$	$\langle 0, \frac{\pi}{2} \rangle$
$\langle \frac{\pi}{2}, \pi \rangle$	$\langle \frac{\pi}{2}, \pi \rangle$	$\langle 0, \frac{\pi}{2} \rangle$
$\langle -\frac{\pi}{2}, 0 \rangle$	$\langle 0, \frac{\pi}{2} \rangle$	$\langle -\frac{\pi}{2}, 0 \rangle$
$\langle -\pi, -\frac{\pi}{2} \rangle$	$\langle \frac{\pi}{2}, \pi \rangle$	$\langle -\frac{\pi}{2}, 0 \rangle$

Náš model zpětné rekonstrukce úhlu natočení obsahuje dvě nezávislé větve výpočtu úhlu natočení, z nichž jedna používá arcsin a druhá arccos. Z tabulky 4.1 potom zjistíme, která větev dává pro který interval validní výsledky a tu prohlásíme za správnou. V intervalu $\langle 0, \frac{\pi}{2} \rangle$ jsou validní obě větve, v intervalu $\langle \frac{\pi}{2}, \pi \rangle$ jsou validní výsledky z větve arccos, pro $\langle -\frac{\pi}{2}, 0 \rangle$ jsou validní výsledky z větve arcsin.

Problém je, že v intervalu $\langle -\pi, -\frac{\pi}{2} \rangle$ nejsou validní výsledky ani z jedné větve. Tento problém jsme vyřešili simulací. Na vstup našeho systému jsme dali lineární signál začínající v čase 0 funkční hodnotou $-\pi$ rad, který bude symbolizovat neustálé rovnoměrné otáčení volantu kolem své osy. Za výstup funkcí arcsin a arccos po přičtení úhlu ϕ jsme dali blok scope a na něm jsme zobrazili charakteristiku uvedenou na obrázku 4.4. Z něho lze vysledovat, že v intervalu $\langle -\pi, \frac{\pi}{2} \rangle$ odpovídá vstupní signál Theta original zápornĕ vzatému signálu z větve arcsin tedy −Theta sin.

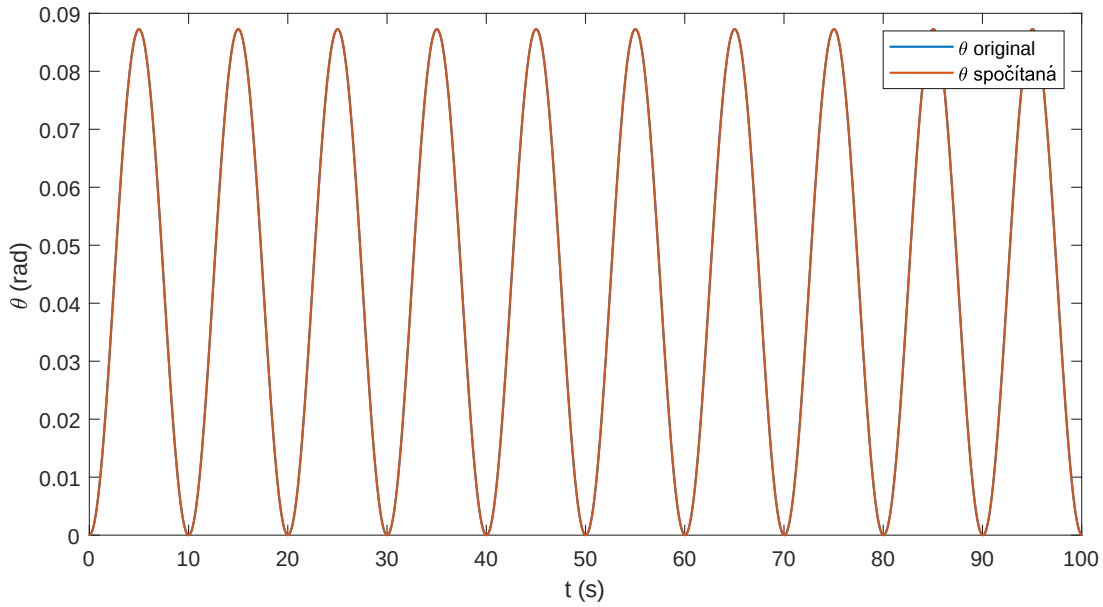


Obr. 4.4: Reakce obou větví rekonstrukčního algoritmu na lineární vstupní signál

Pomocí tabulky 4.1 a znalostí získaných z obrázku 4.4 jsme sestavili rozhodovací

funkci, která vybere na základě intervalů, do kterých vypočítané úhly spadají, validní úhel a ten pošle na výstup celého systému. Ve skutečnosti jsou některé intervaly uvedené v tabulce prodlouženy o velmi malou experimentálně zjištěnou konstantu odpovídající cca 1°, aby nedocházelo k hazardním stavům v rozhodovací funkci a celém algoritmu. Výsledné zapojení je vidět na obrázku B.2.

Výsledky rekonstrukce úhlu pro testovací signál je zobrazen na obrázku 4.5 Z obrázku je vidět, že rekonstrukce signálu z akcelerometru bez šumu funguje správně. Otázkou je, jak moc se změní úspěšnost rekonstrukce, když použijeme reálné akcelerometry, které obsahují šum.

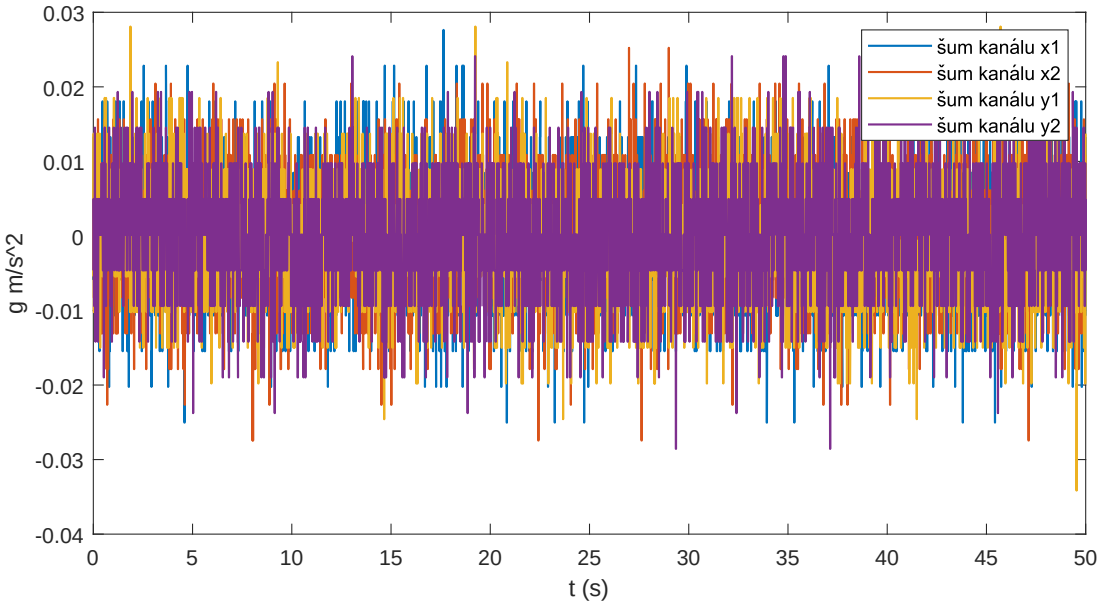


Obr. 4.5: Výsledek kompletního rek. algoritmu pro ideální akcelerometr ve 2D

4.2.2 Matematický model rekonstrukčního algoritmu s šumem

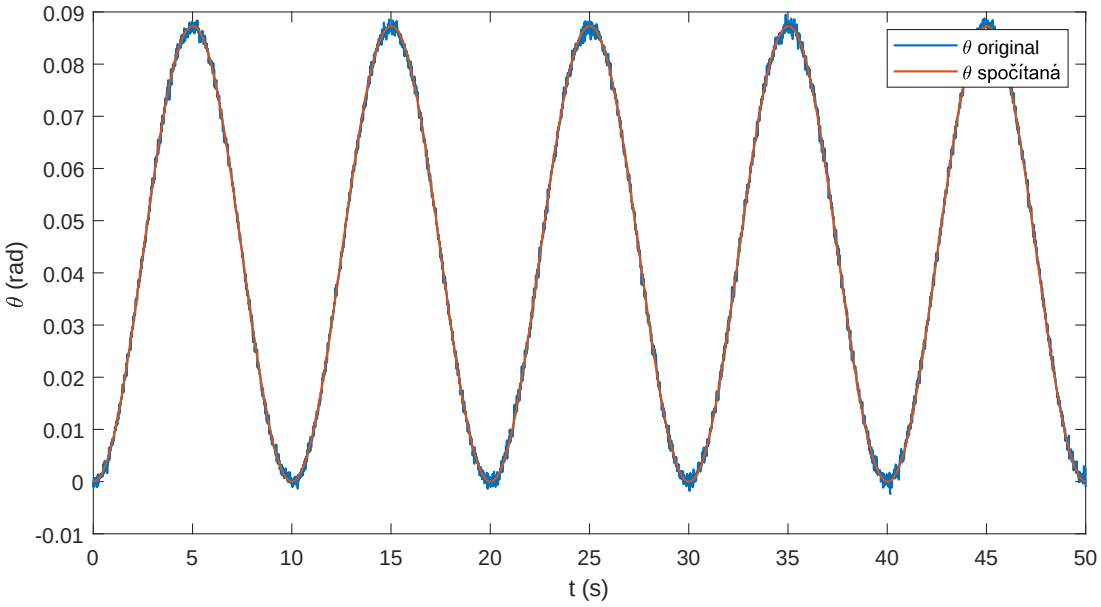
Abychom zjistili, zda je naše metoda použitelná s reálnými akcelerometry, přičítali jsme ke vstupu rekonstrukční části modelu systému šum naměřený na výstupu akcelerometru desky GY-91. Ten jsme získali z tabulky naměřených hodnot na nehybném akcelerometru. Od každého kanálu signálu jsme odečetli jeho průměrnou hodnotu, a tím jsme získali 3 kanály signálu šumu. K simulaci jsme ale použili 4 kanály různých šumových signálů, aby nehrozilo, že se od sebe navzájem odečtou, nebo že spolu budou jakýmkoliv jiným způsobem korelovat. Proto jsme provedli dvě měření šumu reálného akcelerometru. Program, který jsme pro účel sběru dat vytvořili, ukládá data do .csv souboru. Tato data jsme převedli do .mat souboru, který umožňuje jejich přímé načtení do simulace. Detaily zapojení měřeného akcelerometru a sběru dat jsou uvedeny ve čtvrté kapitole.

Na obrázku 4.6 můžeme vidět naměřený šum na výstupu akcelerometru. Šum je dostupný pro 50 sekund simulace. Pokud je simulace delší, tak se začne signál cyklicky od začátku opakovat.

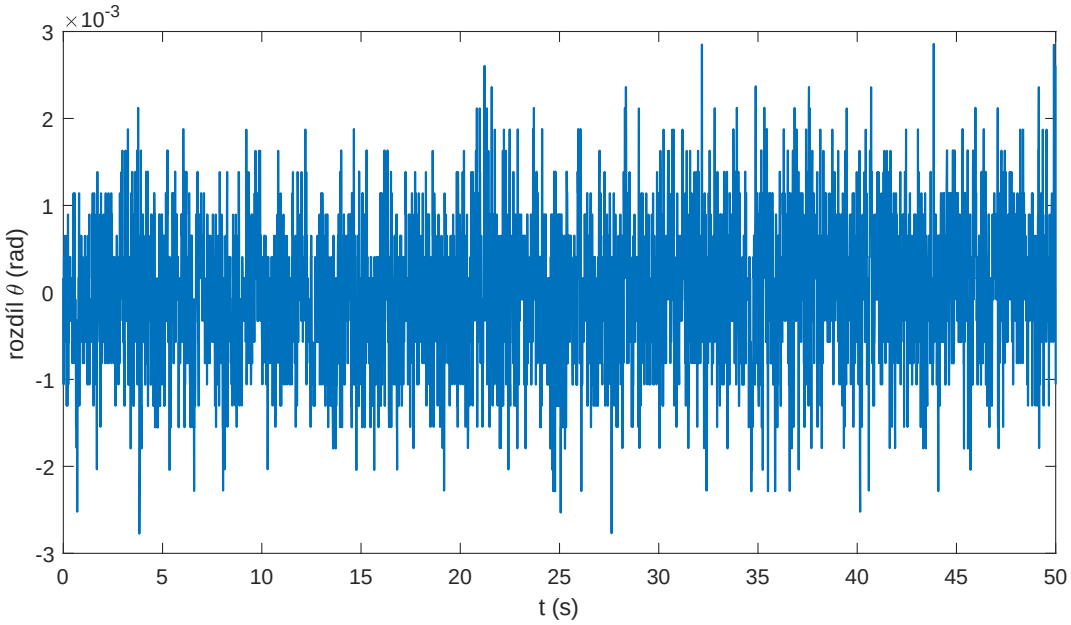


Obr. 4.6: Čistý šum naměřený na nehybných akcelerometrech

Signál šumu jsme implementovali do simulačního schématu. Výsledek rekonstrukce signálu byl nad očekávání dobrý. Vstupní signál a rekonstruované signály můžete vidět na obrázku 4.7 Jejich rozdíl pak můžeme vidět na obrázku 4.8.



Obr. 4.7: Výsledek rekonstrukčního algoritmu pro zašuměný akcelerometr ve 2D



Obr. 4.8: Rozdíl původního a rekonstruovaných úhlů pro zašuměný akc. ve 2D

Tím jsme dokázali schopnost našeho systému rekonstruovat úhel natočení na základě dat z reálného akcelerometru.

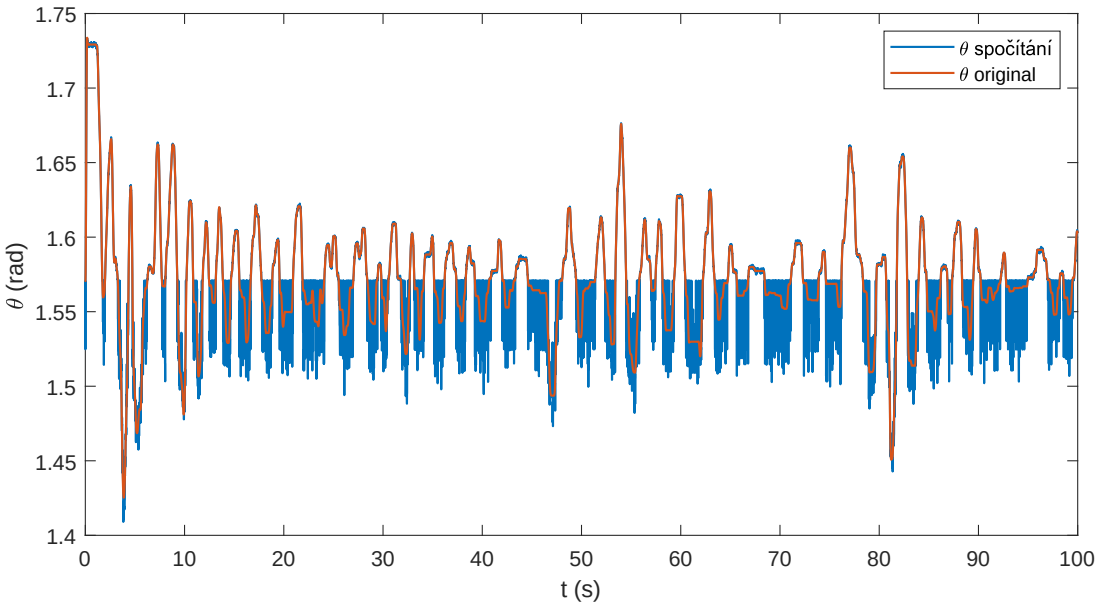
Simulační schéma pro tuto verzi rekonstrukce signálu můžeme najít v příloze B.3.

4.2.3 Vylepšení rekonstrukčního algoritmu

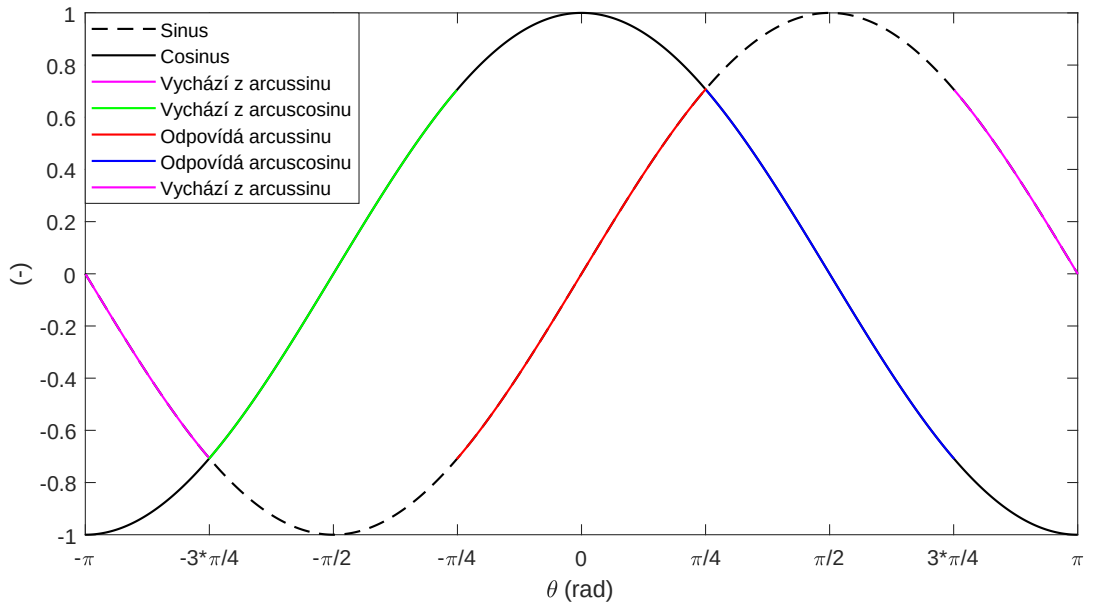
Pro ověření funkčnosti rekonstrukčního algoritmu jsme na jeho vstup dali signál reálného natáčení volantu, který jsme naměřili na simulátoru v laboratořích VUT FEKT. Reálný testovací signál jsme změřili s periodou vzorkování $T_s = 10$ ms. Délka měřeného času jsme stanovili na $t = 100$ s. Znormalizovaná naměřená data jsme zpátky převedli na úhel v radiánech, numericky jsme spočítali jejich první a druhou derivaci a tyto signály jsme přivedli na vstup systému. Výsledek rekonstrukčního algoritmu s šumem můžeme vidět na obrázku 4.9.

Z obou obrázků je vidět, že nám šum naprosto znehodnotil výsledek rekonstrukce. Tato chyba je způsobena vysokou citlivostí funkce arcsin v okolí bodu jedna. Vysoká citlivost funkce arcsin způsobuje, že velmi malá změna hodnoty na vstupu vlivem šumu velmi ovlivní hodnotu úhlu na výstupu. U původního testovacího signálu ke znehodnocení nedošlo, protože se pohyboval v intervalu, který se nenacházel na hraně definičních oborů ani jedné funkce. Proto jsme odhalili tento problém až při testu na reálných datech.

Tento problém vyřešíme rozdělením intervalu $\langle -\pi, \pi \rangle$ na více menších částí. Každou část budeme rekonstruovat větví rekonstrukčního algoritmu, která zde vykazuje menší citlivost. Celý proces rozdělení na menší intervaly popisuje obrázek 4.10.



Obr. 4.9: Výsledek rekonstrukčního algoritmu pro reálný signál



Obr. 4.10: Rozdělení intervalu $\langle -\pi, \pi \rangle$ na úseky méně citlivé na šum

Jak můžete z obrázku 4.10 vidět, rozdělili jsme interval $\langle -\pi, \pi \rangle$ na 5 menších částí. Červeně zvýrazněná část přímo odpovídá úhlu vypočítaném ve větvi s arcsin, modře zvýrazněná přímo odpovídá úhlu na výstupu větve s arccos, zeleně zvýrazněný interval vychází z větve s arccos, ale úhel na výstupu této větve se na správný musí teprve přepočítat, a růžově zvýrazněné intervaly vycházejí z větve s arcsin, ale na správný úhel se taky ještě musí přepočítat.

Abychom získali o celé situaci větší přehled, rozebereme si ji pomocí tabulky 4.2.

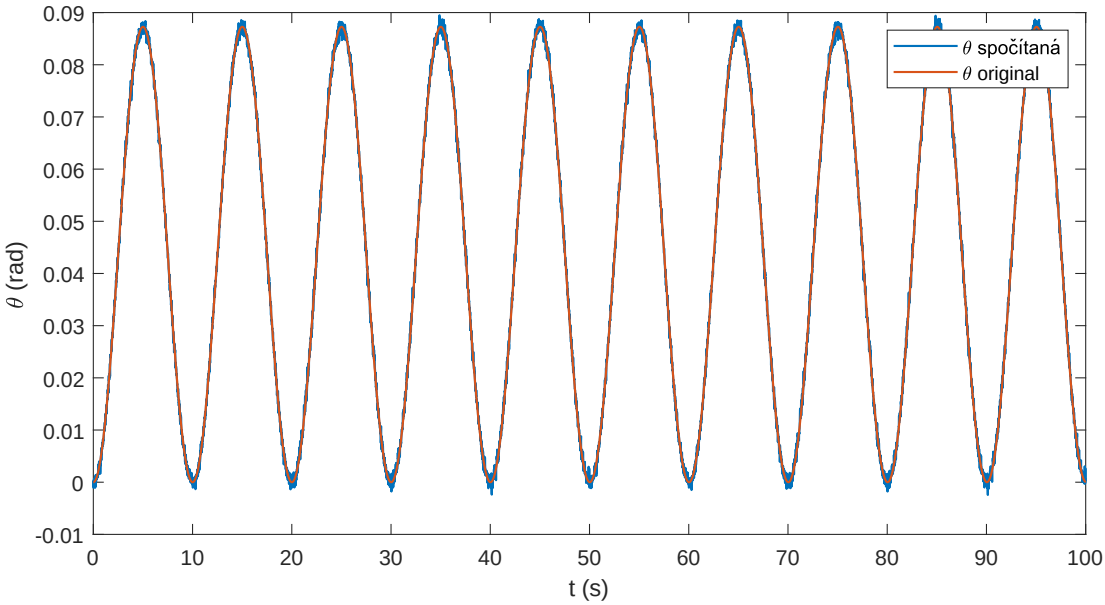
Tab. 4.2: Tabulka zobrazení úhlů pomocí funkcí arcsin a arccos

Skutečný úhel	Úhel z arccos	Úhel z arcsin	Výpočet θ_{vysl}
$\left\langle -\pi, -\frac{3}{4}\pi \right\rangle$	$\left\langle \pi, \frac{3}{4}\pi \right\rangle$	$\langle -\frac{\pi}{4}, 0 \rangle$	$\theta_{vysl} = -\pi - \theta_{sin}$
$\left\langle -\frac{3}{4}\pi, -\frac{\pi}{4} \right\rangle$	$\left\langle \frac{\pi}{4}, \frac{3}{4}\pi \right\rangle$	$\langle -\frac{\pi}{2}, -\frac{\pi}{4} \rangle$	$\theta_{vysl} = -\theta_{cos}$
$\langle -\frac{\pi}{4}, \frac{\pi}{4} \rangle$	$\langle 0, \frac{\pi}{4} \rangle$	$\langle -\frac{\pi}{4}, \frac{\pi}{4} \rangle$	$\theta_{vysl} = \theta_{sin}$
$\left\langle \frac{\pi}{4}, -\frac{3}{4}\pi \right\rangle$	$\left\langle \frac{\pi}{4}, \frac{3}{4}\pi \right\rangle$	$\langle \frac{\pi}{4}, \frac{\pi}{2} \rangle$	$\theta_{vysl} = \theta_{cos}$
$\left\langle \frac{3}{4}\pi, \pi \right\rangle$	$\left\langle \frac{3}{4}\pi, \pi \right\rangle$	$\langle 0, \frac{\pi}{4} \rangle$	$\theta_{vysl} = \pi - \theta_{sin}$

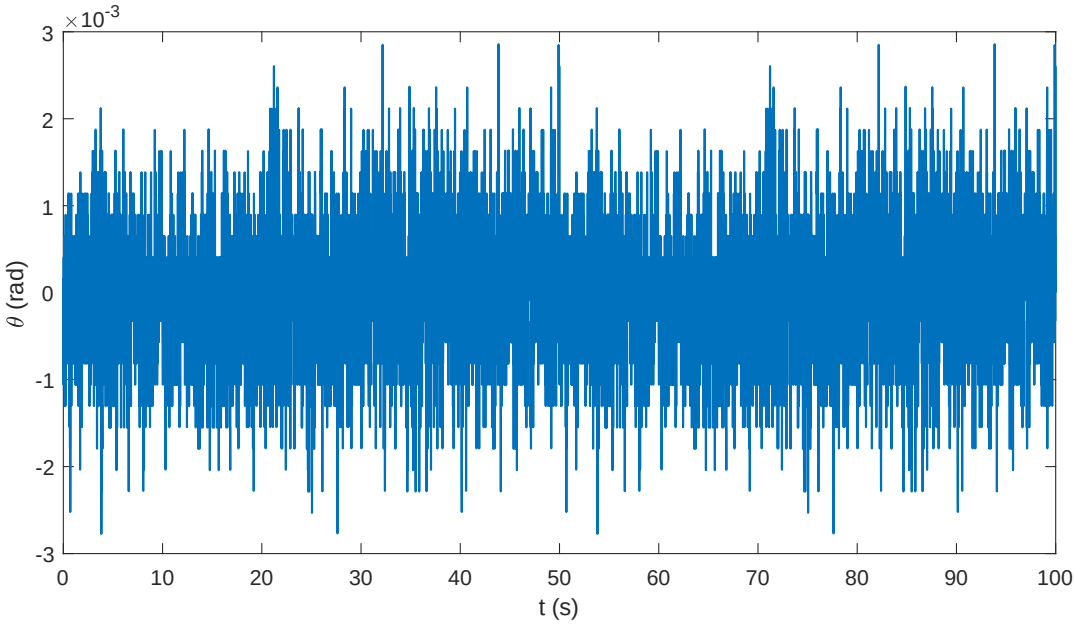
Z tabulky vychází pravidla pro rozhodovací mechanismus. Podle úhlu na výstupu obou větví rekonstrukčního algoritmu zjistíme, který řádek tabulky je pro nás validní a správný úhel vypočítáme pomocí vzorců v pravém sloupci, které byly odvozeny z obrázku 4.10. Ve skutečnosti jsou některé intervaly v tabulce prodlouženy o velmi malou experimentálně zjištěnou konstantu, která zamezuje hazardním stavům v algoritmu.

Celé schéma zapojení vylepšeného rekonstrukčního algoritmu se nachází v příloze B.4 pro ideální verzi, a v příloze B.5 pro zašuměnou verzi. V algoritmu jsme postupně testovali základní testovací signál (viz. obrázky 4.11 a 4.12), lineárně rostoucí signál (viz. obrázky 4.13 a 4.14) a signál z reálného volantu (viz. obrázky 4.15 a 4.16). Všechny testy byly provedeny s šumem, abychom si mohli být jistí, že šum neznehodnotí rekonstrukci.

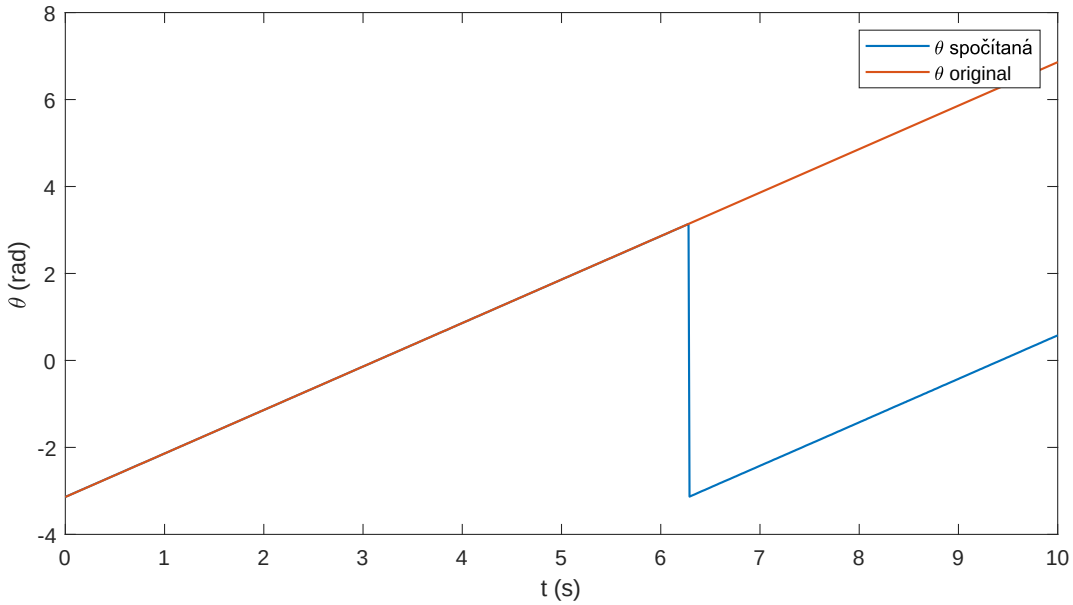
Z obrázků 4.11, 4.13 a 4.15 je vidět, že tento algoritmus pracuje správně i s reálnými akcelerometry.



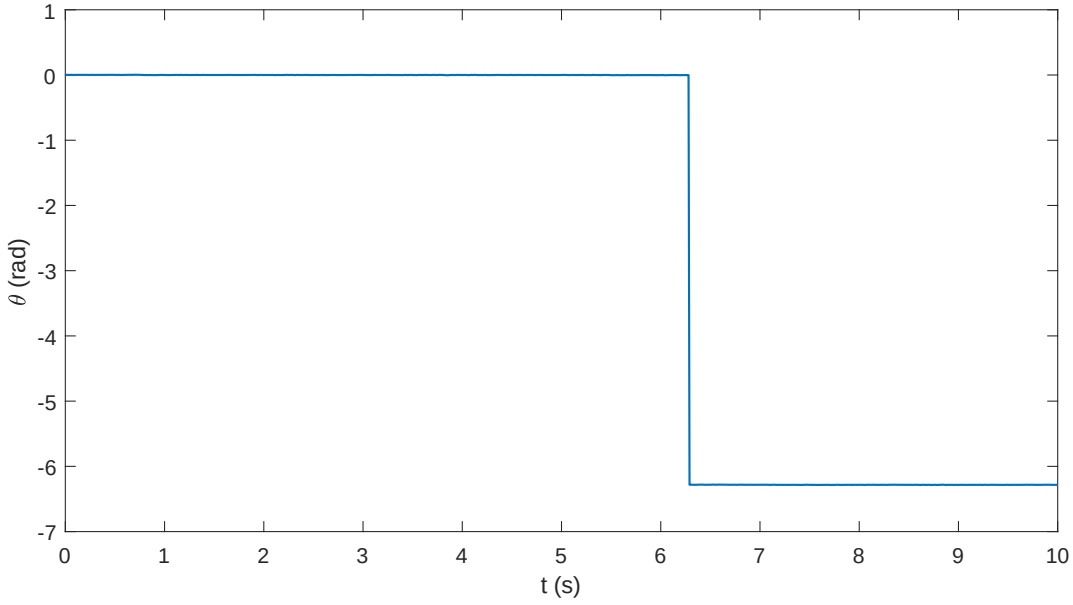
Obr. 4.11: Výsledek rekonstr. alg. pro zašuměný základní testovací signál ve 2D



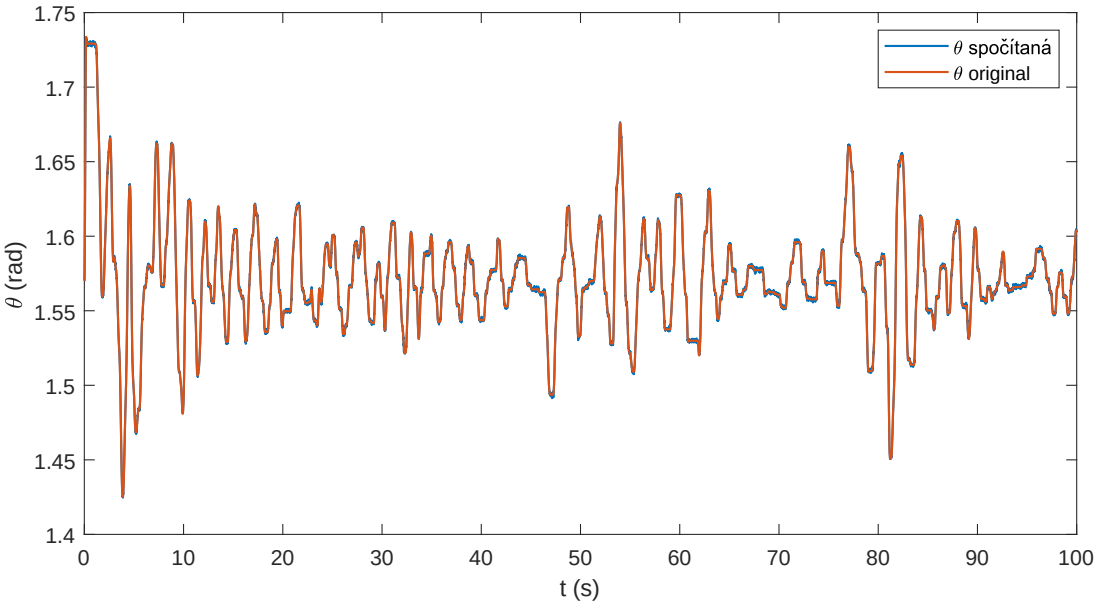
Obr. 4.12: Rozdíl původního a rekonstruovaných úhlů pro případ z obr. 4.11



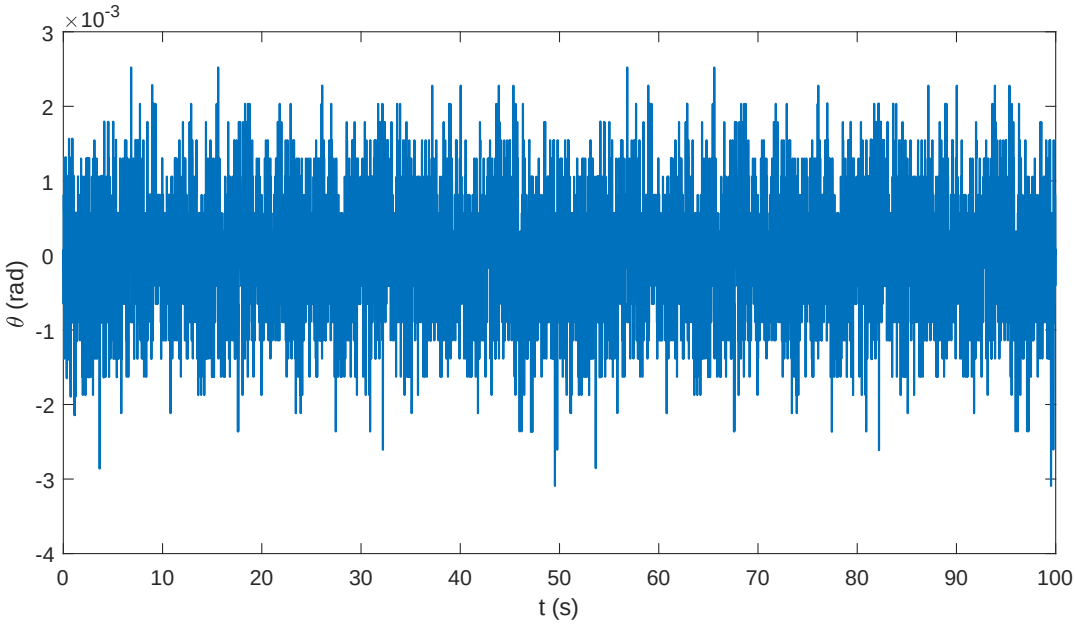
Obr. 4.13: Výsledek rekonstr. alg. pro zašuměný lineární testovací signál ve 2D



Obr. 4.14: Rozdíl původního a rekonstruovaných úhlů pro případ z obr. 4.13



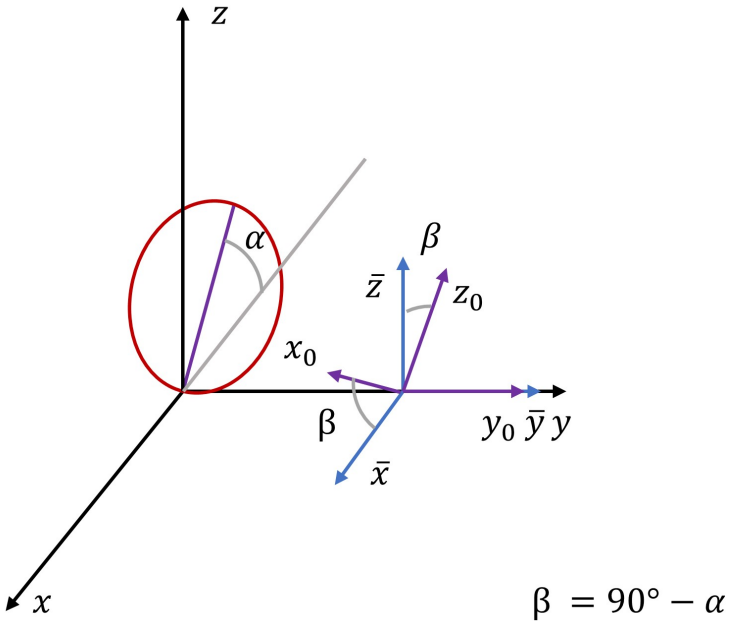
Obr. 4.15: Výsledek rekonstr. alg. pro zašuměný reálný testovací signál ve 2D



Obr. 4.16: Rozdíl původního a rekonstruovaných úhlů pro případ z obr. 4.15

4.3 Transformace do 3D prostoru

V průběhu této kapitoly jsme odvodili kompletní 2D model soustavy pro měření natočení volantu pomocí akcelerometrů. Na závěr této kapitoly tento model zobecníme pomocí transformací na 3D prostor.



Obr. 4.17: 3D model volantu s akcelerometry

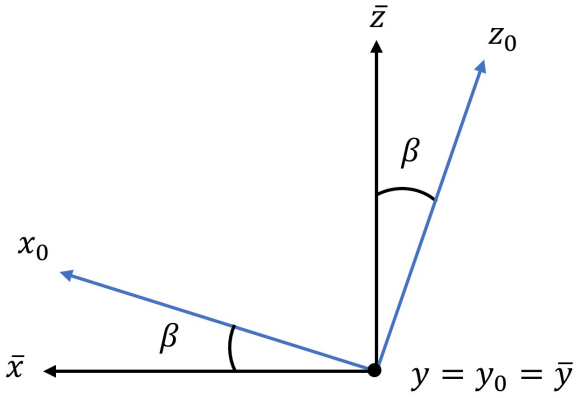
Matematický model pro 2D prostor, který jsme odvodili, odpovídá situaci, kdy je volant kolmý na rovinu silnice, po které se vůz pohybuje. Toto naklonění ovšem není časté a proto jsme se rozhodli umístit volant do prostoru podle obrázku 4.17, kde volant svírá s rovinou xy úhel α. Momentálně budeme řešit případ, kdy je úhel α pevně dán. Proměnnému úhlu α a jeho určení se budeme věnovat v kapitole 6.

Pokud nakloněným volantem proložíme rovinu, tak získáme 2D případ, který jsme řešili výše. Bližším zkoumáním zjistíme, že průmětem souřadných os akcelerometrů do zvolené roviny se osy x_1, x_2, y_1 a y_2 (viz. obrázek 4.1)nezmění a osy z_1 a z_2 se do ní nepromítnou vůbec. Z těchto poznatků můžeme sestavit maticovou rovnici (4.25) a (4.26). Jsem si vědom toho, že nejsme schopni akcelerometry umístit přesně v rovině volantu. Korekcí těchto nepřesností se budeme věnovat v kapitole 6.

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{1-3D} \\ y_{1-3D} \\ z_{1-3D} \end{bmatrix} \tag{4.25}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{2-3D} \\ y_{2-3D} \\ z_{2-3D} \end{bmatrix} \tag{4.26}$$

Pro měření vlivu vnějších sil na volant budeme používat referenční akcelerometr, který bude umístěn mimo rovinu volantu, jak můžeme vidět na obrázku 4.17. Pro měření vlivu vnějších sil na snímače v rovině volantu musíme data z třetího akcelerometru transformovat do 2D prostoru, který je pootočen vůči rovině x_0y_0 třetího akcelerometru o úhel $\beta = \frac{\pi}{2} - \alpha$, kde α je úhel natočení volantu vůči rovině xy . Transformaci popisuje obrázek 4.18.



Obr. 4.18: Rotace referenčního akcelerometru

Z obrázku 4.18 můžeme odvodit, že platí rovnice.

$$\frac{z_0}{z_{a3}} = \cos\beta \tag{4.27}$$
$$\frac{z_0}{x_{a3}} = -\sin\beta \tag{4.28}$$
$$\frac{x_0}{z_{a3}} = \sin\beta \tag{4.29}$$
$$\frac{x_0}{x_{a3}} = \cos\beta \tag{4.30}$$
$$y_{a3} = y_0 \tag{4.31}$$
$$\beta = \frac{\pi}{2} - \alpha \tag{4.32}$$

Z rovnic (4.27) až (4.32) můžeme odvodit transformační rovnici (4.33):

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{a3} \\ y_{a3} \\ z_{a3} \end{bmatrix} \tag{4.33}$$

Pro výpočet vektorů zrychlení $\vec{g_x}$ a $\vec{g_y}$, které působí v osách x_0 a y_0, použijeme rovnici (4.33)

$$\begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{4.34}$$

kde a_x, a_y a a_z jsou velikosti vektorů zrychlení působících v osách x_{a3}, y_{a3} a z_{a3} a g_x, g_y jsou velikosti vektorů zrychlení po transformaci do 2D prostoru, které jsou zobrazeny na obrázku 4.1.

Pomocí rovnic (4.25), (4.26), (4.33) a (4.34) jsme dokázali zobecnit matematický 2D model popsaný rovnicemi (4.23) a (4.24) na 3D prostor, ve kterém je umístěný volant, který je nakloněný vůči rovině xy o úhel α. Tento model také zjednodušuje některé skutečnosti. Mezi jinými například předpokládá ideální umístění snímačů v rovině volantu, předem daný úhel natočení volantu α a přesné umístění třetího referenčního snímače mimo rovinu volantu. Dalšímu řešení těchto nepřesností se bude věnovat kapitola 6.

5 Praktická realizace

V této kapitole se zaměříme na praktickou realizaci celého zařízení. K testování zařízení používáme herní volant Thrustmaster Ferrari Red Legend Edition. Tomu jsme uzpůsobili některé požadavky kladené na zařízení. Například jsme mohli mikrokontrolér Arduino UNO umístit na čelo volantu, což by v normálním automobilu nebylo možné vzhledem k přítomnosti airbagu.

Jak už jsme se zmiňovali v předchozích kapitolách, všechny senzory jsou s mikrokontrolérem spojeny pomocí nepájivého pole, což by pro praktické použití nebylo příliš vhodné, ale pro testování funkčnosti je to naprosto dostačující.

V praxi může být zařízení konstruováno tak, že každý senzor bude vybaven vlastním mikrokontrolérem (např Arduino Nano), které budou posílat data bezdrátově přímo do datového koncentrátoru. Implementace bezdrátového spojení je ovšem nad rámec této bakalářské práce.

5.1 Požadavky na upevnění senzorů

Připevnění senzorů na volant musí splňovat určité požadavky. Mělo by být schopné zůstat na místě a neměnit svoje místo či polohu při otáčení volantu. Dále by nemělo vůbec omezovat řidiče při ovládání vozidla, či blokovat některé bezpečností funkce, nebo ovládací prvky na volantu, jako například airbag, ovládání klaksonu či různých jiných komfortních systémů, jejichž ovládací prvky jsou často ve volantu integrovány.

Místo, kam se umisťují senzory podle obrázku 4.1 je zvoleno vzhledem k standardnímu držení volantu, které bychom mohli popsat pomocí hodinového ciferníku, kdy řidič by měl mít ruce na druhé a desáté hodině. Umístění senzorů by tedy nemělo řidiči v průběhu řízení překážet.

5.2 Upevnění senzorů na volant

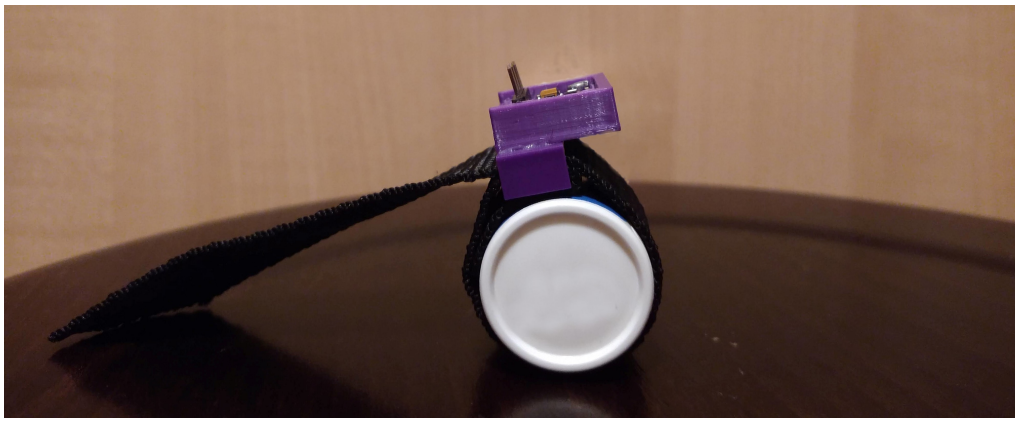
Abychom mohli na volant upevnit inerciální senzory, potřebovali jsme obaly, do kterých bychom je mohli umístit, a které by je alespoň částečně chránily před poškozením. Použili jsme kryt na snímač MPU-6050, který je možné najít na portálu *https://www.thingiverse.com*, a který tvarově odpovídá námi použitému senzoru. Pro každý senzor jsme ho vytiskli na 3D tiskárně. Kryt sice chrání desku před nechtěným poškozením, ale pomocí něho nelze upevnit senzor na volant.

Pro připevnění senzoru v obalu k volantu jsme se inspirovali náramkem u hodinek, u kterého jsou podobné požadavky na stabilitu a nepohyblivost na ruce jako u našeho zařízení. Prvotní návrh držáku senzoru můžete vidět na obrázku 5.1.

Obr. 5.1: První verze úchytu senzoru na volant

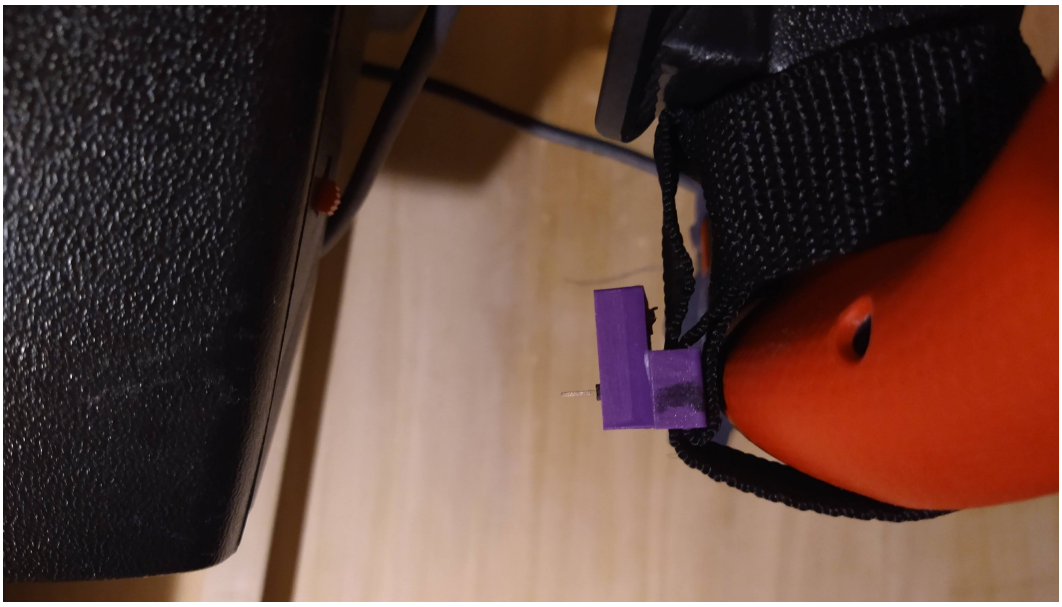
Prvotní návrh náramku se skládá z látkového pásku dostatečně tuhého a hrubého materiálu, na který je připevněna senzorová deska v obalu a na konci pásku je k němu připevněna spona, která je dostatečně úzká, aby jí šla protáhnou pouze jedna vrstva pásku a aby tření pásku bránilo jeho povolení. Pro snadnější prostrčení pásku úzkou mezerou je pásek na konci zkosen.

Finální verze úchytu můžeme vidět na obrázku 5.2. Od první verze se liší pouze tím, že je spona delší, takže na ni lze umístit kryt se senzorem, což je výhodné protože na užším volantu by si mohly tyto dvě plastové části navzájem překážet. Spona je také vytisknuta pomocí 3D tiskárny.



Obr. 5.2: Finální verze úchytu senzoru na volant

Aby fungoval rekonstrukční algoritmus, musí být druhý senzor umístěn kolmo na první. U některých typů volantu to může být problém, protože mohou mít v místě umístění senzoru paprsek kola volantu. Tento problém nastal i u našeho testovacího volantu. Proto jsme systém uchycení lehce modifikovali, jak můžeme vidět na obrázku 5.3. Modifikace se od standardního uchycení liší malou smyčkou za senzorem, která umožňuje uchycení senzoru na paprsek volantu.



Obr. 5.3: Modifikace úchytu pro uchycení v místě paprsku volantu

5.3 Schéma zapojení

Celé zařízení se skládá ze tří inerciálních senzorů, které jsou pomocí SPI sběrnice spojeny s mikrokontrolérem. Kromě SPI sběrnice jde do jednotlivých desek z Arduina také dvouvodičově stejnosměrné napájení 3,3 V DC. K mikrokontroléru je také připojeno tlačítko, které bude použito při přepínání jednotlivých fází kalibrační procedury.

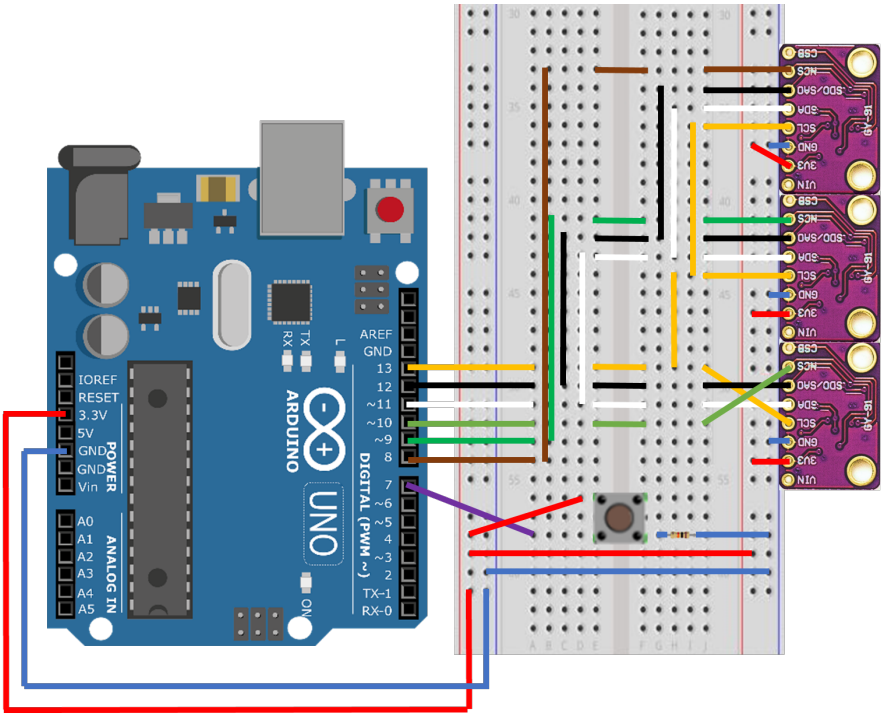
Při propojování akcelerometrů s Arduinem budeme vycházet ze sekce 2.1, kde je podrobně popsáno využití jednotlivých pinů. Softwarová knihovna `SPI.h` zajišťující komunikaci Arduina přes SPI sběrnici definuje piny použité Arduinem. Pro Clock kabel používá pin 13, pro MISO kabel používá pin 12 a pro MOSI kabel požívá pin 11. Vzájemné propojení Arduina a akcelerometrů popisuje tabulka 5.1

Tab. 5.1: Tabulka propojení Arduina a jeho periferií

Název SPI signálu	Pin Arduina	Pin akcelerometru
MISO	12	SD0/SAO
MOSI	11	SDA
SCK	13	SCL
SS	libovolný digitální pin	SCN

Všechna zařízení vzájemně propojíme na nepájivém poli, kde budeme zařízení

testovat. Celé schéma zapojení můžeme vidět na obrázku 5.4.



Obr. 5.4: Schéma zapojení měřícího zařízení

5.4 Snímání dat ze sériové linky a export do MATLABu

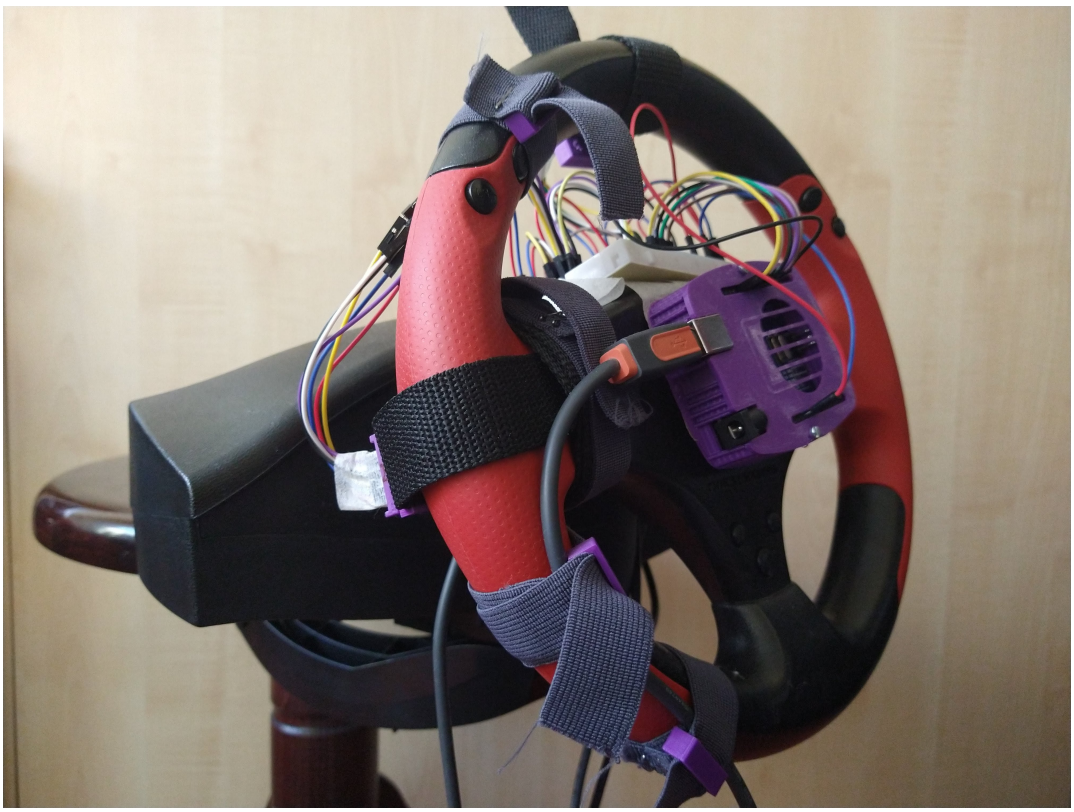
Pro snímání a ukládání dat ze sériové linky do textového souboru vytvoříme krátký skript v Pythonu. Ke čtení dat ze sériové linky použijeme knihovnu `Serial`. Pomocí příkazu `ser = serial.Serial('COM4', 500000)` jsme vytvořili objekt, který slouží k přístupu na sériovou linku s adresou COM4 vysílající rychlostí 500000 bps. Poté ve for smyčce pomocí příkazu `data=str(ser.readline())`, přečteme řádek poslaný po sériové lince. Další řádky for cyklu slouží k zformátování řádku a jeho uložení do .csv souboru.

Kompletní zdrojový kód skriptu je uveden na přiloženém CD.

V MATLABu si do workspace načteme .cvs soubor. Data jednotlivých kanálů uložíme do samostatných .mat souborů, protože s těmi se v MATLABu snadněji pracuje, než s .csv soubory. S takto získanými daty se dá přímo pracovat v simulaci.

5.5 Realizace

Z důvodu nemožnosti testovat zařízení na simulátoru v laboratořích VUT FEKT, jsme zařízení testovali v domácích podmínkách na herním volantu od společnosti Thurstmaster. Výslednou realizaci můžeme vidět na obrázku 5.5.



Obr. 5.5: Fotografie prototypu výsledného zařízení

6 Vylepšení rekonstrukčního algoritmu

Při odvozování rovnic rekonstrukčního algoritmu v kapitole 4 jsme často zanedbali některé skutečnosti, abychom měli snadnější odvozování. Například jsme neuvažovali natočení senzorů vůči ose volantu, nebo jsme předpokládali, že známe úhel naklopení volantu. Abychom ale dokázali úhel natočení volantu zrekonstruovat co nejpřesněji, musíme tyto výchylky a nepřesnosti identifikovat a navrhnout vhodnou kompenzaci.

6.1 Kalibrace akcelerometrů

Při ověřování funkčnosti akcelerometrů jsme zjistili, že akcelerometry mají v každé ose velmi znatelný offset. Proto abychom byli schopni pomocí akcelerometrů provádět rozumně přesná měření, musíme je nejdříve zkalibrovat. Každý akcelerometr jsme před kalibrací uložili do obalu z 3D tiskárny. Poté jsme je natáčeli, aby měly jednotlivé osy ve směru a proti směru gravitačního zrychlení a zaznamenali jsme průměrnou hodnotu na výstupu AD převodníku. Tento proces jsme provedli pro všechny akcelerometry. Výsledkem je tabulka 6.1.

Tab. 6.1: Tabulka naměřených hodnot z AD převodníků akcelerometrů

Číslo akcelerometru	Osa X	Osa Y	Osa Z
Akcelerometr 1 pro zrychlení g	16700	16100	16968
Akcelerometr 1 pro zrychlení -g	-16100	-16600	-16100
Akcelerometr 2 pro zrychlení g	12800	19200	-18900
Akcelerometr 2 pro zrychlení -g	-20000	-13400	14100
Akcelerometr 3 pro zrychlení g	19100	11800	12490
Akcelerometr 3 pro zrychlení -g	-13700	-21000	-20600

Pokud by byl akcelerometr správně nakalibrován už z výroby, bylo by na výstupu AD převodníku, při zrychlení o velikosti $1g$ na rozsahu $\pm 2g$, číslo 16384. Abychom akcelerometry zkalibrovali, sestavili jsme rovnici (6.1)

$$\pm 16384 \quad = \quad kx + q \tag{6.1}$$

kde k a q jsou hledané parametry kalibrace a x je hodnota naměřená akcelerometrem. Pomocí této rovnice jsme sestavili kalibrační tabulku 6.2. Parametr k vyšel pro všechny akcelerometry 1. Proto je v tabulce pouze parametr q.

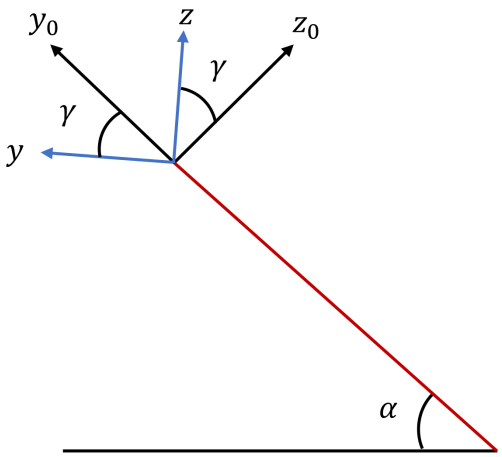
Hodnoty z tabulky 6.2 stačí přičíst k výsledné hodnotě na výstupu AD převodníku jednotlivých kanálů akcelerometrů.

Tab. 6.2: Velikost parametru q pro kalibraci akcelerometrů

Číslo akcelerometru	Osa X	Osa Y	Osa Z
Akcelerometr 1	-300	251	-430
Akcelerometr 2	3596	-2915	-2383
Akcelerometr 3	-2697	4595	4016

6.2 Natočení senzoru vůči ose volantu

Pro umístění senzoru na volant jsme navrhli v kapitole 5 systém, který zaručuje, že se senzor nemůže natáčet podél osy y. Tím jsme velmi zjednodušili kalibrační proces a matematickou náročnost celého problému. Rotaci kolem osy z je způsobena posunem náramku po obvodu volantu. Tuto rotaci dostatečně vykompenzujeme správným umístěním senzoru na volant. Při jejich umístění budeme snímat data osy x prvního akcelerometru (umístěn v horní části volantu) a osy y druhého akcelerometru. Při umístění senzorů na volant se budeme snažit, aby obě hodnoty byly minimální. Pokud bude pásek náramku vyroben z hrubého neklouzavého materiálu a bude v prvním kroku kalibrace správně nastaven, máme minimalizovanou rotaci kolem osy y a z inerciálního senzoru. Bohužel rotaci kolem osy x inerciálních senzorů nelze rozumně kompenzovat při usazení měřícího pásku, takže ji bude třeba kompenzovat matematicky. Natočení senzoru podle osy x o úhel γ popisuje obrázek 6.1, kde červenou barvou je znázorněna osa volantu, která je naklopena o úhel α vůči rovině podložky. Černé barvou jsou znázorněné osy ideálně umístěného snímače a modrou barvou jsou znázorněny osy snímače, který je pootoče o náhodný úhel γ.



Obr. 6.1: Schéma natočení senzoru na volantu

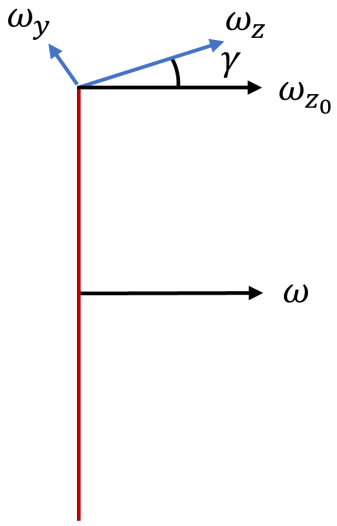
Z obrázku 6.1 můžeme odvodit rovnice (6.2) a (6.3), které popisují transformaci nakloněného snímače do ideální polohy.

$$z_0 = z\cos(\gamma) - y\sin(\gamma) \tag{6.2}$$

$$y_0 = y\cos(\gamma) + z\sin(\gamma) \tag{6.3}$$

Abychom mohli výše uvedené transformační rovnice použít, musíme zjistit, o jaký úhel jsou snímače natočeny. K tomu jsme vytvořili druhý krok kalibrační procedury, kde s výhodou použijeme gyroskopy.

Při plynulém otáčení volantu by při ideálním natočení snímače docházelo k jeho rotaci pouze kolem osy z. Pokud je snímač natočen o úhel γ, otáčí se částečně podle osy z i y. Celá situace je zobrazena na obrázku 6.2. Červená čára znázorňuje pohled na volant z boku. Vektor $\vec{\omega}$ znázorňuje vektor úhlové rychlosti otáčejícího se volantu a vektor $\omega_{z0}^{\rightarrow}$ znázorňuje úhlovou rychlost naměřenou ideálně umístěným snímačem. Modré šipky znázorňují vložky vektoru $\omega_{z0}^{\rightarrow}$, které v jednotlivých osách naměří víceosý gyroskop pootočený o úhel γ.



Obr. 6.2: Schéma pro zjištění natočení senzoru

Z obrázku 6.2lze odvodit rovnici (6.4), ze které lze zjisti úhel natočení volantu. Tento úhel lze zjistit, pouze pokud dochází k rotaci volantu kolem své osy.

$$\gamma = \arctan\left(\frac{-\omega_y}{\omega_z}\right) \tag{6.4}$$

Výhodou rovnice (6.4) je její nezávislost na úhlové rychlosti volantu. Proto jsme vytvořili jednoduchou kalibrační proceduru, kdy uživatel několikrát zakmitá volantem a na základě snímané úhlové rychlosti zjistíme úhel natočení snímače. Pro filtraci

dat a získání korektnějšího výsledku použijeme metodu PCA (Principal component analysis), která najde ve stavovém prostoru všech naměřených dat přímku prokládající všechny body měření metodou nejmenších čtverců. Směrnice přímky je dána jako hlavní vlastní vektor matice A, která vznikne autokorelací naměřených dat. Pro nalezení hlavního vlastního vektoru matice A použijeme iterativní metodu, která se velmi snadno implementuje.[27]

Náklon snímačů na volantu jsme vyřešili pomocí druhého kroku kalibrační procedury. Náklon referenčního snímače není třeba řešit, protože třetí akcelerometr bude součástí výsledného datového koncentrátoru, který není součástí této práce. Pro laboratorní testování tohoto zařízení není nutné zjišťovat náklon referenčního snímače, protože jsme schopni v laboratorních podmínkách zajistit jeho správnou polohu.

6.3 Určení náklonu volantu

Abychom byli schopni měřit natočení volantu s rozumnou přesností, musíme transformovat měřené hodnoty z referenčního akcelerometru do volantu, jak jsme si ukázali v kapitole 4. Abychom to mohli udělat, musíme především určit náklon volantu vůči rovině silnice. K zjištění tohoto parametru bude sloužit třetí část kalibrační procedury, kdy bude auto v klidu a volant ve výchozí poloze (tzn. bude mít nulové natočení). Pokud zajistíme tyto podmínky, tak můžeme zjistit natočení volantu z vektoru gravitačního zrychlení který bude působit na oba akcelerometry umístěné na volantu.

Působení vektoru gravitačního zrychlení na první akcelerometr umístěný na vrchu volantu popisuje obrázek 6.3.

Z obrázku 6.3 lze vyjádřit rovnici (6.5), pomocí které jsme schopni zjistit úhel natočení volantu.

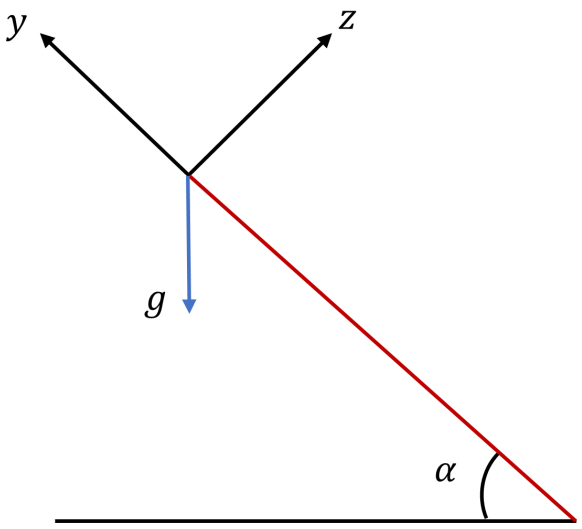
$$\alpha = \arctan\left(\frac{z}{y}\right) \tag{6.5}$$

kde z je velikost zrychlení působící v ose z, y je velikost zrychlení působící v ose y a α je úhel natočení volantu vůči podložce podle obrázku 4.17.

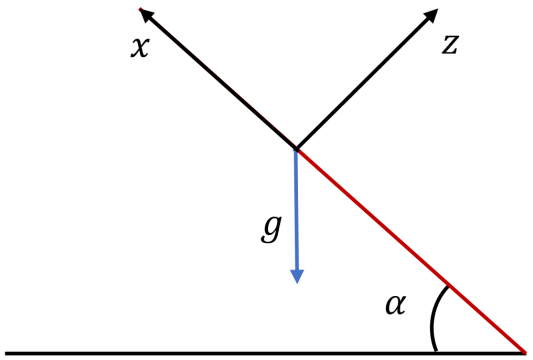
Podobným způsobem můžeme získat úhel náklonu volantu z druhého akcelerometru, který je umístěn na boku volantu. Působení vektoru gravitačního zrychlení na druhý akcelerometr popisuje obrázek 6.4.

Z obrázku 6.4 lze podobně jako v případě prvního akcelerometru vyjádřit rovnici (**??**), pomocí které jsme schopni zjistit úhel natočení volantu.

$$\alpha = \arctan\left(\frac{x}{z}\right) \tag{6.6}$$



Obr. 6.3: Schéma pro zjištění úhlu naklonění volantu z prvního senzoru



Obr. 6.4: Schéma pro zjištění úhlu naklonění volantu z druhého senzoru

kde x je velikost zrychlení působící v ose x, z je velikost zrychlení působící v ose z a α je úhel natočení volantu vůči podložce podle obrázku 4.17.

Díky výše odvozeným rovnicím jsme schopni v kalibrační proceduře určit úhel náklonu volantu a díky němu také spočítat transformace zrychlení působících na referenční akcelerometr.

6.4 Detekce několikanásobného otočení volantu

Při odvození základního algoritmu v kapitole 4 jsme jej odvodili pro základní rozsah $\langle -\pi, \pi \rangle$. Ve většině skutečných automobilů lze ovšem volantem otáčet na jednu stranu vícekrát. Aby rekonstrukce úhlu natočení byla úplná, měli bychom reflektovat i tuto vlastnost automobilového volantu.

Z toho důvodu jsme zavedli stavovou proměnou, která reflektuje překročení původně definovaného rozsahu algoritmu $\langle -\pi, \pi \rangle$. Pomocí tabulky 4.2 jsme tento interval rozdělili na pět menších intervalů. Abychom byli schopni určit přechod původního intervalu, algoritmus si pamatuje ve kterém intervalu se nacházel minule. Pokud došlo k přechodu z intervalu $\left\langle \frac{3}{4}\pi, \pi \right\rangle$ do intervalu $\left\langle -\pi, -\frac{3}{4}\pi \right\rangle$ přičteme ke stavové proměnné jedničku. Pokud by přechod probíhal opačným směrem, jedničku odečteme. Vzorec pro výpočet výsledného úhlu je popsán rovnicí (6.7)

$$\theta_{fin} \quad = \quad \theta + 2\pi k \tag{6.7}$$

kde k je stavová proměnná, θ je úhel natočení získaný ze základního algoritmu a θ_{fin} je výsledný úhel natočení.

6.5 Využití gyroskopů a komplementárního filtru

Úhel natočení volantu by mělo být v ideálním případě možné snímat i pomocí gyroskopů. Respektive v případě ideálního natočení senzorů pouze integrálem jejich gyroskopu snímajícího rotaci kolem osy z. Bohužel tuto metodu nelze použít, protože gyroskopy mají velké stejnosměrné rušení, které by v případě neustálé integrace jejich výstupního signálu vedlo k nestabilitě celého algoritmu. Naproti tomu akcelerometr se potýká s vysokofrekvenčním rušením, které do měření také vnáší nepřesnosti.

Pokud bychom výsledky měření pomocí akcelerometrů a gyroskopů vhodně zkombinovali, dostaneme přesnější výsledek. Pro tuto aplikaci můžeme použít tzv. komplementární filtr, který navrhujeme ve frekvenční oblasti, a který tvoří dolní propust pro akcelerometry a horní propust pro gyroskopy tak, aby se jejich data vzájemně ve frekvenční rovině doplňovala. V našem případě budeme počítat s diskrétní variantou komplementárního filtru, která jde vyjádřit rovnicí (6.9)

$$\theta_{gyro}(k) \quad = \quad \theta_{akc}(k-1) + \Delta\theta_{gyro-z}(k) \tag{6.8}$$

$$\theta(k) \quad = \quad \omega\ \theta_{gyro}(k) + (1-\omega)\theta_{akc}(k) \tag{6.9}$$

kde ω je volitelný parametr, který volíme v intervalu $(0, 1)$, θ_{gyro} je úhel natočení získaný díky gyroskopu a θ_{akc} je úhel natočení získaný díky akcelerometrům a

$\theta_{gyro-z}(k)$ vyjadřuje otočení volantu snímané pomocí gyroskopu od posledního čtení senzorů. Hodnotu ω jsme volili podle online článků 0,96. [24] [25]

7 Program pro výpočet úhlu natočení

V této kapitole se budeme věnovat programu, který umožňuje čtení dat akcelerometrů a gyroskopů a na základě nich určí aktuální úhel natočení volantu. Program používá metody knihovny `MPU_9250_SPI`, která implementuje základní funkce pro práci s SPI sběrnicí. Program při čtení dat pracuje s frontou, která je implementována přímo v rámci čipu MPU-9250. Mimo jiné tento program obsahuje kalibrační proceduru, aby mohl být přístroj použit na jakýkoliv volant. V průběhu této kapitoly si jednotlivé části programu podrobně probereme.

7.1 Knihovna MPU_9250_SPI

Program, pro periodické vyčítání dat z akcelerometrů a gyroskopů používá knihovnu `SPI_MPU_9250.h`, kterou jsme pro tento účel vytvořili úpravou volně šiřitelné knihovny `MPU9250.h` od autora Hideaki Tai ze serveru *github.com*. Tato knihovna je poskytována pod licencí MIT, která umožňuje bezplatné použití i libovolnou úpravu zdrojového kódu. Knihovna `SPI_MPU_9250.h` využívá standardní knihovnu `SPI.h`, která implementuje základní funkce umožňující Arduinu komunikovat pomocí SPI sběrnice. Knihovnu `SPI_MPU_9250.h` jsme se snažili vytvořit co nejjednodušeji a proto obsahuje pouze metody pro základní inicializaci, čtení z registrů, zápis do registrů a drobné servisní funkce. Všechny metody patří do třídy `SPIMPU9250`, jejíž objekty reprezentují jednotlivé inerciální senzory s čipem MPU-9250.

7.2 Inicializace senzorů

Inicializace senzorů probíhá v bloku `void setup()` hned po definici proměnných a startu sériové komunikace s PC. Metoda `SPIMPU9250::begin()` provede prvotní inicializaci, která nastaví žádané rozsahy senzorů, resetuje celou komunikaci a ověří, jestli je senzor schopný komunikovat přes SPI sběrnici. V případě chyby uloží do proměnné status zápornou hodnotu a zbytek programu už se dál nevykonává.

Dále se sérií zápisů hodnot do registrů senzoru nastavíme frekvenční filtr senzorů, abychom se zbavili šumu. Pro dolnofrekvenční propust implementovanou přímo v inerciálních senzorech jsme zvolili mezní frekvenci 43 Hz. Poté nastavíme vzorkovací frekvenci pro frontu na 100 Hz a na závěr nastavíme, které své vnitřní senzory má inerciální snímač zapisovat do své fronty. Tuto inicializaci provedeme pro všechny tři inerciální senzory a poté zápisem do jejich `USER_CTRL` registrů nastartujeme ukládání hodnot z vnitřních senzorů do fronty. Tento zápis nebude pro všechny tři inerciální senzory nikdy úplně synchronní, protože se vykoná jako sekvence příkazů, ale měli

bychom se snažit nastartovat ukládání dat do front jednotlivých senzorů v co nejtěsnějším sledu, aby se minimalizovala chyba způsobená časovým posunem. Časový posun je nutné minimalizovat, protože chceme, aby se co nejlépe odečetly rušivé složky odstředivého a setrvačného zrychlení působící na senzory na volantu.

7.3 Kalibrační procedura

Kalibrační procedura je velmi důležitá pro celý výpočetní algoritmus, protože díky ní jsme schopni odhalit špatné umístění senzorů, korigovat jejich špatné natočení a také zjistit úhel naklopení volantu. Všechny tyto skutečnosti je nutné korigovat. Kalibrační procedura se nachází v bloku `void setup()`, který probíhá pouze jednou. Při kalibraci ale musíme neustále cyklicky číst a zpracovávat data ze senzorů. Proto je celá procedura umístěna uvnitř smyčky `while()`, která skončí ve chvíli, kdy proběhne poslední krok kalibrační procedury. Celá kalibrační procedura je řešena jako stavový automat.

Přestože v této části programu využíváme pouze část dat z inerciálních senzorů, musíme je číst všechny, protože ke všem datům z fronty se přistupuje pomocí jednoho registru a není možné si je vybírat.

7.3.1 Nastavení pozice senzorů

V první části kalibrační procedury upevňujeme senzory na volant. Pro zajištění správné pozice senzorů podle obrázku 4.1 vykreslujeme přes sériovou linku na obrazovku počítače data z AD převodníku osy x prvního snímače umístěného na vrcholu volantu a osy y druhého snímače umístěného na boku volantu. Posouváním senzorů po volantu se snažíme docílit, aby byly obě hodnoty minimální. Jakmile jsme s umístěním volantu spokojení, zmáčkneme tlačítko a přesuneme se do druhé části procedury.

7.3.2 Zjištění úhlu natočení senzorů pomocí gyroskopů

Zjištění úhlu natočení senzorů probíhá ve dvou krocích. V prvním kroku zaznamenáváme data a provádíme jejich autokorelaci. Pokud bychom tuto úlohu řešili v MATLABu, stačilo by vektorově vynásobit matici naměřených dat s její transponovanou obdobou. Toto provést nemůžeme, protože Arduino nemá dostatek paměti. Proto tento výpočet provedeme iterativně, kdy vytvoříme matici z aktuálních dat na vstupu a výsledek přičteme k výsledné matici. Tu na závěr podělíme počtem vstupních dat. Výsledkem je matice $\mathbf{A}$ o rozměrech 3×3. Tento postup vyjadřuje maticová rovnice (7.1). Její iterativní variantu představuje rovnice (7.4).

$$\mathbf{A} = \mathbf{d}^T \times \mathbf{d} \tag{7.1}$$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \tag{7.2}$$

$$a_{i,j} = \frac{1}{K} \sum_{n=1}^{\infty} d_i(k) d_j(k) \tag{7.3}$$

$$i, j = 1, 2, 3, \ldots \tag{7.4}$$

V druhém kroku hledáme hlavní vlastní vektor matice **A**. Výpočet probíhá iterativně podle algoritmu Power method. Výsledkem je hlavní vlastní vektor matice udávající směr osy volantu, který dosadíme do rovnice (6.5) a vypočteme výsledný úhel naklonění senzorů.

7.3.3 Určení úhlu náklonu volantu

V třetí části kalibračního procesu nás program vyzve, abychom uvedli volant do výchozí polohy a zmáčkli tlačítko. Jakmile to uděláme, program na základě průměru dat z obou akcelerometrů umístěných na volantu vypočte úhel natočení volantu podle vzorce (6.4). Důležité je, aby se během této fáze neotáčel volant ani se nepohybovalo auto. Algoritmus by také vyhodnotil chybně úhel, pokud by vozidlo stálo v prudkém kopci.

Po ukončení měření, které trvá 10 sekund, bude uživatel vyzván ke zmáčknutí tlačítka. Po jeho stisknutí začne program měřit úhel natočení volantu.

7.4 Čtení dat a výpočet úhlu z akcelerometru

Čtení dat a výpočet úhlu z akcelerometru probíhá v sekci `void loop()`. Jedná se o smyčku, která se provádí neustále dokola. Většina dat, která čteme z akcelerometrů, jsou šestnáctibitová čísla, která se nachází ve dvou osmibitových registrech nebo za sebou ve frontě. Jednotlivé registry vyčítáme pomocí
metody `SPIMPU9250::readRegister(uint8_t adresa)`. Pro získání šestnáctibitové hodnoty registru musíme provést kód `Var=VarH<<8|VarL;`, kde bity z H registru bitově posuneme o osm bitů doleva a spojíme s bity z L registru. Výsledek uložíme do proměnné typu unsigned int. Tuto hodnotu znormalizujeme podle zvoleného rozsahu, na kterém měříme.

Tímto způsobem na začátku zjistíme, jestli se ve frontě nachází data, která bychom mohli číst. Tzn. každá fronta má v sobě všechna data ode všech senzorů. Pokud je v každé frontě alespoň dvanáct bytů, začneme je postupně číst. Všechna data uložená ve frontě se nachází na jedné adrese, ze které je postupně čteme a skládáme do jednoho šestnáctibitového čísla podle postupu uvedeného výše.

Data následně znormalizujeme do inženýrských jednotek, transformujeme podle natočení jednotlivých snímačů, které jsme naměřili v kalibrační proceduře. Dále data použijeme v obou větvích rekonstrukčního algoritmu. Další část je rozhodovací mechanizmus, který vybírá správnou větev rekonstrukčního algoritmu pro výsledný úhel. Algoritmus se rozhoduje podle tabulky 4.2. V závěru je implementována funkce pro detekci kroucení volantu umožňující snímání úhlu natočení v libovolně velkém rozsahu. Na závěr je výsledek zpřesněn pomocí dat z gyroskopů a komplementárního filtru. Výsledný úhel pošleme pomocí funkce `Serial.print();` po sériové lince do počítače, kde ho můžeme ukládat, či dále zpracovávat.

Kompletní zdrojový kód programu je uveden na přiloženém CD.

8 Ověření funkčnosti zařízení

Cílem této práce bylo výsledné zařízení otestovat v laboratorních podmínkách na simulátoru vozidla VUT FEKT a porovnat data naměřená simulátorem s daty naměřenými pomocí mého zařízení. Bohužel vzhledem k současné situaci, kdy je přístup žáků i doktorandů do prostor VUT FEKT značně omezen, jsme se rozhodli test zařízení provést na herním volantu Ferrari Racing Wheel Red Legend Edition, který byl k dispozici i v domácích podmínkách.

8.1 Vyčítání dat ze senzoru ve volantu

Herní volant se po připojení do počítače chová jako jakýkoliv herní ovladač. Odesílá tedy do počítače data otočení v několika osách, z nichž využívá pouze jednu. Tato data zpravidla reprezentuje reálné číslo v rozsahu $\langle -1, 1 \rangle$. Dále volant posílá informace o zmáčknutí jednotlivých tlačítek, reprezentované boolovskou hodnotou.

Abychom byli tato data schopni číst, použili jsme knihovnu `Pygame`, která se používá pro tvorbu jednoduchých počítačových her v jazyku Python. Z této knihovny použijeme modul `pygame.joystick`, který slouží pro čtení dat z herních ovladačů. Pro čtení dat z herního volantu jsme použili ukázkový kód dostupný na stránkách s dokumentací [26], který jsme upravili, aby byl schopný tato data převést na stupně a uložit je do .csv souboru.

8.2 Porovnání dat z testovaného senzoru volantu

Abychom nemuseli psát vícevláknovou aplikaci, která by zároveň pracovala s daty z volantu a testovaného senzoru, rozhodli jsme se použít dva samostatné programy a jejich výsledky synchronizovat v MATLABu při závěrečném porovnání. Tím vyřešíme také problém s rozdílnou vzorkovací frekvencí, kdy testovaný senzor má vzorkovací periodu 10 ms, kdežto vnitřní snímač volantu má vzorkovací periodu 50 ms.

Pro čtení dat z testovaného senzoru jsme použili program popsaný v sekci 5.4, který periodicky čte data ze sériové linky a ukládá je do .csv souboru. Pro čtení dat z volantu jsme použili program popsaný výše.

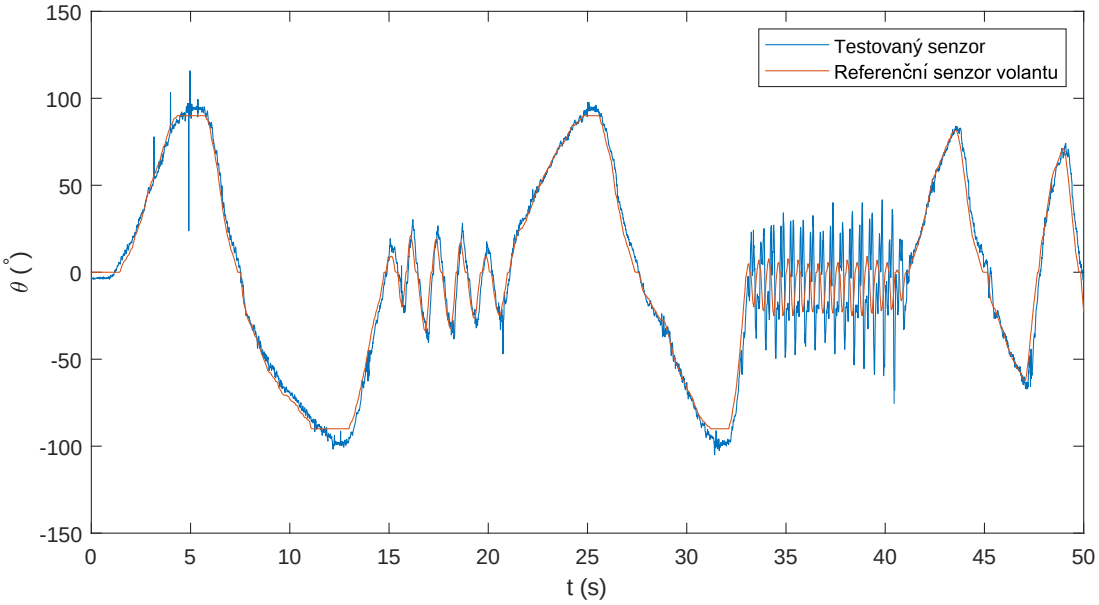
Tento test bude pouze orientační, vzhledem k poměrně velké mrtvé oblasti, kde snímač volantu hlásí nulové natočení a skutečnosti, že snímač volantu se saturuje na $\pm 90°$ natočení, i když samotný volant se může ještě dále otáčet.

8.3 Výsledky porovnání

Výsledný graf 8.1 porovnávající hodnoty měřené snímačem umístěným na volantu a referenčním snímačem uvnitř volantu ukazuje, že zařízení funguje velmi dobře. Vzhledem k výše popsaným problémům referenčního snímače nemůžeme určit jeho skutečnou přesnost, ale jedná se o důkaz použitelnosti této metody pro měření úhlu natočení volantu.

Zařízení má občas problém s kmity, jak například můžeme vidět v 5. sekundě naměřeného průběhu (viz. obrázek 8.1). Tyto záchvěvy jsou naprosto ojedinělé, takže je nelze brát jako podstatnou závadu. V úseku od 35. do 40. sekundy průběhu, kdy jsme co nejrychleji kmitali volantem, se výsledek poměrně značně odchyluje od reference. Toto může být způsobeno nedokonalým odečtem odstředivých a setrvačných sil působící na akcelerometry na volantu nebo filtru dolní propustí, který bude pravděpodobně implementován v referenčním snímači ve volantu. Nelze s jistotou rozhodnout, jestli je chyba na straně testovaného zařízení nebo reference.

Ovšem při běžné jízdě automobilu, zejména na dálnici, není možné kmitat s volantem takto vysokou frekvencí. Proto si myslím, že tento problém není kritický pro naší aplikaci. Test bych tedy hodnotil jako úspěšný.



Obr. 8.1: Výsledek rekonstrukčního algoritmu pro reálný signál

Závěr

Cílem této bakalářské práce bylo navrhnout a sestrojit prototyp zařízení, které je schopné změřit úhel natočení volantu pomocí dat z akcelerometrů a tato data ve formě časových řad odeslat do počítače či jiného datového koncentrátoru. Dále toto zařízení otestovat v laboratorních podmínkách a zhodnotit použitelnost této metody měření úhlu natočení volantu.

Základním principům snímačů polohy a zrychlení se věnuje první kapitola, ve které jsou zpracovány především MEMS snímače, které jsou díky svým parametrům pro náš projekt nejvhodnější.

Návrhu zapojení pro sběr dat ze snímačů se věnují druhá, třetí a částečně i pátá kapitola. V druhé kapitole jsou popsána zařízení, která jsou v projektu použita, nebo jsme je alespoň plánovali použít. Jedná se především o desky GY-521, GY-91 a mikrokontrolér eses klon Arduino UNO. Třetí kapitola se zabývá výběrem komunikační sběrnice na propojení mikrokontroléru s měřícími deskami. Vybírali jsme mezi sběrnicí I2C a SPI. Nakonec jsme vybrali SPI sběrnici, protože umožňuje podstatě rychlejší komunikaci po sběrnici. Navíc I2C má problém s připojením většího počtu stejných zařízení k jedné sběrnici. Kvůli výběru SPI sběrnice jsme museli vyřadit desku GY-521, která podporuje pouze I2C rozhraní. Pátá kapitola mimo jiné řeší realizaci zapojení měřícího zařízení, ukládání naměřených dat do .csv souboru a jejich exportu do MATLABu.

Ve čtvrté kapitole se věnujeme rekonstrukčnímu algoritmu pro získání úhlu natočení z dat akcelerometrů. Nejprve jsme vytvořili jednoduchý model celého zařízení včetně výpočtu úhlu ve 2D, který jsme následně simulovali v prostředí MATLAB Simulink na několika testovacích signálech. Dále jsme do modelu přidali šum, abychom ověřili, že je algoritmus použitelný i pro skutečné snímače. Bohužel jsme zjistili, že v některých intervalech je algoritmus nestabilní, kvůli vysoké citlivosti funkcí arcsin a arccos. Tuto chybu se nám podařilo odstranit díky vhodné kombinaci výsledků obou větví rekonstrukčního algoritmu. Na závěr čtvrté kapitoly jsme výsledný 2D model transformovali do 3D prostoru.

Vylepšením rekonstrukčního algoritmu se zabývala šestá kapitola. Na začátku jsme určili aditivní chybu senzorů, kterou jsme následně kompenzovali. Druhá části je popsána nepřesnost, která je způsobena nevhodným umístěním senzorů na volant. Další část řeší vliv naklopení volantu a jeho kompenzaci. V této kapitole jsme také rozšířili algoritmus pro detekci několikanásobného otočení volantu. V závěru vylepšujeme algoritmus použitím komplementárního filtru.

V páté kapitole jsme následně jsme celé zařízení sestavili a umístili na volant. V osmé kapitole se věnujeme programu, díky kterému je Arduino schopné číst data ze senzorů a vypočítat úhel natočení volantu. Díky kalibrační proceduře je algoritmus

implementovaný v mikrokontroléru schopný kompenzovat velké množství vlivů, které do měření vnášejí nepřesnosti.

Otestovat zařízení v laboratorních podmínkách bohužel nebylo možné z důvodu omezeného přístupu studentů a doktorandů do školy. Zařízení jsme byli nuceni testovat v domácích podmínkách na herním volantu. Bohužel vnitřní snímač herního volantu nebyl jako referenční příliš vhodný, a to především kvůli pásmu necitlivosti v oblasti nulového natočení a saturaci na $\pm 90°$ natočení, přestože volant se mohl natočit o něco víc. I když bylo testování spíše orientační, můžeme prohlásit, že úhel naměřený systémem s akcelerometry odpovídal referenčnímu snímači. Pouze při velké rychlosti změny úhlu se úhly lišily. Tato odlišnost je podle mého názoru způsobena spíše dolnofrekvenční propustí implementované v referenčním snímači. Celé zařízení je tedy funkční a v laboratorních podmínkách použitelné.

Celá bakalářská práce slouží jako dokumentace výsledného zařízení včetně zdrojových kódů, které jsou součástí přílohy.

Literatura

[1] HSU, Tai-Ran. *MEMS and microsystems: design and manufacture.* Boston: McGraw-Hill, c2002. ISBN 0072393912.

[2] FRADEN, Jacob. *Handbook of modern sensors: physics, designs, and applications.* 4th ed. New York: Springer, c2010. ISBN 9781441964663.

[3] RIPKA, Pavel a Alois TIPEK. *Modern sensors handbook.* Newport Beach, CA: ISTE USA, 2007. ISBN 1905209665.

[4] Memsnet.org. 2020. *What Is MEMS Technology?.* [online] Dostupné z URL: https://www.memsnet.org/about/what-is.html [Citováno 6. října 2020].

[5] Stanford.edu. 2020. *MPU-9255 Product Specification.* [online] Dostupné z URL: https://stanford.edu/class/ee267/misc/MPU-9255-Datasheet.pdf [Citováno 10. října 2020].

[6] Stanford.edu. 2020. *MPU-9255 Register Map And Descriptions Revision 1.0.* [online] Dostupné z URL: https://stanford.edu/class/ee267/misc/MPU-9255-Register-Map.pdf [Citováno 10. října 2020].

[7] NAROM. 2020. *GY-91.*[online] Dostupné z URL: https://www.narom.no/undervisningsressurser/the-cansat-book/v6-2/getting-started-with-the-primary-mission/sensors-for-primary-mission-using-the-version-6-2-cansat-kit/gy-91/ [Citováno 10. října 2020].

[8] Qqtrading.com.my. 2020. *10-DOF Accelerometer, Gyroscope, Magnetic Compass, Pressure Sensor GY-91, MPU-9250 BMP280.* [online] Dostupné z URL: http://qqtrading.com.my/10-dof-gy-91-mpu9250-bmp280 <http://qqtrading.com.my/10-dof-gy-91-mpu9250-bmp280> [Citováno 10. října 2020].

[9] ProtoSupplies. 2020. *MPU-6050 GY-521 3-Axis Accel & Gryo Sensor Module - Protosupplies.* [online] Dostupné z URL: https://protosupplies.com/product/mpu-6050-gy-521-3-axis-accel-gryo-sensor-module/ [Citováno 11. října 2020].

[10] Invensense.tdk.com. 2020. *MPU-6000 And MPU-6050 Product Specification Revision 3.4..* [online] Dostupné z URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf [Citováno 11. října 2020].

[11] Arduino-shop.cz. 2020. *Eses Klon Arduino UNO R3 CH340.* [online] Dostupné z URL: https://arduino-shop.cz/docs/produkty/0/761/eses1459967190.pdf [Citováno 11. října 2020].

[12] Datasheetspdf.com. 2020. *Atmega328p Datasheet.* [online] Dostupné z URL: https://datasheetspdf.com/pdf-file/1057332/ATMEL/ATmega328P/1 [Citováno 11. října 2020].

[13] Learn.sparkfun.com. 2020. *Serial Communication - Learn.Sparkfun.Com.* [online] Dostupné z URL: https://learn.sparkfun.com/tutorials/serial-communication/all [Citováno 16. října 2020].

[14] Byteparadigm.com. 2020. *Introduction To I2C And SPI Protocols – Byte Paradigm – Speed Up Embedded System Verification.* [online] Dostupné z URL: https://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/?/article/AA-00255 [Citováno 16. října 2020].

[15] Dudáček, K., 2020. *Sériová Rozhraní SPI, Microwire, I2C A CAN.* [online] Dostupné z URL: http://home.zcu.cz/ dudacek/NMS/Seriova$_r ozhrani.pdf$ [Citováno 19. října 2020].

[16] GitHub. 2020. *Bolderflight/MPU9250.* [online]Dostupné z URL: https://github.com/bolderflight/MPU9250 [Citováno 10. listopadu 2020].

[17] ResearchGate. 2020. *Researchgate | Find And Share Research.* [online] Dostupné z URL: https://www.researchgate.net/figure/Basic-layout-of-accelerometer-In-the-mass-spring-damper-system-the-loading-force-drives$_f ig1_6$441258 [Citováno 10. listopadu 2020].

[18] Analog.com. 2020. *Mixed-Signal And Digital Signal Processing Ics | Analog Devices.* [online] Dostupné z URL: https://www.analog.com/en/analog-dialogue/articles/mems-accelerometers-as-acoustic-pickups.html [Citováno 10. listopadu 2020].

[19] Hindawi.com. 2020. *Figure 1 | Adaptive Fuzzy Sliding Mode Control Of MEMS Gyroscope With Finite Time Convergence.* [online] Dostupné z URL: https://www.hindawi.com/journals/js/2016/1572303/fig1 [Citováno 10. listopadu 2020].

[20] ResearchGate. 2020. *Researchgate | Find And Share Research.* [online] Dostupné z URL: https://www.researchgate.net/figure/Schematic-illustration-of-MEMS-Vibratory-Gyroscope$_f ig1_2$67594265 [Citováno 10. listopadu 2020].

[21] Pinterest. 2020. *Pin On Mp3 Player.*[online] Dostupné z URL: https://cz.pinterest.com/pin/836825174491483176/ [Citováno 11 listopadu 2020] *https://cz.pinterest.com/pin/836825174491483176/*

[22] Picotech.com. 2020. I²C - *Serial Protocol Decoding.* [online] Dostupné z URL: https://www.picotech.com/library/oscilloscopes/serial-protocol-decoding-i2c [Citováno 12. listopadu 2020].

[23] Programmersought.com. 2020. *SPI Protocol Analysis - Programmer Sought.* [online] Dostupné z URL: https://www.programmersought.com/article/32763154305/ [Citováno 13. listopadu 2020].

[24] Narkhede, P. et al. 2019. *'Least square estimation-based adaptive complimentary filter for attitude estimation',* Transactions of the Institute of Measurement and Control, 41(1), pp. 235–245. doi: 10.1177/0142331218755234.

[25] Pieter, J., 2021. *Reading a IMU Without Kalman: The Complementary Filter | Pieter-Jan.com.* [online] Pieter-jan.com. Dostupné na: https://www.pygame.org/docs/ref/joystick.html [Citováno 4. dubna 2021].

[26] Pygame.org. 2021. *pygame.joystick — pygame v2.0.1.dev1 documentation.* [online] Dostupné z URL: https://www.pygame.org/docs/ref/joystick.html [Citováno 24. března 2021].

[27] Learn.lboro.ac.uk. 2021. *Numerical Determination of Eigenvalues and Eigenvectors.* [online] Dostupné z URL: https://learn.lboro.ac.uk/archive/olmp/olmp_resources/pages/workbooks_1_50_jan2008/W [Citováno 28. března 2021].

Seznam symbolů a zkratek

MEMS	Micro electro-mechanical systems - Mikro elektricko-mechanický systém
MSBFIRST	Most significant bit first - pomocí sériové komunikace se odešle nejvýnamnější bit jako první
dps	Degrees per second - stupně za sekundu
RISC	Model procesoru s řadičem s omezenou instrukční sadou
SPI	Serial Peripheral Interface - datová sběrnice
I2C	Inter-Integrated Circuit - datová sběrnice
MOSI	Master out slave in - datový vodič SPI sběrnice
MISO	Master in slave out - datový vodič SPI sběrnice
SCK	Serial clock - vodič s hodinovým signálem SPI sběrnice
SS	Slave select - vodič pro výběr slave zařízení SPI sběrnice
SDA	Serial data - datový vodič I2C sběrnice
SCLK	Serial clock - vodič s hodinovým signálem I2C sběrnice
VIN	Pin GY-91 - napájecí pin, pro nás platí, že má napětí 5V
GND	Pin GY-91 a GY-521 - zemnící pin. Má napětí 0V
3V3	Pin GY-91 - pro externí napájení, na výstupu má 3,3V
SCL	Pin GY-91 - pro hodinový signál
SDA	Pin GY-91 - pro datovou komunikaci
SD0/SAO	Pin GY-91 - pro datovou komunikaci, nebo nastavení I2C adresy
NCS	Pin GY-91 - pro určení aktivity zařízení
CSB	Pin GY-91 - pro určení aktivity senzoru BMP280
VCC	Pin GY-521 - napájecí pin, pro nás platí, že má napětí 5V
AD0	Pin GY-521 - pro nastavení I2C adresy
INT	Pin GY-521 - pro vyvolání přerušení

XDA	Pin GY-521 - pro připojení datového signálu modulů třetích stran
XCL	Pin GY-521 - pro připojení hodinového signálu modulů třetích stran
H	High - označení pro vysokou úroveň napětí (typicky 5V)
L	Low - označení pro nízkou úroveň napětí (typicky 0V)

Seznam příloh

A Zdrojové kódy

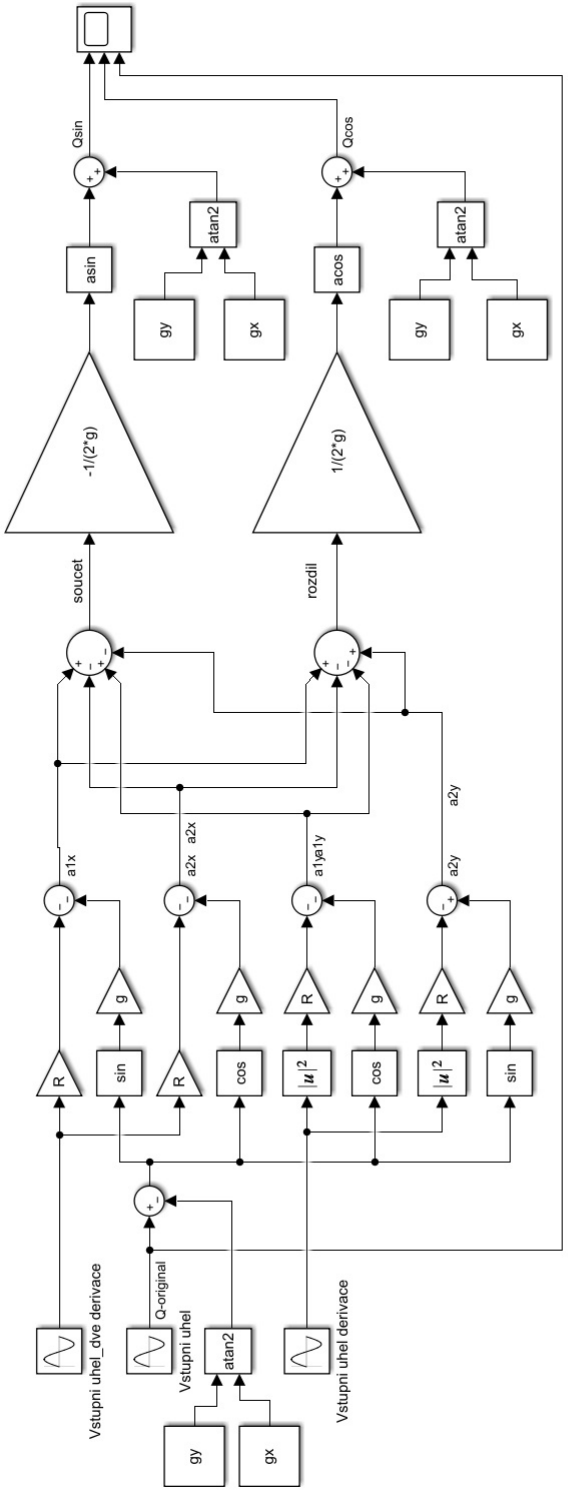
A.1 Zdrojový kód pro MATLAB Simulink

Ukázka souboru s daty pro MATLAB Simulink `DataSimulace.m`, sloužící pro nahrání vstupních proměnných pro simulink do workspace.

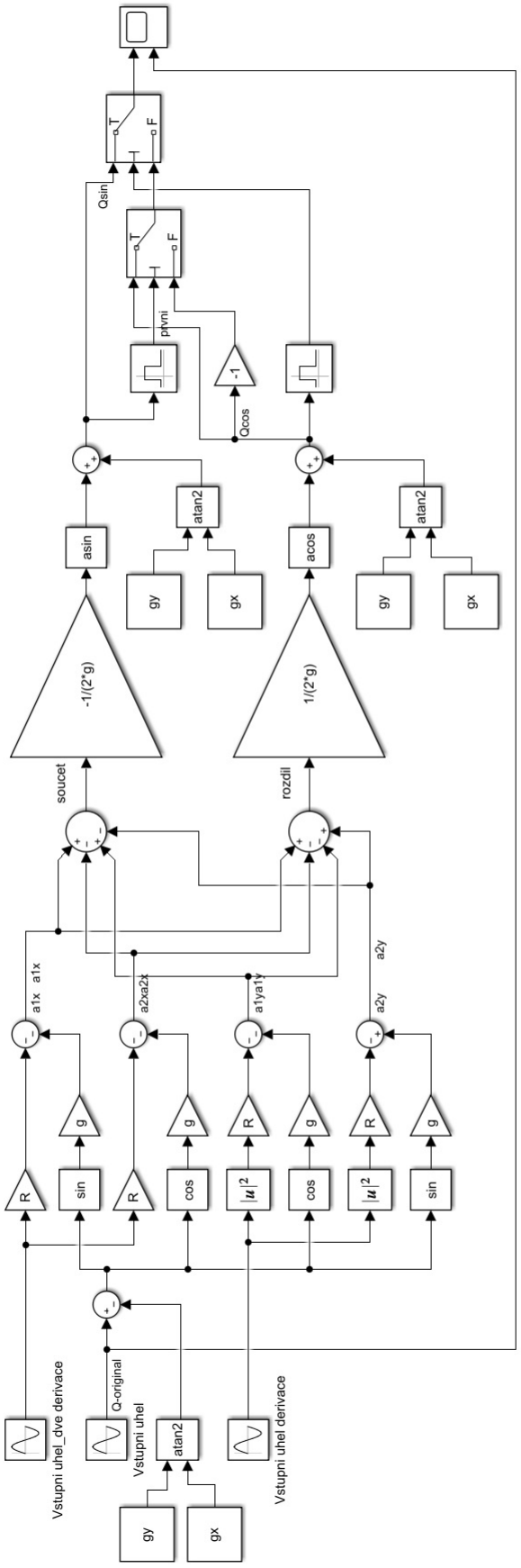
```
%% inicializace
clc
clear all
close all
%% Data do simulinku
A=5*pi/180;
%A=20;
R=0.24;%polomer volantu
gx=9.81;
gy=0.00000000001;
pi=3.14;
T = 10;
f=1/T;
w=2*pi*f;%omega
d_odm = sqrt(2);

%% Simulacni_sum
load('A1x.mat')
load('A1y.mat')
load('A2x.mat')
load('A2y.mat')
load('data_Mih.mat')

A1x_noise= A1x-mean(A1x);
A1y_noise= A1y-mean(A1y);
A2x_noise= A2x-mean(A2x);
A2y_noise= A2y-mean(A2y);

t = ((0:1:4999).*0.01)';

Data_a1x = [t A1x_noise];
Data_a1y = [t A1y_noise];
```

```
33 Data_a2x = [t A2x_noise];
34 Data_a2y = [t A2y_noise];
```

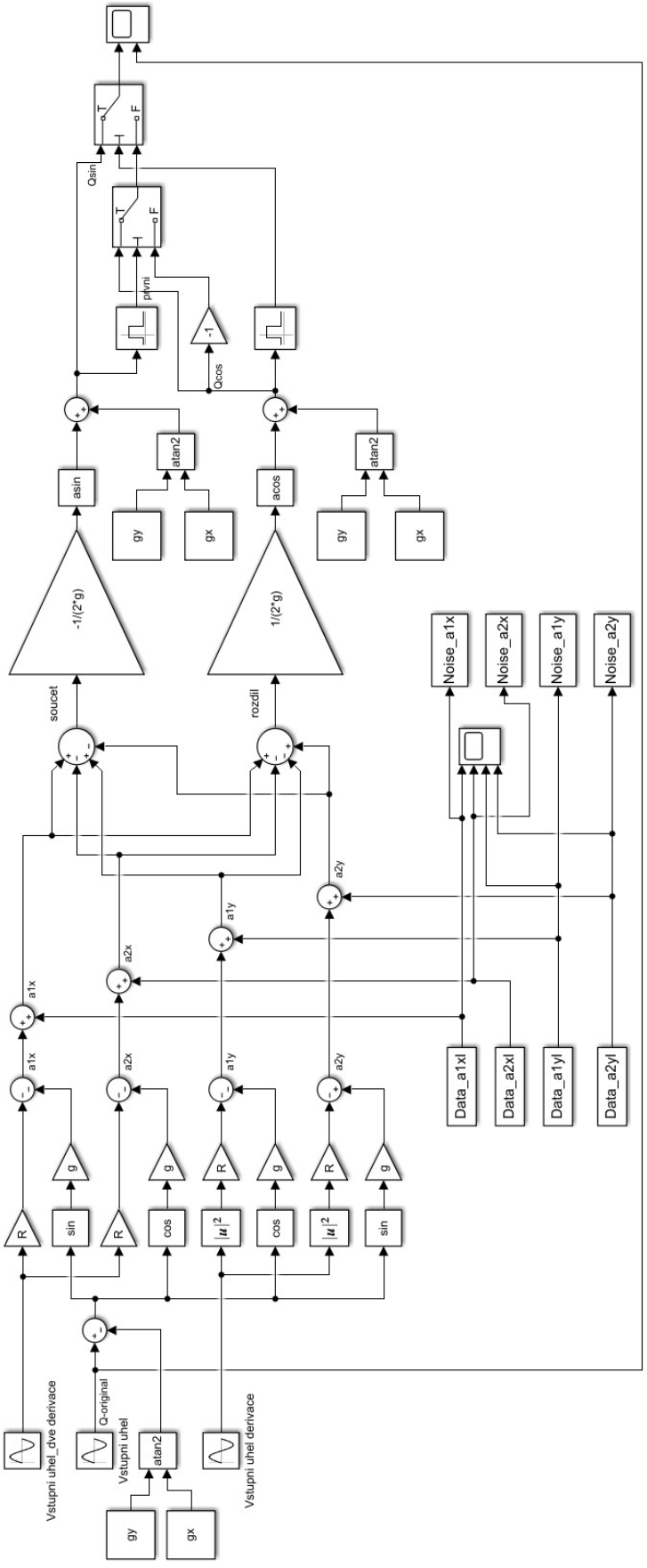
B Simulační schémata pro Simulink



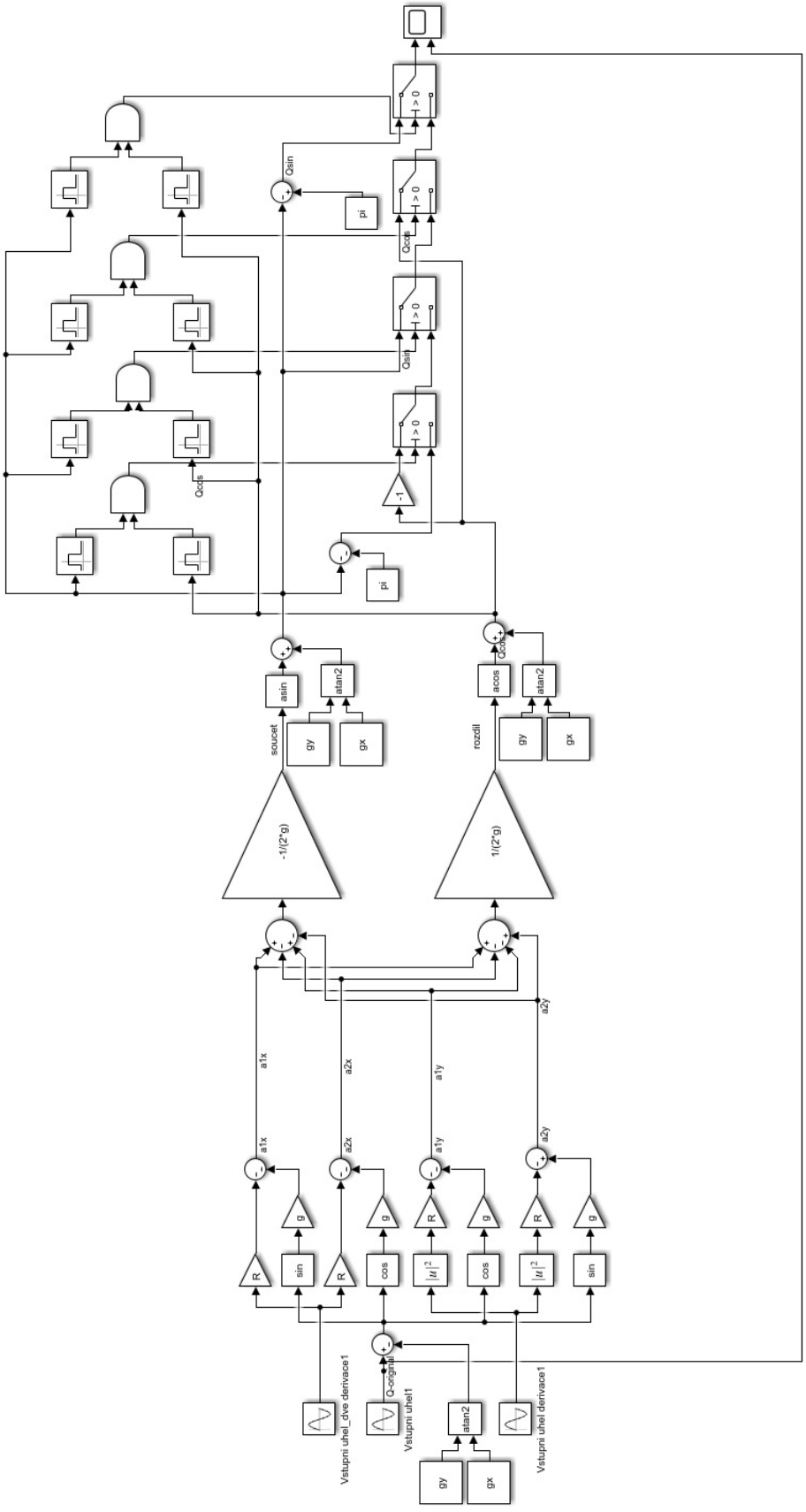
Obr. B.1: Základní 2D model bez šumu



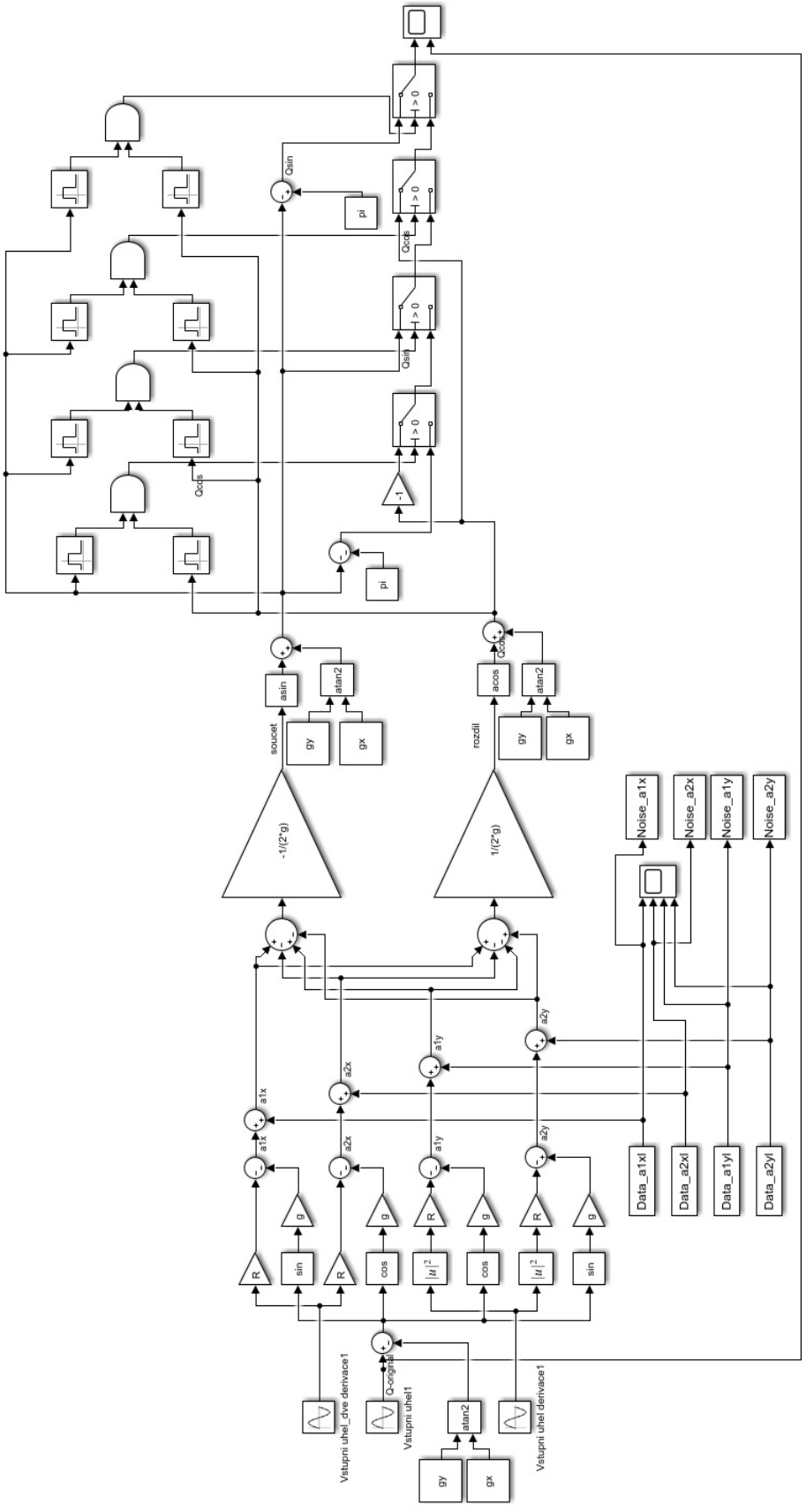
Obr. B.2: Základní 2D model bez šumu



Obr. B.3: Základní 2D model s šumem



Obr. B.4: Finální 2D model bez šumu



Obr. B.5: Finální 2D model s šumem

C Obsah přiloženého CD

Zde najdete kompletní seznam všech příloh, které se nachází na přiloženém CD.

- `/` ..... kořenový adresář přiloženého CD
  - `Arduino` ..... Zdrojový kód pro ovládání Arduina
    - `Final_uhel_volantu`
      - `Final_uhel_volantu.ino`
  - `SPI_MPU_9250` ..... Zdrojový kód knihovny použité v programu pro Arduino
    - `SPI_MPU_9250.h`
    - `SPI_MPU_9250.cpp`
  - `Fotografie` ..... Fotografie výsledného zařízení
    - `realizace_volantu.png`
    - `uchyceni_senzoru2.jpg`
    - `uchyceni_senzoru3.jpg`
    - `uchyceni_senzoru3_detail.jpg`
  - `Grafy` ..... Grafy vyexportované z MATLABu a požité v práci
    - `2D_advanced_graph_basic.pdf`
    - `2D_advanced_graph_ramp.pdf`
    - `2D_advanced_graph_real.pdf`
    - `2D_advanced_noise_basic.pdf`
    - `2D_advanced_noise_real.pdf`
    - `2D_easy_Graf_pouze_sum.pdf`
    - `2D_easy_Graf_Q_noise.pdf`
    - `2D_easy_Graf_rozdil_Q_noise.pdf`
    - `2D_final_graf_final.pdf`
    - `2D_full_Graf_Q_bez_sumu.pdf`
    - `2D_full_graf_rampa.pdf`
    - `graf_porovnani.pdf`
    - `rozdeleni_grafu.pdf`
  - `Matlab` ..... Simulace matematických modelů zařízení v MATLAB Simulink
    - `Model_1_zakladni_bez_sumu`
      - `A1x.mat`
      - `A1y.mat`
      - `A2x.mat`
      - `A2y.mat`
      - `Acos_vstup.mat`
      - `Asin_vstup.mat`
      - `data_Mih.mat`
      - `Data_simulace.m`
      - `simulacni_schema.slx`
    - `Model_2_zakladni_2D_model_bez_sumu`
      - `A1x.mat`
      - `A1y.mat`
      - `A2x.mat`
      - `A2y.mat`
      - `data_Mih.mat`

- Data_simulace.m
- simulacni_schema.slx

Model_3_zakladni_2D_model_s_sumem
- A1x.mat
- A1y.mat
- A2x.mat
- A2y.mat
- data_Mih.mat
- Data_simulace.m
- simulacni_schema.slx

Model_4_finalni_2D_model_bez_sumu
- A1x.mat
- A1y.mat
- A2x.mat
- A2y.mat
- data_Mih.mat
- Data_simulace.m
- simulacni_schema.slx

Model_5_finalni_2D_model_s_sumem
- A1x.mat
- A1y.mat
- A2x.mat
- A2y.mat
- data_Mih.mat
- Data_simulace.m
- simulacni_schema.slx

Nakresy ..... Vlastní schémata použitá v práci
- 2D_model.png
- 3_akcel_graf.jpg
- 3D_graf.jpg
- Arduino_zapojeni.png
- Arduino_zapojen1i.png
- g0xy.png
- jednotko_kruznice.jpg
- nakloneni_IMU.png
- nakloneni_IMU_kalibrace.png
- Nakloneni_volantu1.png
- Nakloneni_volantu2.png
- Soubor_vsech_nakresu.docx
- uchyceni_senzoru1.png

Python ..... Zdrojové kódy programů pro čtení dat ze senzorů natočení do PC
- ReadDataFromStWheel.py
- serial2csv.py

Text_bakalarske_prace ..... Hlavní soubor pro sazbu práce
- Text_bakalářské_práce.pdf