

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Dlouhodobý monitoring pohybu lidského těla v prostoru

Milan Němý

Duben 2014

Vedoucí práce: Ing. Matouš Pokorný



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Milan Němý**

Studijní program: **Kybernetika a robotika**
Obor: **Senzory a přístrojová technika**

Název tématu česky: **Dlouhodobý monitoring pohybu lidského těla v prostoru**

Název tématu anglicky: **Long Term Monitoring of Human Body Motion in Area**

Pokyny pro vypracování:

Realizujte jednoduché přenosné zařízení umožňující dlouhodobý záznam trajektorií pohybů lidského těla s využitím vývojového přípravku Raisonance EvoPrimer s mikrokontrolérem z rodiny STM32F a inerciální vztahné jednotky EvoPrimer Extension Board. Do zařízení implementujte kalibraci inerciální vztahné jednotky a záznam naměřených dat na microSD kartu. Se zařízením proveďte experiment zaměřený na relevanci a reprodukovatelnost výsledků. Diskutujte možnosti optimalizace těchto parametrů.


Seznam odborné literatury:

- [1] J. Fraden: Handbook of modern sensors. Berlin: Springer, 2010.
- [2] B. Siciliano and O. Khatib, Ed.: Handbook of Robotics. Berlin: Springer, 2008.
- [3] S. Thrun, W. Burgard and D. Fox: Probabilistic Robotics. Cambridge, MA: The MIT Press, 2005.

Vedoucí bakalářské práce: Ing. Matouš Pokorný (K 13135)

Datum zadání bakalářské práce: 26. listopadu 2013

Platnost zadání do¹: 30. ledna 2015


Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry




Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 11. 2013

¹ Platnost zadání je omezena na dobu dvou následujících semestrů.

Poděkování / Prohlášení

Děkuji Ing. Matouši Pokornému za pomoc při vedení bakalářské práce. Mé poděkování patří též Ing. Martinu Šipovi za cenné rady týkající se inerciální navigace a kalibrace senzorů.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 22. 5. 2014

.....

Abstrakt / Abstract

Tato bakalářská práce řeší problematiku kalibrace MEMS senzorů IMU jednotek a sledování pohybu lidského těla pomocí dat z nich naměřených. Cílem bylo nalezení jednoduchých kalibračních procedur, které zvládne provést běžný uživatel i v domácích podmínkách, a dále použití některého z algoritmů pro určení orientace a polohy jednotky v prostoru.

Práce představuje hned několik algoritmů pro snadnou kalibraci a následné zpracování, především pomocí Kalmanova filtru.

Vytvořené řešení poskytuje možnost sledovat nejen orientaci, ale v krátkém časovém intervalu i rychlost a polohu. Dále je navržen postup řešení pomocí přídatných markerů pro další zlepšení sledovaných parametrů.

Klíčová slova: IMU, MEMS senzory, kalibrace, sledování pohybu, Kalmanův filtr.

This bachelor's thesis deals with the calibration of MEMS sensors in IMUs and human body motion tracking using the data measured with an IMU. The aim was to find simple calibration procedures that an ordinary user would be able to perform at home, and to apply some of the algorithms for determining a unit's orientation and position in space.

The thesis presents several methods for easy calibration and subsequent processing, mainly using the Kalman filter.

The proposed solution allows tracking not only orientation, but also velocity and position, though only in a short time period. The thesis also suggests using additional markers in order to improve measuring accuracy.

Keywords: IMU, MEMS sensors, calibration, motion tracking, Kalman filter.

Title translation: Long Term Monitoring of Human Body Motion in Area

Obsah /

1 Úvod	1	6.6 Fixed-gain observer	35
2 Použité vybavení	3	6.7 Kalmanův filtr	36
2.1 Vývojový kit Raisonance STM32 Primer2	3	6.7.1 Inicializace	38
2.2 Inerciální měřící jednotka (IMU)	4	6.7.2 Odhad	38
2.3 CircleOS	6	6.7.3 Korekce	39
2.3.1 Činnost CircleOS	6	6.7.4 Použitý model	39
3 Přenos naměřených dat	8	6.7.5 Kompenzace zrychlení ...	40
3.1 Vyčítání dat ze senzorů	8	6.8 Experimentální ověření	41
3.2 Ukládání dat na paměťovou kartu	10	6.8.1 Experiment – otáčení kolem jedné osy	41
3.3 Virtual COM Port	10	6.8.2 Experiment – data se zrychlením	41
4 Program pro Primer2	13	6.8.3 Experiment – určení orientace při chůzi	43
4.1 Inicializace	13	6.8.4 Experiment – určení polohy při chůzi	44
4.2 Hlavní programová smyčka	13	7 Závěr	48
4.3 Obsluha přerušení	13	Literatura	49
4.4 Kalibrace senzorů	13	A Inertial Center 1.0.1	51
4.5 Flow diagram	14	A.1 Zapnutí aplikace	51
4.6 Snímky obrazovky programu ..	15	A.2 Hlavní obrazovka	51
5 Kalibrace IMU jednotky	16	A.3 Záznam dat na paměťovou kartu	51
5.1 Chybový model jednotky	16	A.4 Kompenzace	52
5.1.1 Náhodné chyby	17	A.5 Kalibrace	52
5.1.2 Systematické chyby	17	A.6 Kalibrace akcelerometru	52
5.2 Kalibrace a kompenzace sen- zorů	17	A.7 Kalibrace magnetometru	52
5.2.1 Parametry kompenzač- ního modelu	17	A.8 Kalibrace gyroskopu	53
5.2.2 Výpočet parametrů kompenzačního modelu ..	18	B Inertial Center for PC	54
5.3 Kalibrace akcelerometru	20	B.1 Komunikace	55
5.4 Kalibrace magnetometru	24	B.2 Vizualizace orientace pomocí 3D kostky	55
5.5 Kalibrace gyroskopu	29	B.3 Kalibrace gyroskopu	55
5.6 Implementace kalibrace	31	B.4 Kalibrace akcelerometru	56
5.6.1 Kalibrace v Primeru	31	B.5 Kalibrace magnetometru	56
5.6.2 Kalibrace v PC – Iner- tial Center for PC	31	B.6 Režim kompasu	56
6 Sledování pohybu	32	B.7 Implementace algoritmu fixed-gain observer	56
6.1 Souřadná soustava	32	B.8 Kalmanův filtr	57
6.2 Eulerovy úhly	32		
6.3 Kvaterniony	33		
6.4 Určení orientace z výstupu gyroskopu	34		
6.5 Určení orientace z výstupu akcelerometru a magnetome- tru – kompas	34		

Tabulky / Obrázky

2.1. Hlavní charakteristiky použitých senzorů	5
3.1. ODR jednotlivých senzorů	9
5.1. Znaménková konvence akcelerometru	21
5.2. Výsledky kalibrace akcelerometru – model 1	22
5.3. Výsledky kalibrace akcelerometru – model 2	23
5.4. Prahování gyroskopu	29
5.5. Výsledky kalibrace gyroskopu .	30
2.1. STM32 Primer2	3
2.2. IMU jednotka	4
2.3. Životní cyklus CircleOS	6
2.4. Rozvržení paměti s CircleOS	7
4.1. Flow diagram programu	14
4.2. Snímek obrazovky – hlavní obrazovka č. 1	15
4.3. Snímek obrazovky – hlavní obrazovka č. 2	15
4.4. Snímek obrazovky – výběr senzoru pro kalibraci	15
5.1. Chybový model IMU jednotky	16
5.2. Znázornění neortogonality os a chyby měřítka	18
5.3. Souřadnicový systém IMU	20
5.4. Náhodné 3D rotace magnetického senzoru	25
5.5. Surová data magnetometru před jeho kalibrací	28
5.6. Data magnetometru po kompenzaci hard-iron efektu a chyby měřítka	28
5.7. Data magnetometru po konečné kompenzaci	28
6.1. Schéma fixed-gain observer	36
6.2. Rotace jednotky kolem jedné osy	41
6.3. Rychlé rotace jednotky	42
6.4. Rychlé rotace jednotky (R-adaptivní algoritmus)	42
6.5. Rychlé rotace jednotky (prahování zrychlení)	43
6.6. Záznam chůze (orientace)	44
6.7. Záznam chůze č. 2 (orientace) .	45
6.8. Záznam chůze č. 2 (neupravená trajektorie)	45
6.9. Záznam chůze č. 2 (trajektorie X-Y, po filtraci)	46
6.10. Záznam chůze č. 2 (trajektorie X-Z, po filtraci)	46
A.1. Popis úvodní obrazovky Inertial Center 1.0.1	51
B.2. Základní popis Inertial Center for PC	54

Kapitola 1

Úvod

Jedním z úkolů dnešního biomedicínského inženýrství je poskytnout objektivní podklady pro diagnostiku onemocnění muskuloskeletálního systému a monitorování pokroku jeho léčby. Tyto podklady však mohou vznikat na různém základě a být více či méně relevantní. Proto se v poslední době stává standardem kvantitativní měření a vyhodnocování. Jako efektivní řešení pro analýzu pohybů a výkonu lidského těla se pak ukazuje snímání pohybu. Nejčastěji užívanými technologiemi jsou především systémy postavené na optickém nebo elektromagnetickém základu.

Obvyklé optické metody pro snímání pohybu jsou spolehlivé a přesné. Používají značky (angl. markers), jako například LED diody nebo pasivní odrazové plošky a vyhodnocují jejich přesnou polohu pomocí dat z kamer pomocí triangulace. Nicméně systémy postavené na tomto principu nejsou snadno přenositelné, a tak takřka nepoužitelné pro mnohé aplikace.

Magnetické metody používají většinou permanentní magnet a senzor magnetického pole. Tento přístup však dokáže pouze poskytnout polohu bodového zdroje a následné výpočty rotace a zrychlení, počítané jako derivace, vnáší do měření další šum. V neposlední řadě je potřeba zmínit, že je téměř nemožné sledovat polohu více zdrojů najednou.

Často je pro potřeby léčby a rehabilitace vyžadováno používání snímacího systému nejen v nemocničním zařízení, ale i v domácím prostředí a každodenním životě. Proto je nezbytné, aby tyto systémy byly kompaktní a vhodné k pohodlnému nošení na lidském těle. V tomto směru se velmi slibně jeví nasazení inerciálních měřících jednotek (angl. inertial measurement unit, IMU). Dostupnost levných a malých MEMS senzorů umožnila stavbu kompaktních a lehkých inerciálních modulů, které se většinou skládají z tříosého akcelerometru, gyroskopu a magnetometru. Teoreticky, ideálně zkalibrovaná IMU jednotka měří úhlovou rychlost, zrychlení a magnetické pole Země. A při znalosti počáteční polohy a orientace v prostoru nesou tyto signály dostatečnou informaci o kinematice této jednotky.

Na trhu se v současné době pohybuje několik výrobců systémů snímání pohybu. Mezi nejvýznamnější se řadí Vicon Motion Systems¹⁾, jejichž metody jsou založené především na kombinaci dat z několika IMU jednotek a videokamer. Kromě toho také současně využívají tlakové podložky a měření EMG signálu. Tento přístup je velmi přesný, nicméně kvůli velkému množství nákladné techniky je prakticky nepřenositelný. Firma Vicon pomáhá chirurgům a fyzioterapeutům s objektivním hodnocením velké škály patologií spojených s pohybovým aparátem jako např. mozková obrna, mrtvice a Parkinsonova choroba. Dále se zabývá analýzou sportovních výkonů a evaluací tělesných náhrad.

V kategorii komerčních IMU jednotek nalezneme kupříkladu MTx²⁾ (Xsens Inc.) nebo InertiaCube3³⁾ (InterSense Inc.). Ty sice nabízí poměrně dobré parametry (statická

¹⁾ <http://www.vicon.com/>

²⁾ <http://www.xsens.com/products/mtx/>

³⁾ <http://www.intersense.com/pages/18/11/>

přesnost: roll/pitch $< 0,5^\circ$, heading $< 1^\circ$), jejich cena v základním provedení ovšem přesahuje 2000\$.

Mým cílem tak bylo použít vývojový kit *Primer2* s přídatnou IMU jednotkou a ukázat, že je možné sestavit a naprogramovat relativně přesnou IMU jednotku za zlomek ceny, která by byla použitelná běžným uživatelem. Představou bylo implementovat do jednotky jednoduchou kalibraci, proveditelnou běžným uživatelem v domácích podmínkách a umožnit nahrávání dat na SD kartu. Následné zpracování v počítači by mělo umět co nejpřesněji určit orientaci jednotky v prostoru.

Oproti běžným komerčním IMU jednotkám jsem se pokusil i o určení pozice dvojitou integrací naměřeného zrychlení. Avšak vzhledem k použití relativně levných MEMS senzorů a snadné, tedy ne zcela optimální kalibrace, předpokládám, že určení této pozice bude zatíženo poměrně velkou chybou. Vzhledem k tomu, že senzory mnou použité IMU jednotky jsou srovnatelné se senzory jednotky MTx a předpokládám chybu nedokonalosti kalibrace, odhaduji přesnost určení polohových úhlů $< 2^\circ$. Tomu odpovídá chyba v určení polohy rostoucí rychlostí přibližně 3,4 m/s [1]. Teda např. poloha vypočítaná za 10 sekund pohybu se bude lišit od té skutečné o 34 m.

Struktura této bakalářské práce je koncipována chronologicky, v logicky navazujících kapitolách. Součástí každé z nich je (pokud je to vhodné): teoretický úvod, implementace, rešerše a dílčí zhodnocení. Pro snadnější orientaci zde uvádím jejich stručný obsah:

Kapitola 2 se zabývá popisem použitého HW a SW. Kapitoly 3 a 4 představují jednoduché implementace, po řadě, přenosu naměřených dat a měřicího programu pro *Primer2*. Na ně poté navazuje kapitola 5, která jak teoreticky rozebírá proces kalibrace a kompenzace, tak navrhuje konkrétní kalibrační řešení pro jednotlivé senzory IMU jednotky a ukazuje naměřené výsledky. Kapitola 6 popisuje a porovnává několik způsobů, jak lze sledovat pohyb pomocí zkalibrované IMU jednotky; začíná teoretickým rozбором jednotlivých metod, načež prezentuje jejich experimentální ověření a vyvozuje z nich závěry o jejich použitelnosti a navrhuje možná vylepšení. Kapitola 7 je věnována celkovému zhodnocení bakalářské práce. Ta je uzavřena dvěma přílohami s návody k použití vyvinutých programů.

Kapitola 2

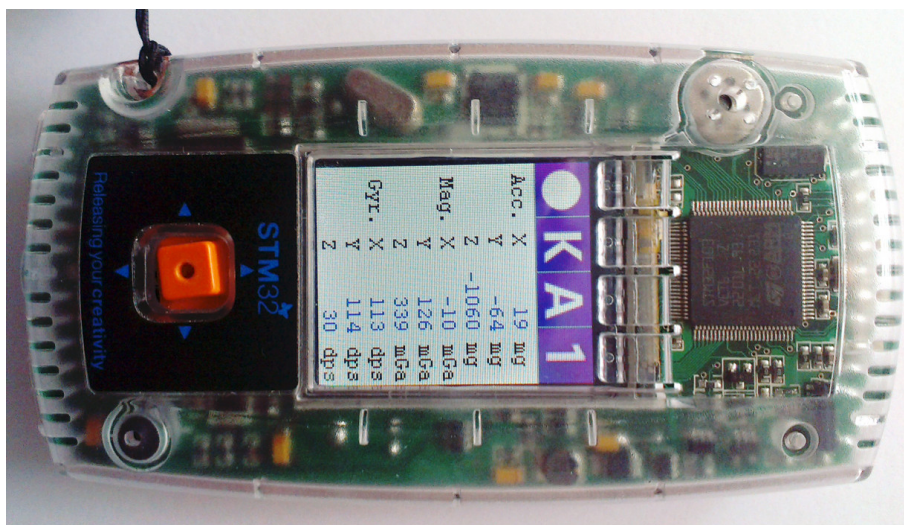
Použité vybavení

K realizaci bakalářská práce byl použit vývojový kit Raisonance STM32 Primer2 s přídatnou inerciální vztažnou jednotkou.

2.1 Vývojový kit Raisonance STM32 Primer2

Vývojový kit STM32 Primer2 od firmy Raisonance, dále jen Primer, je navržen jako přenosná, kompaktní a cenově dostupná jednotka určená pro začínající vývojáře a programátory vestavěných zařízení, kteří se chtějí seznámit se 32bitovými mikrokontroléry s jádrem ARM Cortex-M3 řady STM32F1xx. Primer se vyznačuje následujícími parametry¹⁾:

- mikrokontrolér STM32F103VE (ARM Cortex-M3 32bitový RISC procesor, 72 MHz, 512 KB Flash, 64 KB SRAM)
- dotykový TFT displej (rozlišení: 128x160, barevná hloubka: 24 bitů)
- 4směrový joystick a tlačítko
- MEMS 3D akcelerometr
- slot na Micro SD kartu
- uživatelský USB konektor
- IrDA rozhraní
- mikrofon a reproduktor
- 20pinový rozšiřující konektor pro přístup k rozhraním SPI, I2C, USART, CAN a analogovým a digitálním I/O
- Li-Ion akumulátor
- programátor a debugger RLINK



Obrázek 2.1. Vývojový kit STM32 Primer2.

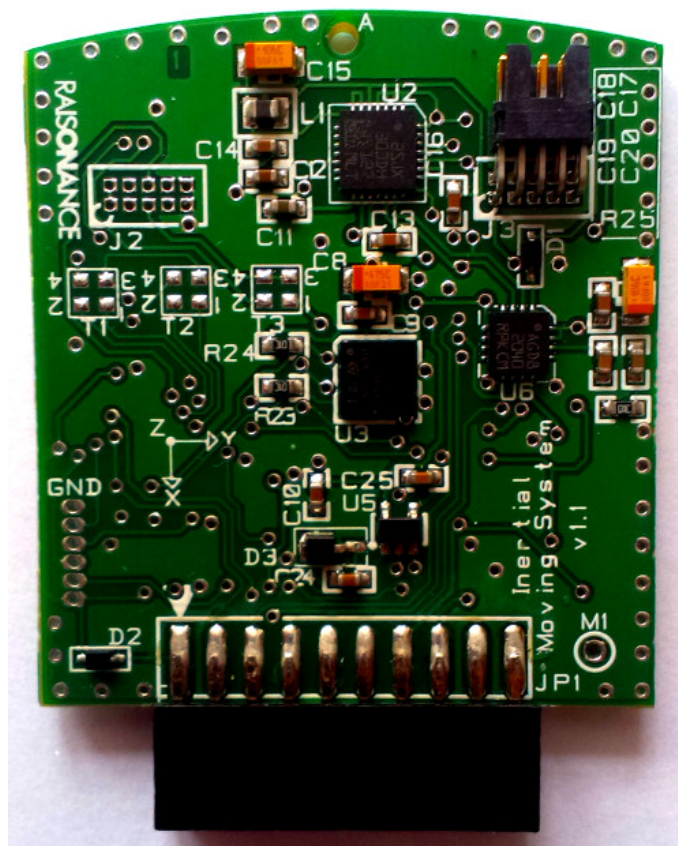
¹⁾ <http://www.stm32circle.com/resources/stm32primer2.php>

2.2 Inerciální měřicí jednotka (IMU)

K Primeru je možné dokoupit inerciální měřicí jednotku (angl. inertial measurement unit – IMU), která se snadno připojí pomocí 20pinového rozšiřujícího konektoru. Na této jednotce se nachází celkem tři nezávislé senzory od firmy STMicroelectronics:

- LSM303DLM – tříosý magnetometr (magnetorezistor) a tříosý akcelerometr (kapacitní) [2]
- L3G4200D – tříosý gyroskop (kapacitní) [3]
- LPS001D – senzor tlaku s teplotní kompenzací (piezorezistivní) [4]

Ve všech případech se jedná o „inteligentní senzory“, které kromě vlastní funkce měření dané veličiny poskytují celou řadu dalších funkcí, jako například: snížení vlivu jiných fyzikálních veličin (např. teploty), autodiagnostiku nebo filtraci. Senzory jsou vybaveny rozhraním pro číslicovou komunikaci, a je tak umožněn přenos různých hodnot měřených veličin a vzdálená parametrizace pomocí zápisů do vnitřních registrů. Na webové stránce výrobce jsou k dispozici katalogové listy jak k IMU jednotce [5], tak i k jednotlivým senzorům ([2], [3], [4]). V nich je uvedeno, že senzory jsou vybaveny komunikačním rozhraním SPI i I2C. Nicméně senzory na IMU jednotce mají na společnou sběrnici připojené pouze piny rozhraní I2C. Charakteristika senzorů v použité IMU jednotce, s kterými se dále pracuje, je shrnuta v tabulce 2.1.



Obrázek 2.2. IMU jednotka.

	LSM303DLM	LSM303DLM	L3G4200D
druh	magnetometr	akcelerometr	gyroskop
rozměry	5 × 5 × 1 mm	5 × 5 × 1 mm	4 × 4 × 1,1 mm
rozsah	±1,3/±1,9/±2,5/±4,0/ ±4,7/±5,6/±8,1 gauss	±2/±4/±8 g	±250/±500/ ±2000 dps
rozlišení	1100(XY)/980(Z) LSB/gauss při ±1,3 gauss; 855(XY)/760(Z) LSB/gauss při ±1,9 gauss; 670(XY)/600(Z) LSB/gauss při ±2,5 gauss; 450(XY)/400(Z) LSB/gauss při ±4,0 gauss; 400(XY)/355(Z) LSB/gauss při ±4,7 gauss; 330(XY)/295(Z) LSB/gauss při ±5,6 gauss; 230(XY)/205(Z) LSB/gauss při ±8,1 gauss;	1 mg/digit při ±2 g 2 mg/digit při ±4 g 8 mg/digit při ±8 g	8,75 mdps/digit při ±250 dps; 17,50 mdps/digit při ±500 dps; 70 mdps/digit při ±2000 dps
zero-rate level (hodnota při nulových vstupech)	N/A	±60 mg při ±2 g	±10 dps při ±250 dps; ±15 dps při ±500 dps; ±75 dps při ±2000 dps
výstupní frekvence dat (ODR)	0,75/1,5/3,0/7,5/15/ 30/75/220 Hz	50/100/400/1000 Hz	100/200/400/800 Hz
šířka pásma	N/A	37 Hz při ODR = 50 Hz; 74 Hz při ODR = 1000 Hz; 292 Hz při ODR = 400 Hz; 780 Hz při ODR = 1000 Hz;	12,5/25 Hz při ODR = 100 Hz; 12,5/25/50/70 Hz při ODR = 200 Hz; 20/25/50/110 Hz při ODR = 400 Hz; 30/35/50/110 Hz při ODR = 800 Hz
napájecí napětí	3 V	3 V	3 V
proudová spotřeba	0,36 mA	0,36 mA	6,1 mA

Tabulka 2.1. Základní charakteristiky použitých senzorů IMU jednotky.

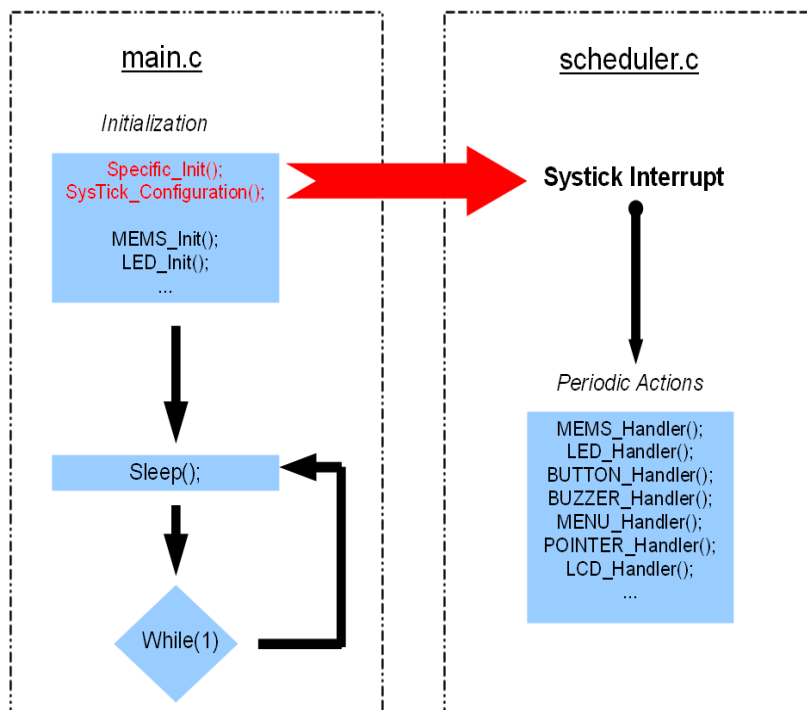
2.3 CircleOS

Pro podporu rychlého vývoje aplikací pro rodinu mikrokontrolérů STM32 je určen operační systém CircleOS¹). Jedná se o multiaplikační open source operační systém, na jehož vývoji se podílí firma Raisonance, a jehož cílem je poskytnutí přívětivého aplikačního rozhraní (API) pro přístup k hardwarovým periferiím i uživatelskému rozhraní. CircleOS se stará například o obsluhu MEMS akcelerometru, generátoru tónů, LCD displeje, menu, souborového systému, spouštění aplikací aj. Aktuálně (duben 2014) je dostupný ve verzi 4.61 a je neustále vyvíjen a rozšiřován. Významným dílem k tomu přispívá i rozsáhlá komunita uživatelů Primerů, která se sdružuje na webovém fóru projektu²).

2.3.1 Činnost CircleOS

Činnost CircleOS se skládá z několika fází. Zaprvé se jedná o inicializaci, při které probíhají veškeré hardwarové konfigurace a nastavuje se systémový periodický časovač (SysTick). Druhou fází je obsluha přerušení volaná SysTickem, kdy systém vyhodnocuje LED, tlačítka a LCD. Poslední a nejdůležitější fází je obsluha aplikací. Aplikace je spuštěna uživatelem přes menu a následně CircleOS zavolá její inicializační rutinu. Poté se opakovaně volá samotný program (Application Handler) a to do té doby, než je vrácena hodnota MENU_LEAVE. Při svém běhu může aplikace volně používat CPU a všechnu RAM paměť, kterou nezabírá operační systém. Životní cyklus CircleOS je schématicky znázorněn na obr. 2.3.

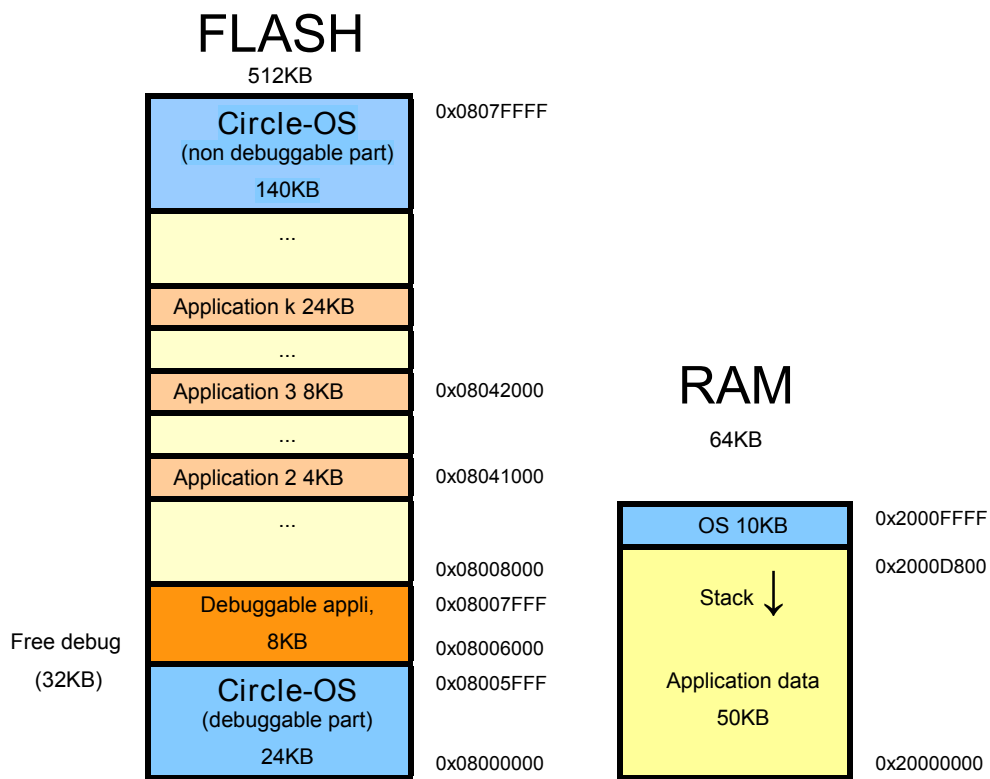
Firmware CircleOS zabírá 164 KB Flash paměti a 10 KB RAM. Zbýlých 349 KB je pak dostupných pro aplikace. Ty mohou být ale programovány jen po 2 KB blocích, viz obr. 2.4



Obrázek 2.3. Životní cyklus CircleOS [6].

¹) <http://www.stm32circle.com/projects/circleos.php>

²) <http://www.stm32circle.com/forum/index.php>



Obrázek 2.4. Příklad rozvržení paměti s CircleOS [6].

Kapitola 3

Přenos naměřených dat

Jedním z dílčích úkolů zadání bylo zajistit vyčítání dat ze senzorů IMU jednotky, následné ukládání na paměťovou kartu a jejich vysílání do PC pomocí rozhraní USB a protokolu Virtual COM Port.

Veškerý software/firmware byl napsán v integrovaném vývojovém prostředí Ride7 v jazyce ANSI C.

3.1 Vyčítání dat ze senzorů

Všechny senzory na IMU jednotce jsou připojeny k I2C sběrnici. Komunikace s nimi pak probíhá pomocí naadresování konkrétního jednoho senzoru a vyčítání dat nebo nastavení parametrů pomocí čtení, resp. zápisu do registrů daného senzoru.

Jako základ programu pro vyčítání dat z jednotky do procesoru byl použit program *Inertial Center 1.0* z internetových stránek projektů v CircleOS¹⁾, který se skládá z hlavního programu pro postupné adresování senzorů a vyčítání dat z nich, dále pak z knihovny pro komunikaci po rozhraní I2C (*STM32F10x standard peripheral library v3.4.0*²⁾). Nicméně tento projekt byl napsán pro příbuzný kit *EvoPrimer STM32E*, bylo tedy nutné zdrojový kód upravit tak, aby fungoval na použitém kitu *Primer2*. To zahrnovalo především přemapování pinů, změnu souřadnic a velikostí objektů vykreslovaných na displej – kity mají totiž různé rozlišení displejů – a modifikaci inicializace I2C komunikace. Provedl jsem tedy portaci demo projektu pro *Primer2*, a poté jsem ji publikoval na webové stránce projektů pro CircleOS³⁾, aby byla dostupná i dalším vývojářům.

Mým záměrem bylo nastavit u senzorů tutéž vzorkovací frekvenci a poté je všechny naráz spustit. Bohužel, ačkoli jsou všechny senzory od jednoho výrobce, nepodporují synchronizované spuštění odměru. Dokonce neumožňují ani spuštění měření. Lze pouze nastavit vzorkovací frekvenci. To znamená, že pokud se senzoru dotazujeme na naměřenou hodnotu se stejnou frekvencí, jako je jeho vzorkovací frekvence, víme, že čtená data byla naměřena někdy během předchozí vzorkovací periody. Přesný čas ale není znám. Nicméně vzhledem k tomu, že se jednotka bude používat pro monitoring pohybu lidského těla, kde se nepočítá s nijak rychlými ději, lze chybu způsobenou touto nedokonalostí zanedbat. Následující tabulka 3.1 shrnuje výstupní frekvenci dat (Output Data Rate – ODR) jednotlivých senzorů podle údajů z katalogových listů.

¹⁾ <http://www.stm32circle.com/projects/project.php?id=219>

²⁾ <http://www.st.com/web/en/catalog/tools/PF257890>

³⁾ <http://www.stm32circle.com/projects/project.php?id=257>

L3G4200D gyroskop	LSM303DLM akcelerometr	LSM303DLM magnetometr
		0,75 Hz
		1,5 Hz
		3,0 Hz
		7,5 Hz
		15 Hz
		30 Hz
	50 Hz	
		75 Hz
100 Hz	100 Hz	
200 Hz		
		220 Hz
400 Hz	400 Hz	
800 Hz		
	1000 Hz	

Tabulka 3.1. Output Data Rate jednotlivých senzorů

Z tabulky je patrné, že bohužel neexistuje žádná společná frekvence ani žádný celočíselný násobek frekvence, který by odpovídal všem senzorům. Tudíž jsem zvolil u gyroskopu frekvenci 100 Hz, u akcelerometru 100 Hz a u magnetometru 220 Hz, což není zcela násobek předchozích frekvencí, ale domnívám se, že toto přiblížení je pro potřeby snímání lidského pohybu, jak je popsáno výše, postačující.

Tedy procesor se bude s opakovací frekvencí 100 Hz postupně dotazovat senzorů na naměřené hodnoty.

Dotazování je provedeno v obsluze přerušení generované vnitřním časovačem procesoru. Při inicializaci programu je tedy nutné nastavit a spustit časovač, který přeteče každých 10 ms a vygeneruje tak přerušení. Byl zvolen časovač *Timer 3* a nastaven následovně:

```
void init_TIM() {

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    TIM_DeInit(TIM3);

    /* (1/72MHz)*12*60000 = 10 ms */
    TIM_TimeBaseInitTypeDef timerInitStructure;
    timerInitStructure.TIM_Prescaler = 12-1;
    timerInitStructure.TIM_CounterMode = TIM_CounterMode_Up;
    timerInitStructure.TIM_Period = 60000;
    timerInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    timerInitStructure.TIM_RepetitionCounter = 0;
    TIM_TimeBaseInit(TIM3, &timerInitStructure);
    TIM_Cmd(TIM3, ENABLE);
    TIM_ClearFlag( TIM3, TIM_FLAG_Update );

    /* Nastavení přerušení při přetečení */
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
}
}
```

3.2 Ukládání dat na paměťovou kartu

Hlavní smyčka programu slouží především k vykreslování grafiky na displej a ukládání dat na paměťovou kartu. Jedná se o poměrně časově náročné procesy, které ale není potřeba spouštět tak často, jak odpovídá vzorkovací frekvenci měření 100 Hz. Proto je hlavní smyčka spouštěna frekvencí o řád nižší, průměrně jednou za 180 ms bez zápisu nebo 230 ms se zápisem na paměťovou kartu. To však znamená, že mezi jednotlivými průchody hlavní smyčkou je přerušeno spuštěno několikrát, a je tak potřeba naměřené hodnoty uchovávat až do jejich zpracování. K tomu se využívají dva *buffery* (vyrovnávací paměti) implementované jako dostatečně dlouhá pole struktur s naměřenými hodnotami.

Princip kontinuity měření je následující. Jeden *buffer* se vždy plní v obsluze přerušeni až do okamžiku přepisu jeho hodnot na paměťovou kartu. V tento moment jsou *buffery* prohozeny, aby nedošlo k přerušeni měření.

K samotnému zápisu na microSD kartu jsem napsal modul využívající knihovnicí funkcí CircleOS pro obsluhu karty. Tento modul poskytuje následující funkce:

- **void initSD():** Funkce, která připojí kartu, otevře svazek na prvním oddílu, kořenový adresář a vytvoří v něm nový soubor.
- **void openSD():** Funkce, která otevře nově vytvořený soubor.
- **void ulozStrukturu(INERTIAL_DATA* data):** Postupně prochází strukturu *data* s hodnotami získanými od všech senzorů ve stejný okamžik, vytváří v souboru nový řádek (kombinace ASCII znaků CR–0x0D a LF–0x0A) a zapisuje do něj tyto hodnoty oddělené čárkou (0x2C). Takto vzniklý soubor je tudíž ve formátu *CSV (Comma-separated values)*, který je určený pro výměnu tabulkových dat a je široce podporován nejen tabulkovými editory.
- **void closeSD():** Korektně uzavírá otevřený soubor.

3.3 Virtual COM Port

Další částí úkolu bylo posílání naměřených dat z *Primeru* do PC. Využil jsme toho, že *Primer* disponuje jak USB vstupem pro programování a debugging, tak i uživatelským USB portem, který je k dispozici pro celou škálu aplikací. V tomto konkrétním případě byl využit pro přenos dat do PC protokol *Virtual COM Port*, neboli přenosu dat po USB sběrnici, který se tváří jako sériová linka, tedy virtuální sériový port. Nicméně data jsou při přenosu zabalena do USB paketů, to však koncový uživatel nevidí. Pro něho se zařízení jeví jako připojené na sériové lince. Data pak lze vyčítat v libovolném terminálu nebo si napsat vlastní program s využitím patřičných komunikačních knihoven.

Protože komunikace po USB sběrnici není triviální a její kompletní naprogramování by bylo velmi časově náročné, přistoupil jsem k použití knihovny pro USB komunikaci vyvinuté firmou STMicroelectronics (*STM32F10x*, *STM32L1xx* and *STM32F3xx USB full speed device library v4.0.0¹*). Tato knihovna je nakonfigurována tak, aby se procesor jevil jako zařízení standardní třídy *USB communication devices class*. To je především dáno obsahem jednotlivých USB deskriptorů definovaných v jednom ze zdrojových souborů knihovny. Úkolem pak je na začátku programu tuto knihovnu inicializovat, tj. určit zdroj hodinového signálu pro komunikaci, nastavit patřičná přerušeni a zapnout komunikaci. O veškerou obsluhu komunikace (připojení/odpojení, správu napájení, přerušeni, posílání dat, ...) se pak starají knihovní ovladače. Protože se jedná o Virtual

¹) <http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1743/PF258157>

COM Port, knihovna definuje TX endpoint, pomocí něhož můžeme odeslat námi daný řetězec dat připravený v *packet-memory-area* do PC. Posledním krokem je předání řetězce, který je potřeba poslat, pomocí funkce **void Send_by_VirtualCOM**:

```
void Send_by_VirtualCOM( u8* string)
{
  /* Definuje počet poslaných bytů. */
  #define ToSend 64
  /* Pole 64 ASCII znaků */
  static u8 ascii[ToSend];

  /* Vynulování pole */
  memset(ascii, 0, sizeof(u8)*ToSend);

  /* Překopírování vstupního řetězce */
  sprintf(ascii, "%s", string);

  /* Pošle ascii[] do usb packet-memory-area. */
  UserToPMABufferCopy(ascii, ENDP1_TXADDR, ToSend);
  /* Předá informaci o počtu bytů. */
  SetEPTxCount(ENDP1, ToSend);
  /* Povolí USB Tx endpoint pro přenos. */
  SetEPTxValid(ENDP1);
}
```

Tato funkce je pak volána v každém „měřícím“ přerušení po přečtení dat, aby byla data posílána do PC pravidelně. Celá **obsluha přerušení** je popsána v následujícím úryvku ze zdrojového kódu:

```
void TIM3_IRQHandler()
{
  if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
  {
    /* Vynulování příznaku přerušení */
    TIM_ClearITPendingBit(TIM3, TIM_IT_Update);

    /* Sběr dat */
    int i;
    static u8 ascii[64];

    TIM_Cmd( TIM2, DISABLE );
    /* Vypnutí FSMC, protože sdílí piny s I2C */
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, DISABLE);
    for(i=0; i<1000; i++);

    /* Inicializace I2C komunikace */
    GYRO_Init();

    /* Samotný sběr dat */
    DataAcquisition(0xFF , &AcqData);

    /* Obnovení původního nastavení */
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, ENABLE);
    TIM_Cmd(TIM2, ENABLE);
  }
}
```

```
/* Uložení záznamu do bufferu */
if(index_buffer < MAX_BUFFER_LENGTH){
    if(which_buffer == 0) {
        data_buffer0[index_buffer++] = AcqData;
    } else {
        data_buffer1[index_buffer++] = AcqData;
    }
}

/* Vynulování řetězce */
memset(ascii, 0, sizeof(u8)*64);
/* Naplnění řetězce hodnotami ze struktury */
sprintf(ascii, "%d,%d,%d,%d,%d,%d,%d,%d,%d\r",
        AcqData.sAcc[0], AcqData.sAcc[1], AcqData.sAcc[2],
        AcqData.sMag[0], AcqData.sMag[1], AcqData.sMag[2],
        AcqData.sGyro[0], AcqData.sGyro[1], AcqData.sGyro[2]);
/* Odeslání řetězce přes VCP do PC */
Send_by_VirtualCOM(ascii);
}
}
```

Kapitola 4

Program pro Primer2

Firmware pro sběr, ukládání a posílání dat vyvinutý pro *Primer2* se skládá ze tří základních částí: z inicializace, samotné programové smyčky a obsluhy přerušení.

Ačkoli je program velmi intuitivní, příloha A je věnována instrukcím k použití.

4.1 Inicializace

O inicializaci celé aplikace se stará metoda *Application_Init()*. Ta je spuštěna operačním systémem *CircleOS* jednou, po spuštění aplikace. V rámci ní se inicializuje USB Virtual COM Port. Dále se nastaví registry senzorů, tj. přes I2C se nakonfigurují tak, aby senzory měřily s požadovanou frekvencí, posílaly data v požadovaném formátu atd. Poté jsou načteny z paměťové karty kalibrační konstanty. A nakonec je nastaven časovač *Timer3* s příslušným přerušením.

4.2 Hlavní programová smyčka

Hlavní smyčka je reprezentována funkcí *Application_handler()*, která se opakovaně, každých cca 200 ms volá operačním systémem.

Pokud je tlačítkem povoleno ukládání dat na SD kartu, pak se aktuální naplněný *buffer* uloží na kartu pomocí metod uvedených v sekci 3.2. Mezitím se už nově naměřené hodnoty ukládají do *bufferu* číslo dva. Načež je první *buffer* vymazán a na display jsou vykreslena aktuální data. Pokud je zmáčknuto tlačítko, aplikace se ukončí.

4.3 Obsluha přerušení

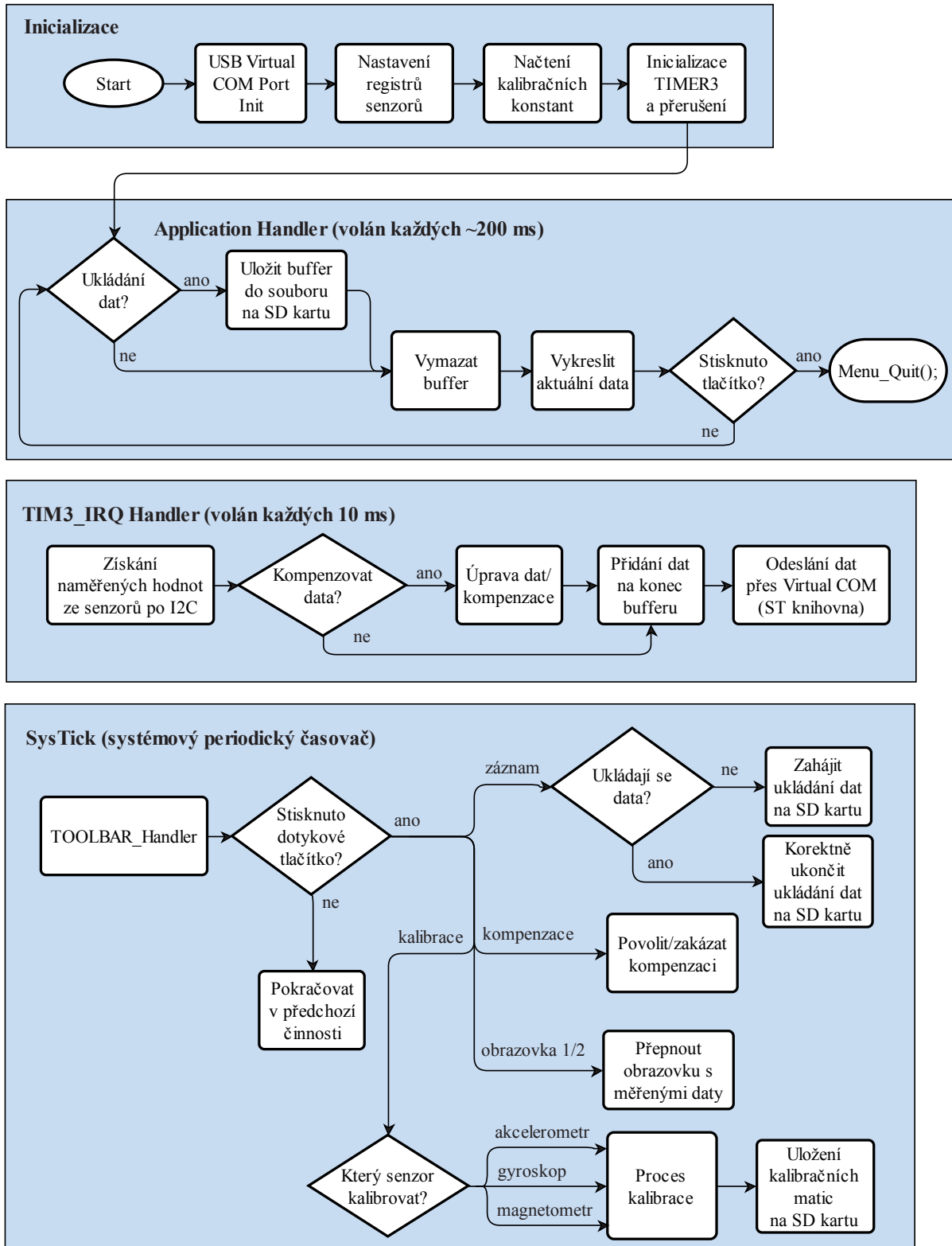
Obsluha přerušení je metoda, která je volána vždy po přetečení čítajícího *Timeru3*, tj. každých 10 ms. V této metodě jsou nejprve získána data ze senzorů po I2C sběrnici a uložena do jedné struktury. Dále je provedena volitelná kompenzace dat (viz dále). Následně se struktura uloží na konec *bufferu*. Nakonec se data ze struktury zformátují do řetězce a odešlou přes Virtual COM Port do počítače.

4.4 Kalibrace senzorů

Po zvolení kalibrace senzoru je vypnuto posílání dat po USB a uživatel je instruován v jednotlivých krocích pokyny na obrazovce. Pomocí naměřených dat jsou spočteny kalibrační matice, které jsou následně uloženy na SD kartu a program se vrací zpět do výchozího režimu nepřetržitého měření.

4.5 Flow diagram

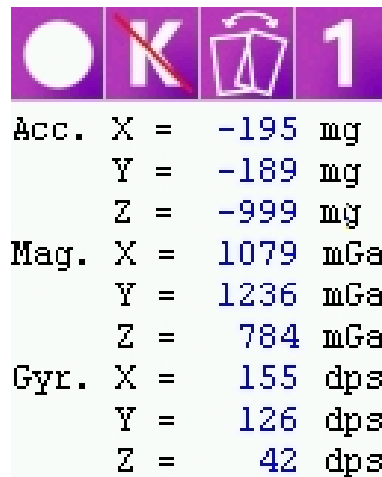
Princip programu *Inertial Center 1.0.1* je pro shrnutí ve stručnosti znázorněn na flow diagramu 4.1.



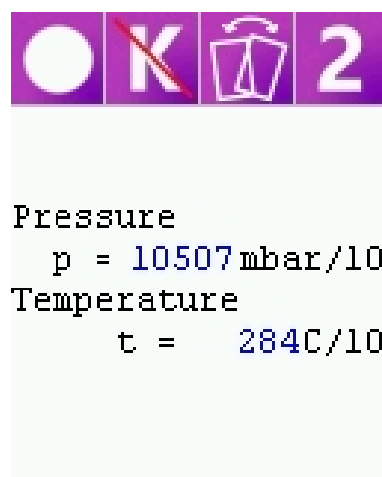
Obrázek 4.1. Diagram průběhu programu Inertial Center 1.0.1.

4.6 Snímky obrazovky programu

Obrázky 4.2, 4.3, 4.4 ukazují pro úplnost vybrané snímky obrazovky programu *Inertial Center 1.0.1*.



Obrázek 4.2. Snímek obrazovky – hlavní obrazovka č. 1 s aktuálními hodnotami senzorů.



Obrázek 4.3. Snímek obrazovky – hlavní obrazovka č. 2 s aktuálními hodnotami senzorů.



Obrázek 4.4. Snímek obrazovky – výběr senzoru pro kalibraci.

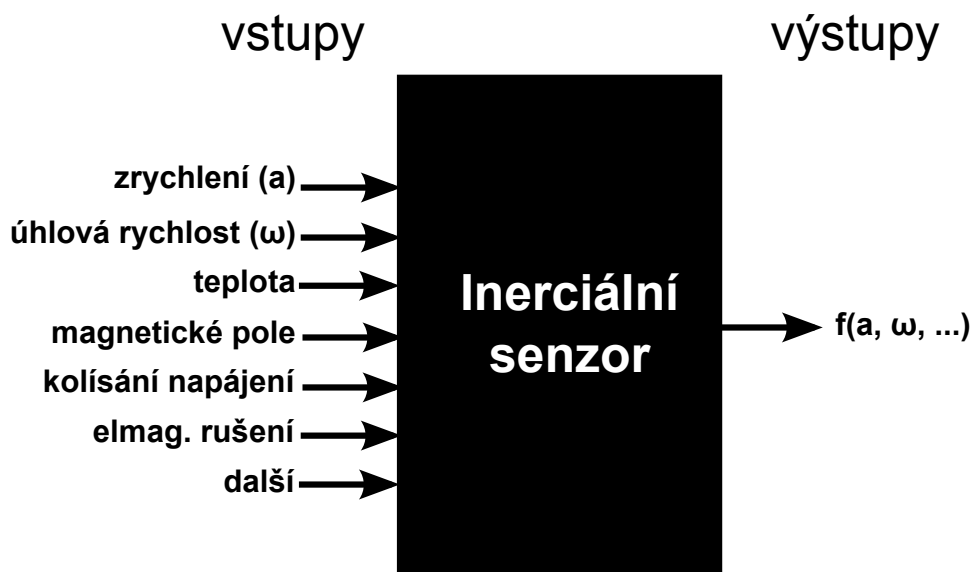
Kapitola 5

Kalibrace IMU jednotky

5.1 Chybový model jednotky

Inerciální navigace odvozuje vnější stavy z vnitřních pozorování. Proto se jí také říká „black-box“ navigace. Problémem ale může být to, že na senzory působí i jiné než měřené veličiny, a tak zkreslí vnitřní stavy senzoru. Jedním z hlavních úkolů inerciální navigace je tedy umět odvodit požadované vstupy senzorů z jejich známých výstupů pomocí návrhu vhodného modelu.

Návrh a výroba senzorů podléhá určitým kritériím na přesnost měření. Nicméně, tyto požadavky jsou většinou v mezích výrobních tolerancí nedosažitelné. Naštěstí je lze často v dostatečné míře splnit pomocí procesu kalibrace a kompenzace.



Obrázek 5.1. Chybový model IMU jednotky (podle [7]).

V chybových modelech nacházíme především dvě základní skupiny chyb:

- Náhodné chyby
- Systematické chyby

■ 5.1.1 Náhodné chyby

Tato skupina chyb se často používá v modelech s Kalmanovým filtrem (viz [7]) a slouží především k modelování nepředvídatelných výsledků měření. V této skupině nacházíme několik druhů chyb:

- bílý šum
- exponenciálně korelovaný šum
- chyba „náhodné procházky“ – v nejjednodušším případě se jedná o proces, jehož aktuální hodnota chyby je součtem chyby předchozího kroku a bílého šumu.
- harmonický šum
- další

■ 5.1.2 Systematické chyby

Do této skupiny náleží chyby výstupů senzorů, které jsou, oproti náhodným chybám, opakovatelné. Nejčastější chyby jsou například:

- offset (bias): nenulový výstup senzoru při nulovém vstupu,
- chyba měřítka: jednotka na výstupu není shodná s referenční jednotkou,
- nelinearita: ve větší či menší míře zastoupena u všech senzorů,
- pásmo necitlivosti,
- kvantizační chyba: přítomna ve všech digitálních systémech,
- další

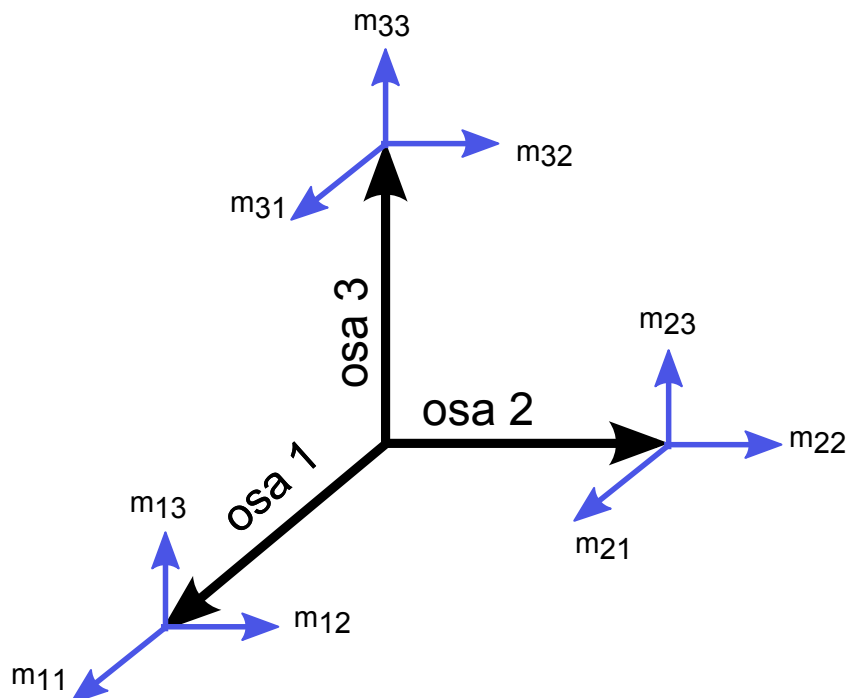
Vstupy senzoru je pak možné získat z naměřených výstupů jen v tom případě, že vztah vstup–výstup je známý a reverzibilní. Z výše vyjmenovaných to dobře není možné u pásma necitlivosti a kvantizační chyby.

■ 5.2 Kalibrace a kompenzace senzorů

Kompenzace senzoru je proces získávání vstupů senzoru z jeho skutečně naměřených výstupů. Oproti tomu **kalibrace** senzoru je proces určení parametrů kompenzačního modelu senzoru. Většinou probíhá způsobem, kdy se porovnávají výstupy systému s referenčními údaji a zjišťují se koeficienty systému tak, aby měřené výstupy odpovídaly referenčním hodnotám.

■ 5.2.1 Parametry kompenzačního modelu

Mezi nejčastější chyby, které se zanáší do tohoto modelu jsou: offset, chyba měřítka a neortogonalita os. Obrázek 5.2 znázorňuje, jak neortogonalita os a chyba měřítka ovlivňují výstup senzoru.



Obrázek 5.2. Znázornění neortogonalit os a chyby měřítka (podle [7]).

Pokud bereme v úvahu tyto tři druhy chyb, pak lze vztah mezi vstupem (skutečným stavem) a výstupem (naměřenými hodnotami) senzoru popsat následujícím maticovým vztahem [7]:

$$\mathbf{z}_{\text{out}} = \mathbf{M}(\mathbf{z}_{\text{in}} + \mathbf{b}_z) \quad (1)$$

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix},$$

kde \mathbf{z}_{in} je vektor reprezentující vstupy (skutečnou hodnotu veličiny), \mathbf{z}_{out} je vektor korespondujících výstupů (naměřených hodnot), \mathbf{b}_z představuje vektor výstupních offsetů a příslušné prvky matice \mathbf{M} jsou vyznačeny v obrázku 5.2.

Parametry m_{ij} a \mathbf{b}_z tohoto modelu mohou být odhadnuty z pozorování výstupů senzoru při známých vstupech, tedy při **kalibraci**.

Účelem kalibrace je **kompensace** senzoru, které dosáhneme inverzí vztahu (1):

$$\mathbf{z}_{\text{in}} = \mathbf{M}^{-1}\mathbf{z}_{\text{out}} - \mathbf{b}_z$$

Získáme tak vstupy senzoru kompenzované o chybu měřítka, neortogonalitu os a offset.

■ 5.2.2 Výpočet parametrů kompenzačního modelu

Rovnici (1) si upravíme do vhodného tvaru [8]:

$$\mathbf{z}_{\text{out}} = \mathbf{M}(\mathbf{z}_{\text{in}} + \mathbf{b}_z) = \mathbf{M}\mathbf{z}_{\text{in}} + \mathbf{M}\mathbf{b}_z = \mathbf{M}\mathbf{z}_{\text{in}} + \mathbf{c},$$

kde

$$\mathbf{c} = \mathbf{M}\mathbf{b}_z = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} b_{z1} \\ b_{z2} \\ b_{z3} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Dále upravujeme:

$$\begin{bmatrix} z_{x,out} \\ z_{y,out} \\ z_{z,out} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} z_{x,in} \\ z_{y,in} \\ z_{z,in} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & c_1 \\ m_{21} & m_{22} & m_{23} & c_2 \\ m_{31} & m_{32} & m_{33} & c_3 \end{bmatrix} \cdot \begin{bmatrix} z_{x,in} \\ z_{y,in} \\ z_{z,in} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} z_{x,out} & z_{y,out} & z_{z,out} \end{bmatrix} = \begin{bmatrix} z_{x,in} & z_{y,in} & z_{z,in} & 1 \end{bmatrix} \cdot \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \\ c_1 & c_2 & c_3 \end{bmatrix}$$

neboli

$$\mathbf{Y} = \mathbf{w} \cdot \mathbf{M},$$

kde

- \mathbf{Y} je vektor surových dat ze senzoru v referenční pozici.
- \mathbf{w} je vektor reference.
- \mathbf{M} je matice 12 parametrů, které hledáme.

Nyní postupujeme tak, že naměříme hodnoty \mathbf{Y}_i v n pozicích s referencí \mathbf{w}_i :

$$\mathbf{Y}_1 = [z_{x,out1} \quad z_{y,out1} \quad z_{z,out1}], \quad \mathbf{w}_1 = [z_{x,in1} \quad z_{y,in1} \quad z_{z,in1} \quad 1]$$

$$\mathbf{Y}_2 = [z_{x,out2} \quad z_{y,out2} \quad z_{z,out2}], \quad \mathbf{w}_2 = [z_{x,in2} \quad z_{y,in2} \quad z_{z,in2} \quad 1]$$

...

$$\mathbf{Y}_n = [z_{x,outn} \quad z_{y,outn} \quad z_{z,outn}], \quad \mathbf{w}_n = [z_{x,inn} \quad z_{y,inn} \quad z_{z,inn} \quad 1].$$

Naměřené vektory zkombinujeme do matic:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \dots \\ \mathbf{Y}_n \end{bmatrix}_{n \times 3}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \dots \\ \mathbf{w}_n \end{bmatrix}_{n \times 4}$$

a koeficienty matice \mathbf{M} spočítáme pomocí *metody nejmenších čtverců* podle vztahu:

$$\mathbf{M} = [\mathbf{w}^T \mathbf{w}]^{-1} \mathbf{w}^T \cdot \mathbf{Y}$$

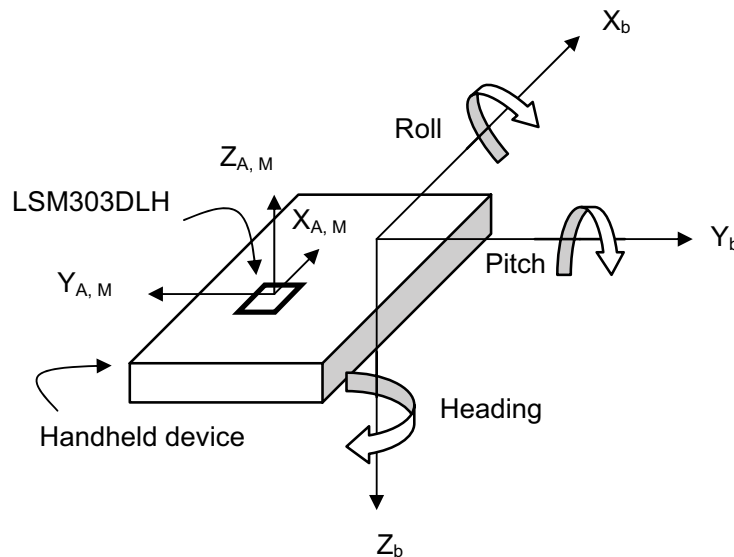
5.3 Kalibrace akcelerometru

Pro kalibraci a následnou kompenzaci akcelerometru byly výše uvedené kompenzační vztahy mírně pozměněny. Ty jsou sice názorné, nicméně následující model je výhodnější pro následnou kompenzaci (podle [8]):

$$\begin{aligned} \begin{bmatrix} A_{x1} \\ A_{y1} \\ A_{z1} \end{bmatrix} &= [A_m]_{3 \times 3} \begin{bmatrix} 1/A_{SC_x} & 0 & 0 \\ 0 & 1/A_{SC_y} & 0 \\ 0 & 0 & 1/A_{SC_z} \end{bmatrix} \begin{bmatrix} A_x - A_{OS_x} \\ A_y - A_{OS_y} \\ A_z - A_{OS_z} \end{bmatrix} = \\ &= \begin{bmatrix} ACC_{11} & ACC_{12} & ACC_{13} \\ ACC_{21} & ACC_{22} & ACC_{23} \\ ACC_{31} & ACC_{32} & ACC_{33} \end{bmatrix} \cdot \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \begin{bmatrix} ACC_{10} \\ ACC_{20} \\ ACC_{30} \end{bmatrix} \end{aligned}$$

Kde $[A_m]$ je matice neortogonality os mezi měřicími osami akcelerometru a osami těla kytu, A_{SC_i} je škálovací faktor a A_{OS_i} představuje offset. Vzorec tedy vyjadřuje vztah mezi surovými výstupními (skutečně změřenými) daty A_x , A_y a A_z a kompenzovanými vstupy senzoru A_{x1} , A_{y1} a A_{z1} . Cílem kalibrace je pak najít všech 12 koeficientů ACC_{10} až ACC_{33} .

Tyto koeficienty jsme našli pomocí *metody nejmenších čtverců* uvedené v sekci 5.2.2. Nejprve je však nutné si nadefinovat znaménkovou konvenci souřadnicových os.



Obrázek 5.3. Souřadnicový systém IMU [8].

Na obrázku 5.3 je schématicky znázorněn celý vývojový kit a akcelerometr včetně příslušných souřadnicových os. Ty jsou u akcelerometru dány umístěním součástky – čipu na tištěném spoji. Osy kytu jsou definovány konvencí: osa X_b míří směrem od tlačítka k displeji, osa Y_b míří při čelním pohledu na zařízení doprava a osa Z_b směřuje při vodorovném položení zařízení do podložky. Z obrázku je patrné, že orientace příslušných os kytu a senzoru může být opačná. To však ničemu nevaří, projeví se to jen znaménky u některých kalibračních parametrů.

Pro potřeby kalibrace byly změřeny výstupy ze senzoru v 6 stacionárních pozicích, vždy s jednou osou kolmou na vodorovnou podložku a ostatními osami rovnoběžnými s touto podložkou. Situaci shrnuje tabulka 5.1.

stacionární pozice	A_x	A_y	A_z
Z_b dolů	0	0	+1g
Z_b nahoru	0	0	-1g
Y_b dolů	0	+1g	0
Y_b nahoru	0	-1g	0
X_b dolů	+1g	0	0
X_b nahoru	-1g	0	0

Tabulka 5.1. Znaménková konvence akcelerometru

Pro každou ze 6 poloh bylo do kalibrace zahrnuto 5 údajů. Každý údaj vzniknul průměrem ze 100 měření. Samotný sběr dat a numerické výpočty byly provedeny ve vlastním programu napsaném v jazyce ANSI C a vyvinutém ve vývojovém prostředí *National Instruments LabWindows/CVI*. A to především z důvodů použití komfortních a spolehlivých knihoven pro sériovou komunikaci a práci s maticemi. Program lze najít na příloženém CD a jeho návod v příloze B.

Výsledná matice kalibračních parametrů vypočtená metodou nejmenších čtverců vyšla pro použitou IMU jednotku následovně:

$$\mathbf{x} = \begin{bmatrix} ACC_{11} & ACC_{21} & ACC_{31} \\ ACC_{12} & ACC_{22} & ACC_{33} \\ ACC_{13} & ACC_{23} & ACC_{33} \\ ACC_{10} & ACC_{20} & ACC_{30} \end{bmatrix} = \begin{bmatrix} 0.9710 & -0.0007 & -0.0236 \\ -0.0060 & 1.0013 & 0.0009 \\ 0.0009 & -0.0026 & -0.9973 \\ -26.7632 & 35.3905 & -44.2094 \end{bmatrix}$$

Tabulka 5.2 pak shrnuje výsledky kalibrace, tj. naměřená a kompenzovaná data ze senzoru. V každé ze šesti poloh bylo učiněno tři sta odměrů, z nich poté vypočítán aritmetický průměr a hodnota RMSE. Ta se spočítá podle vzorce (2).

$$RMSE = \sqrt{\frac{\sum_{k=1}^n (\hat{y}_k - y_k)^2}{n}}, \quad (2)$$

kde n je počet odměrů, y_k je k -tý odměr a \hat{y}_k je očekávaná (referenční) hodnota k -tého odměru.

RMSE dává jistý náhled na to, jak moc se naměřené hodnoty liší od hodnoty referenční. Prakticky tedy platí, že čím menší RMSE, tím lepší kalibrace.

#	reference [mg]	nekomp. senzor [mg]	RMSE [mg]	komp. senzor [mg]	RMSE [mg]
x	1000	1059,1	60,0	1001,7	10,4
1 y	0	-32,2	33,5	2,5	9,4
z	0	-69,1	70,1	-0,3	11,8
x	-1000	-1000,3	10,8	-997,9	10,7
2 y	0	-34,0	35,2	2,1	9,7
z	0	-20,3	23,4	-0,3	11,5
x	0	35,8	37,4	2,3	10,6
3 y	1000	963,5	37,7	1000,3	9,2
z	0	-42,6	44,1	-1,7	11,5
x	0	24,0	26,3	2,7	10,7
4 y	-1000	-1033,2	34,6	-999,1	9,8
z	0	-43,9	45,3	-1,9	11,2
x	0	23,7	25,9	-4,4	11,0
5 y	0	-40,7	41,9	-2,7	10,1
z	1000	-1049,4	2049,5	1001,8	11,3
x	0	22,6	24,8	-3,8	10,9
6 y	0	-35,2	36,4	-2,3	9,5
z	-1000	955,5	1955,5	-997,7	11,5

Tabulka 5.2. Výsledky kalibrace akcelerometru podle prvního modelu (1 mg = 0,001g)

Z tabulky 5.2 je patrné, že ačkoli senzor byl už poměrně dobře nakalibrován z výroby, výše uvedená metoda kalibrace umožnila měření ještě zpřesnit. Dále si povšimneme opačné orientace *osy z* oproti zvolené konvenci, která se projevila především opačným znaménkem oproti referenci a obrovským RSME v 5. a 6. poloze. Tato maličkost se odstanila taktéž kompenzací.

Jedná se o poměrně nenáročnou metodu, běžný uživatel by ji tedy měl být schopen bez problémů provést v domácím prostředí. Vyžaduje pouze umístění jednotky do šesti předem definovaných poloh a ačkoli při kalibraci nebyly použity žádné přesné reference, poskytuje poměrně dobré výsledky.

Pro srovnání byl akcelerometr také zkalibrován pomocí složitějších postupů v *Laboratoři leteckých a kosmických systémů* na *FEL ČVUT*. Měření probíhalo tak, že jednotka se uchytila na kalibrační plošinu a vždy se zafixovala jedna osa. Jednotkou se pak kolem ní otáčelo s krokem přibližně $22,5^\circ$. V každé poloze se naměřil zhruba 15sekundový záznam. Naměřená data byla následně zpracována algoritmem popsáným v [9]. Jeho základní myšlenkou je, že se neporovnávají složky vektoru zrychlení, jak je tomu výše, ale využívá se toho, že pokud je jednotka v naprostém klidu, působí na ni zrychlení o velikosti vektoru tíhového zrychlení g . Bere se tedy v úvahu pouze modul naměřeného zrychlení. Pro tuto metodu tudíž není potřeba znát přesnou polohu jednotky. Na druhou stranu je potřeba naměřit data ve větším počtu poloh. Výsledný kompenzační model je následující:

$$\begin{aligned}
a_p &= T_a^p S F_a (a_m - b_a) = \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0,0045 & 1 & 0 \\ -0,0031 & -0,0027 & 1 \end{bmatrix} \begin{bmatrix} 0,9711 & 0 & 0 \\ 0 & 1,0018 & 0 \\ 0 & 0 & -0,9974 \end{bmatrix} \times \\
&\quad \times \left(\begin{bmatrix} a_{mx} \\ a_{my} \\ a_{mz} \end{bmatrix} - \begin{bmatrix} 24,9033 \\ -38,1473 \\ -44,4979 \end{bmatrix} \right),
\end{aligned}$$

kde a_p je kompenzovaný vektor změřeného zrychlení, T_a^p značí matici neortogonální, $S F_a$ je škálovací matice, b_a je vektor offsetů a a_m představuje vektor změřeného zrychlení.

Pro srovnání jsou v tabulce 5.3 shrnuty výsledky kompenzace stejných dat jako v tabulce 5.2, nyní však podle druhého modelu.

#	reference [mg]	nekomp. senzor [mg]	RMSE [mg]	komp. senzor [mg]	RMSE [mg]	
1	x	1000	1059,1	60,0	1004,3	11,1
	y	0	-32,2	33,5	10,5	13,8
	z	0	-69,1	70,1	21,4	24,5
2	x	-1000	-1000,3	10,8	-995,6	11,3
	y	0	-34,0	35,2	-0,3	9,4
	z	0	-20,3	23,4	-21,0	24,0
3	x	0	35,8	37,4	10,6	14,9
	y	1000	963,5	37,7	1003,5	10,0
	z	0	-42,6	44,1	-4,6	12,3
4	x	0	24,0	26,3	-0,9	10,4
	y	-1000	-1033,2	34,6	-996,8	10,3
	z	0	-43,9	45,3	2,1	11,2
5	x	0	23,7	25,9	-1,2	10,2
	y	0	-40,7	41,9	-2,6	10,1
	z	1000	-1049,4	2049,5	1002,3	11,4
6	x	0	22,6	24,8	-2,3	10,2
	y	0	-35,2	36,4	2,9	9,7
	z	-1000	955,5	1955,5	-997,4	11,6

Tabulka 5.3. Výsledky kalibrace akcelerometru podle druhého modelu (1 mg = 0,001g)

Je patrné, že výsledky kompenzace podle obou modelů jsou srovnatelné. Mimo to, prvně zmíněný přístup lze snadno aplikovat i v domácím prostředí, proto byl dále použit pro kalibraci IMU jednotky.

5.4 Kalibrace magnetometru

Při kalibraci magnetometru postupujeme podobně jako u akcelerometru, avšak do algoritmu bylo přidáno několik úprav. Pozměněný model je následující [8]:

$$\begin{bmatrix} M_{x1} \\ M_{y1} \\ M_{z1} \end{bmatrix} = [M_m]_{3 \times 3} \begin{bmatrix} 1/M_SC_x & 0 & 0 \\ 0 & 1/M_SC_y & 0 \\ 0 & 0 & 1/M_SC_z \end{bmatrix} \cdot [M_si]_{3 \times 3} \begin{bmatrix} M_x - M_OS_x \\ M_y - M_OS_y \\ M_z - M_OS_z \end{bmatrix} = \begin{bmatrix} MR_{11} & MR_{12} & MR_{13} \\ MR_{21} & MR_{22} & MR_{23} \\ MR_{31} & MR_{32} & MR_{33} \end{bmatrix} \cdot \begin{bmatrix} M_x - MR_{10} \\ M_y - MR_{20} \\ M_z - MR_{30} \end{bmatrix},$$

kde M_{x1} , M_{y1} a M_{z1} jsou normalizovaná kompenzovaná data magnetického senzoru, M_x , M_y a M_z jsou surová data ze senzoru, $[M_m]$ představuje matici neortogonality os, M_SC_i ($i = x, y, z$) je škálovací faktor a M_OS_i je offset způsobený hard-iron efektem (viz níže); $[M_si]$ je pak matice způsobená soft-iron efektem (též viz níže).

Při měření s magnetometrem se uplatňují dva rušivé efekty:

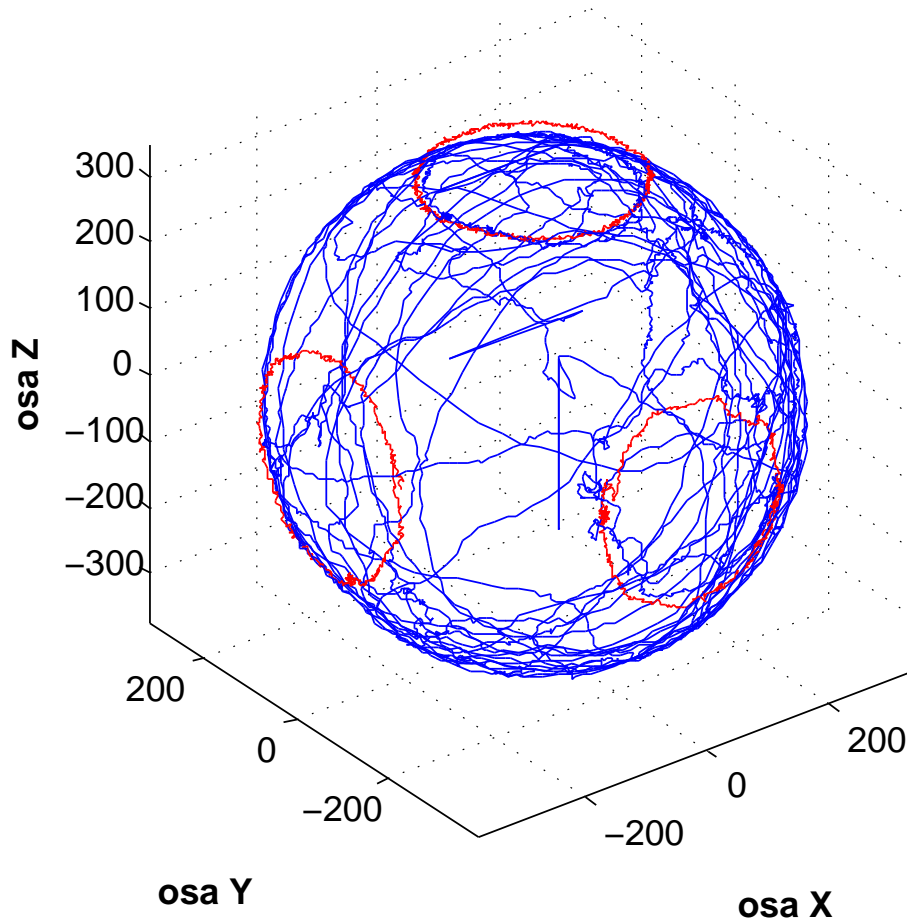
- **Hard-iron efekt:** Je způsoben přítomností permanentních magnetů v okolí senzoru (např. magnet v reproduktoru či mikrofonu). Tento efekt se projevuje jako offset v naměřených datech.
- **Soft-iron efekt:** Způsobují ho magneticky měkké materiály, které se magnetují podle směru magnetického pole na ně působícího, tedy i magnetického pole Země. Tyto materiály tedy způsobují odchylku senzoru závislou na natočení jednotky v magnetickém poli. Pokud bychom naměřili dostatečné množství dat jednotky v různých polohách a zobrazili naměřená data v 3D prostoru (jedním odměrem totiž vždy získáváme trojici čísel), tak se jednotlivé body v ideálním případě zobrazí na povrch kulové plochy o poloměru velikosti působícího magnetického pole, tedy magnetického pole Země. Soft-iron efekt tuto kouli transformuje na elipsoid, který je v případech velmi silného soft-iron efektu navíc zrotovaný.

Zrotovaný elipsoid lze popsat následující rovnicí:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} + \frac{(z - z_0)^2}{c^2} + \frac{(x - x_0)(y - y_0)}{d^2} + \frac{(x - x_0)(z - z_0)}{e^2} + \frac{(y - y_0)(z - z_0)}{f^2} = R^2 \quad (3)$$

- x_0, y_0, z_0 jsou offsety M_OS_i ($i = x, y, z$) způsobené hard-iron efektem,
- x, y, z jsou surová data M_x, M_y a M_z ,
- a, b, c jsou délky poloos,
- d, e, f jsou konstanty natočení elipsy (soft-iron efekt),
- R je velikost síly magnetického pole Země.

Pro přibližné posouzení vlivu těchto dvou efektů byl proveden následující pokus: Jednotkou byla nahrána data z náhodného, poměrně dlouhého pohybu a poté vykreslena v trojrozměrném prostoru v programu MATLAB. Výsledek je znázorněn na obrázku 5.4.



Obrázek 5.4. Náhodné 3D rotace (modře) magnetického senzoru.

Lze si všimnout, že vykreslené křivky leží poměrně přesně na ploše rotačního elipsoidu (který se blíží kulové ploše), jenž není nijak výrazně zrotován. To znamená, že soft-iron efekt se výrazně neuplatňuje, a matice $[M_{si}]$ tak přechází v jednotkovou matici o rozměrech 3×3 . Rovnice (3) pak také přechází do jednoduššího tvaru

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} + \frac{(z - z_0)^2}{c^2} = R^2, \quad (4)$$

který můžeme přepsat do maticového zápisu:

$$x^2 = [x \quad y \quad z \quad -y^2 \quad -z^2 \quad 1] \cdot \begin{bmatrix} 2x_0 \\ \frac{a^2}{b^2} 2y_0 \\ \frac{a^2}{c^2} 2z_0 \\ \frac{a^2}{b^2} \\ \frac{a^2}{c^2} \\ a^2 R^2 - x_0^2 - \frac{a^2}{b^2} y_0^2 - \frac{a^2}{c^2} z_0^2 \end{bmatrix} \quad (5)$$

Pro potřeby kalibrace byla naměřena data z nahodilého pohybu v 3D prostoru. Je výhodné měřit tak, aby hodnoty co nejrovnoměrněji pokrývaly celý povrch elipsoidu a dále je nutné zajistit, aby tato kalibrace probíhala v co magneticky nejčistším prostředí, tj. aby se skutečně zaznamenávalo jen magnetické pole Země. Vznikne tak n trojic čísel

M_x , M_y a M_z , která se uspořádají do matic podle vzoru rovnice (5). Dostaneme tak soustavu rovnic

$$\mathbf{w}_{n \times 1} = \mathbf{H}_{n \times 6} \cdot \mathbf{X}_{6 \times 1},$$

z které se pomocí *metody nejmenších čtverců* dopočítá vektor \mathbf{X} :

$$\mathbf{X} = [\mathbf{H}^T \mathbf{H}]^{-1} \mathbf{H}^T \cdot \mathbf{w}$$

Pak jednotlivé parametry z tohoto vektoru se dopočítají následovně:

$$M_OS_x = x_0 = X(1)/2$$

$$M_OS_y = y_0 = X(2)/(2 \cdot X(4))$$

$$M_OS_z = z_0 = X(3)/(2 \cdot X(5))$$

$$A = a^2 R^2 = X(6) + x_0^2 + X(4) \cdot y_0^2 + X(5) \cdot z_0^2$$

$$B = A/X(4)$$

$$C = A/X(5)$$

Pokud nyní označíme

$$xx = M_x - M_OS_x$$

$$yy = M_y - M_OS_y$$

$$zz = M_z - M_OS_z,$$

pak rovnice (4) přejde do tvaru

$$\frac{xx^2}{A} + \frac{yy^2}{B} + \frac{zz^2}{C} = 1, \quad (6)$$

z kterého je patrné, že pro škálovací faktory platí následující vztahy:

$$M_SC_x = \sqrt{A}$$

$$M_SC_y = \sqrt{B}$$

$$M_SC_z = \sqrt{C}$$

Tímto procesem byly určeny škálovací faktory M_SC_i ($i = x, y, z$), offset způsobený hard-iron efektem M_OS_i a identifikována matice soft-iron efektu $[M_si]$.

Pro použitý kit s IMU jednotkou vyšly následující parametry:

$$M_OS_x = -18,53$$

$$M_OS_y = -35,35$$

$$M_OS_z = -4,19$$

$$M_SC_x = 358,13$$

$$M_SC_y = 367,52$$

$$M_SC_z = 366,15$$

$$[M_si] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Lze si všimnout nenulových hodnot M_{OSi} způsobených pravděpodobně přítomností reproduktoru a mikrofonu v těle kitu. A dále je vidět, že jednotlivé hodnoty M_{SCi} jsou si relativně podobné, tedy na obrázku 5.4 byla data skutečně téměř dokonale vykreslena na kulové ploše.

Nyní ještě zbývá kompenzovat neortogonalitu os. Tato korekce se provedeme pomocí algoritmu z [8]:

Nejprve rovnici (6) přepíšeme pomocí následujících rovností:

$$xxx = xx/M_{SC_x}$$

$$yyy = yy/M_{SC_y}$$

$$zzz = zz/M_{SC_z}$$

$$\Rightarrow xxx^2 + yyy^2 + zzz^2 = 1$$

Následně naměříme plné 2D rotace zařízení kolmo na jednotlivé souřadnicové osy, celkem tedy tři rotace.

Například, provedeme rotaci s osou Z_b směrem dolů, rovnoběžně s rovinou $X_b - Y_b$. Po kompenzaci měřítka a hard-iron efektu se tyto rotace zobrazí jako vystředěné kružnice. Ideálně by měl být kruh kolmý na osu zařízení Z_b a všechny hodnoty zzz_z tohoto naměřeného kruhu by měly být stejné, menší než 1. Pokud tomu tak není, lze najít vektor, který tento kruh posune do správné pozice.

Nechť

$$\mathbf{H}_{m \times 3} = [xxx_z \quad yyy_z \quad zzz_z]$$

$$\mathbf{w}_{m \times 1} = \sqrt{xxx_z^2 + yyy_z^2 + zzz_z^2}$$

jsou data rotace s osou Z_b kolmo dolů po korekci hard-iron, soft-iron efektu a chyby měřítka, pak

$$\mathbf{X}_{3 \times 1} = [\mathbf{H}^T \mathbf{H}]^{-1} \mathbf{H}^T \cdot \mathbf{w}.$$

Takže normalizovaný vektor rotace pro Z_b dolů je:

$$\mathbf{R}_z = \mathbf{X} / \sqrt{X(1)^2 + X(2)^2 + X(3)^2}$$

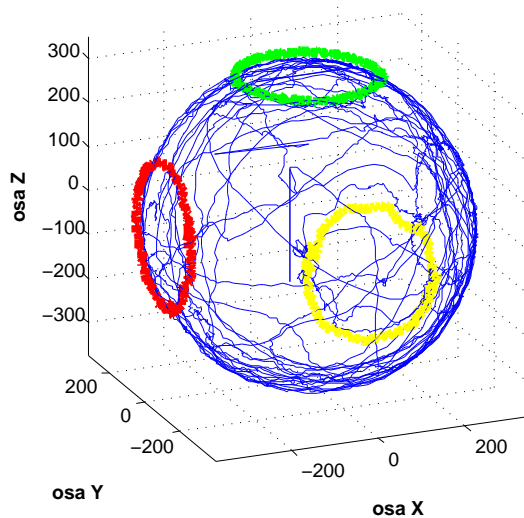
Obdobně byly nalezeny normalizované rotační vektory \mathbf{R}_x pro X_b dolů a \mathbf{R}_y pro Y_b dolů. Nakonec se sestaví výsledná matice pro kompenzaci neortogonalit:

$$\mathbf{M}_{m \times 3 \times 3} = [R_x \quad R_y \quad R_z]$$

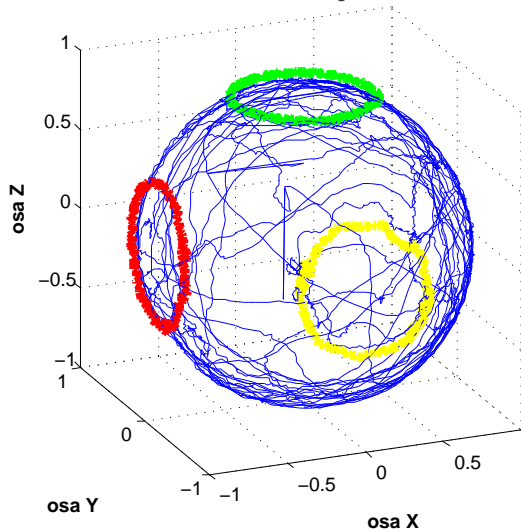
Pro použitou IMU jednotku vychází následující matice:

$$\mathbf{M}_m = \begin{bmatrix} -0.9964 & -0.0123 & -0.0269 \\ -0.0827 & -0.9994 & -0.0978 \\ 0.0202 & -0.0325 & 0.9948 \end{bmatrix}$$

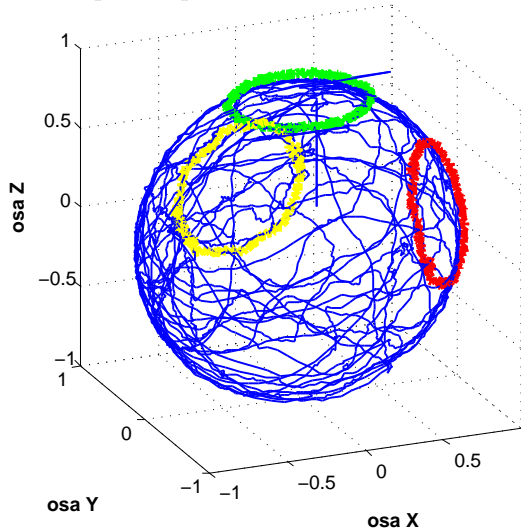
Na následujících obrázcích je pak graficky znázorněn průběh kalibrace akcelerometru:



Obrázek 5.5. Surová data před kalibrací.



Obrázek 5.6. Data po kompenzaci hard-iron efektu a chyby měřítka.



Obrázek 5.7. Data po konečné kompenzaci včetně kompenzace neortogonality os.

5.5 Kalibrace gyroskopu

V první řadě je potřeba určit směry otáčení a jejich znaménkovou konvenci. Ta je znázorněna na obrázku 5.3. Změna úhlu ve směru šipek je s kladným znaménkem, v opačném směru se záporným znaménkem.

Protože jsou data z gyroskopu poměrně hodně zašuměná, je dobré podle [10] nastavit určitý práh, při jehož nepřekročení se bude na výstupu objevovat nulová hodnota. Odstraníme tím klidový šum a nebude se tak načítat úhlový přírůstek, když přístroj bude v klidu, což je žádoucí stav. Rozumnou hranicí se jeví trojnásobek směrodatné odchylky výstupu senzoru v klidu (viz [10]). Tedy

$$\Delta R = R_m - R_0 = 0, \text{ právě když } |R_m - R_0| < R_{th}.$$

Kde R_m je naměřená hodnota, R_{th} je stanovený práh a R_0 je offset určený například jako klouzavý průměr hodnot ze senzoru v klidu.

Nevýhodou tohoto přístupu je, že jsou zcela ignorovány velmi pomalé rotace. Nicméně vzhledem k potenciálnímu použití přístroje, tj. měření pohybů lidského těla v prostoru, si myslím, že toto omezení není nikterak zásadní a výrazně měření nepoškodí.

Výpočet prahových hodnot pro konkrétní případ je shrnut v následující tabulce:

#	R_x	R_y	R_z
1	97	111	26
2	100	121	26
3	117	102	35
4	130	94	30
...
99	86	84	11
100	95	116	26
$R_0 = \overline{R_i}$	114,67	113,95	23,28
σ	17,17	10,06	16,05
$R_{th} = 3\sigma$	51,51	30,18	48,15

Tabulka 5.4. Prahování gyroskopu

Kompenzační model je analogický k modelu gyroskopu [10]:

$$\begin{aligned} \begin{bmatrix} R_{x1} \\ R_{y1} \\ R_{z1} \end{bmatrix} &= [G_m]_{3 \times 3} \begin{bmatrix} SC_x & 0 & 0 \\ 0 & SC_y & 0 \\ 0 & 0 & SC_z \end{bmatrix} \begin{bmatrix} R_x - R_{OS_x} \\ R_y - R_{OS_y} \\ R_z - R_{OS_z} \end{bmatrix} = \\ &= \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \cdot \begin{bmatrix} R_x - R_{OS_x} \\ R_y - R_{OS_y} \\ R_z - R_{OS_z} \end{bmatrix} \end{aligned}$$

- R_{i1} jsou kompenzovaná data senzoru.
- R_i jsou čistá naměřená data senzoru.
- G_m je matice kompenzace neortogonality os.
- SC_i je škálovací faktor.
- R_{OS_i} je offset.
- G_{ij} jsou kalibrační konstanty, které je potřeba zjistit společně s offsety.

Kalibrace gyroskopů je obecně poměrně technicky náročná. Aby ji tedy byl schopen provést i běžný uživatel, byl použit postup podle [11]. Kalibrace tak neprobíhá na

úrovni měření a porovnávání úhlových rychlostí, ale využívá se toho, že integrál z úhlové rychlosti je úhel a tedy při kalibraci porovnáváme úhly.

Kalibrační procedura se skládá z následujících kroků:

1. Zapnout gyroskop a počkat než se teplotně stabilizuje.
2. Začít měřit výstup gyroskopu v první poloze.
3. Počkat 5 s a poté pomalu otočit jednotku do druhé polohy (většinou otočené o $+90^\circ$ od první polohy).
4. Počkat 5 s a poté pomalu otočit jednotku zpět do výchozí polohy.
5. Počkat 5 s a poté zastavit měření. Uložit data. Opakovat kroky 3-5 pro všechny osy.

Poté jsou naměřená data zpracována:

1. Výpočet offsetu průměrováním prvních 3 sekund klidového záznamu. Tím se získají hodnoty $R.OS_i$, ($i = x, y, z$).
2. Offset se odečte od celého záznamu a poté se integrují data příslušející pohybu mezi první a druhou polohou. Škálovací faktor se získá vydělením 90° a integrací vypočteného úhlu.
3. Zintegrují se data pohybu z druhé do první polohy a vydělí se -90° získaným úhlem.
4. Výsledný škálovací faktor se získá průměrem hodnoty získané v 2. a 3. kroku.

Je zřejmé, že touto metodou se vypočítá pouze matice offsetů a škálovacích faktorů. Matici neortogonalit tímto postupem nelze najít, proto ji v zjednodušení položíme rovnu jednotkové matici.

Touto metodou jsem dospěl k následujícím kompenzačním maticím:

$$\mathbf{SF} = \begin{bmatrix} -0,016400 & 0 & 0 \\ 0 & -0,016771 & 0 \\ 0 & 0 & 0,016404 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 112,00 \\ 107,92 \\ 24,13 \end{bmatrix}$$

Následující tabulka 5.5 shrnuje výsledky pokusu s kompenzovaným senzorem. Otočení jednotky o daný úhel probíhalo tak, jak by to zvládnul sám uživatel v domácích podmínkách, čili jako reference sloužil na papíře narysovaný pravoúhlý kříž. Pro každou referenční hodnotu byl pokus proveden desetkrát a následně spočten aritmetický průměr jednotlivých pokusů a RMSE vzhledem k referenční hodnotě.

data set		reference [°]	komp. senzor [°]	RMSE [°]
1	x	90	88,25	3,15
2	x	-90	-89,3	1,77
3	y	90	91,17	1,28
4	y	-90	-90,56	1,02
5	z	90	91,34	2,31
6	z	-90	-88,89	2,25

Tabulka 5.5. Výsledky kalibrace gyroskopu

Tabulka 5.5 ukazuje, že senzor dává po kalibraci poměrně přijatelné výsledky. Více prohlásit nelze, protože jak kalibrace, tak i reference nebyly z principu použitých metod přesné. Domnívám se však, že tato kalibrační metoda je přes svoji jednoduchost a nepříliš velkou přesnost přijatelná, neboť takto zkompenzovaná data budou dále zpracována v Kalmanově filtru, který nepřesnosti ještě vyrovná.

5.6 Implementace kalibrace

Pro potřeby kalibrace IMU jednotky byly jednak implementovány kalibrační postupy v programu pro *Primer*, tak i vznikl program pro PC, jehož jednou z hlavních funkcionalit je právě kalibrace. Oba programy lze dohledat na příloženém CD včetně návodu k použití.

5.6.1 Kalibrace v Primeru

Výše popsané kalibrační postupy byly implementovány do programu *Inertial Center 1.0.1* pro *Primer* včetně podpůrné knihovny pro maticové výpočty (násobení, transpozice, inverze). Nicméně protože právě tyto metody používají maticový počet, ukázal se tento postup pro *Primer* jako numericky nestabilní, takže bylo od něj odstoupeno. Místo toho byly do *Primeru* implementovány matematicky jednodušší postupy, které se zakládají na tom, že se změří v každé ose minimální a maximální hodnota a z nich se poté spočítá offset a chyba měřítka. Tímto se mimo jiné zanedbají matice neortogonalit. Vzhledem k tomu, že při použití přesnějších metod vyšly tyto matice téměř jednotkové, lze tento postup použít aspoň pro orientační zjištění kalibračních matic.

Po zkalibrování senzoru jsou kalibrační matice uloženy na SD kartu, z níž jsou opětovně načteny při každém startu programu.

5.6.2 Kalibrace v PC – Inertial Center for PC

Protože *Primer* nebyl schopen správně numericky spočítat kalibrační matice podle výše uvedených metod, vyvinul jsem ve vývojovém prostředí LabWindows/CVI program *Inertial Center for PC*. Jedna z jeho hlavních součástí je kalibrace IMU jednotky. Nejprve je tedy potřeba zapnout *Primer* a na něm program *Inertial Center 1.0.1* a poté ho připojit pomocí USB k počítači. Aplikace pro PC pak vyčítá přijaté hodnoty a za pomoci uživatele kalibruje jednotku. Používá k tomu algoritmy popsané v předchozích sekcích této kapitoly. Po kalibraci je možné výsledné kalibrační matice uložit do souboru pro budoucí použití. Pokud jsou tyto soubory uloženy na SD kartu *Primeru*, jsou při nejbližším spuštění *Inertial Center 1.0.1* načteny a využity pro kompenzaci dat. Poté se USB Virtual COM portem posílají již kompenzovaná data.

Kapitola 6

Sledování pohybu

Při sledování pohybů lidského těla se zabýváme především natočením a případně i polohou zvolené části těla. V následující kapitole je popsáno a porovnáno několik způsobů, jak toho docílit s IMU jednotkou zkalibrovanou podle předchozí kapitoly.

6.1 Souřadná soustava

V první řadě je nezbytné si nadefinovat souřadné soustavy, ve kterých se bude počítat. Důležité jsou především dvě soustavy:

- Inerciální soustava I (inertial frame) – pevně daná soustava spojená se Zemí. Osa X_i směřuje na sever, osa Y_i na východ a osa Z_i do středu Země.
- Tělesová soustava B (body frame) – soustava pevně spojená s tělem *Primeru*. Je vyznačena na obrázku 5.3 s osami označenými jako X_b , Y_b a Z_b .

6.2 Eulerovy úhly

Pro popis prostorové rotace se používá výhodně rozklad na tři dílčí rotace. Příslušné úhly se nazývají *Eulerovy úhly* (viz [1]).

Nejdříve otočíme soustavu okolo osy Z_i o úhel ψ (yaw, heading, azimut). Tím přejde osa X_i do polohy X_{i2} a osa Y_i do polohy Y_{i2} .

Poté vykonáme rotaci okolo osy Y_{i2} o úhel θ (pitch, podélný sklon), čímž přejde osa X_{i2} do polohy X_b a osa Z_i do polohy Z_{i2} .

Nakonec otočíme soustavu okolo osy X_b o úhel φ (roll, příčný náklon). Tak přejde osa Y_{i2} do polohy Y_b a osa Z_{i2} do polohy Z_b .

Každou z těchto rotací lze popsat rotační maticí. Výsledná rotace se pak popíše maticí vzniklou jako součin rotačních matic jednotlivých dílčích rotací. Důležité je dodržet pořadí rotací v pořadí yaw $\psi \rightarrow$ pitch $\theta \rightarrow$ roll φ . Avšak rotační matice se násobí zleva, takže zde bude zdánlivě pořadí opačné:

$$\begin{aligned} \mathbf{R}_I^B(\psi, \theta, \varphi) &= R_\varphi R_\theta R_\psi = \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & \sin \varphi \\ 0 & -\sin \varphi & \cos \varphi \end{pmatrix} \cdot \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} = \\ &= \begin{pmatrix} \cos \psi \cos \theta & \cos \theta \sin \psi & -\sin \theta \\ \cos \psi \sin \varphi \sin \theta - \cos \varphi \sin \psi & \cos \varphi \cos \psi + \sin \varphi \sin \psi \sin \theta & \cos \theta \sin \varphi \\ \sin \varphi \sin \psi + \cos \varphi \cos \psi \sin \theta & \cos \varphi \sin \psi \sin \theta - \cos \psi \sin \varphi & \cos \varphi \cos \theta \end{pmatrix} \quad (1) \end{aligned}$$

Nevýhodou tohoto přístupu je, že při $\theta = 90^\circ$ nelze jednoznačně určit orientaci pomocí Eulerových úhlů (gimbal lock). Proto se často orientace popisuje pomocí kvaternionů.

6.3 Kvaterniony

Kvaternion je obecně definován jako (podle [12])

$$\bar{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} + q_4$$

kde \mathbf{i} , \mathbf{j} , \mathbf{k} jsou čísla, pro která platí:

$$\mathbf{i}^2 = -1, \mathbf{j}^2 = -1, \mathbf{k}^2 = -1,$$

$$-\mathbf{ij} = \mathbf{ji} = \mathbf{k}, -\mathbf{jk} = \mathbf{kj} = \mathbf{i}, -\mathbf{ki} = \mathbf{ik} = \mathbf{j}.$$

Kvaterniony se také často zapisují ve vektorovém tvaru jako

$$\bar{q} = [q_1 \quad q_2 \quad q_3 \quad q_4]^T$$

Pokud se omezíme na jednotkové kvaterniony $|\bar{q}| = 1$, pak kvaternion

$$\bar{q} = \begin{bmatrix} k_x \sin(\theta/2) \\ k_y \sin(\theta/2) \\ k_z \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix}$$

reprezentuje rotaci kolem osy ve směru $\mathbf{k} = [k_1 \quad k_2 \quad k_3]$ o úhel φ . Pomocí něho lze sestavit rotační matici:

$$\mathbf{R}_I^B(\bar{q}) = \begin{pmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{pmatrix} \quad (2)$$

Z porovnání prvků matic $\mathbf{R}_I^B(\bar{q})$ a $\mathbf{R}_I^B(\psi, \theta, \varphi)$ plynou důležité převodní vztahy:

$$\psi = \arctan \frac{2(q_1q_2 + q_3q_4)}{q_1^2 - q_2^2 - q_3^2 + q_4^2}$$

$$\theta = \arcsin(2(q_2q_4 - q_1q_3))$$

$$\varphi = \arctan \frac{2(q_1q_4 + q_2q_3)}{-q_1^2 - q_2^2 + q_3^2 + q_4^2}$$

Přechod mezi vektorem \mathbf{p}_B vyjádřeným v inerciálních souřadnicích a tím samým vektorem \mathbf{p}_I vyjádřeným v tělesových souřadnicích se spočte pomocí maticového násobení:

$$\mathbf{p}_B = \mathbf{R}_I^B(\bar{q})\mathbf{p}_I$$

Použití kvaternionů pro vyjádření rotace ve třídímním prostoru má velké množství výhod, např. menší výpočetní náročnost při násobení kvaternionů oproti násobení rotačních matic nebo nepřítomnost singularity *gimbal lock* (viz sekce 6.2), proto bude dále v této práci používáno. Nicméně člověku je bližší vyjádření rotace v úhlech, proto budou konečné výsledky přepočteny do Eulerových úhlů.

6.4 Určení orientace z výstupu gyroskopu

Pro gyroskopem měřené hodnoty úhlové rychlosti $\omega = [\omega_x \ \omega_y \ \omega_z]$ a kvaternion $\bar{q} = [q_1 \ q_2 \ q_3 \ q_4]^T$ platí vztah (podle [12]):

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{pmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{pmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad (3)$$

Z této soustavy diferenciálních rovnic lze při znalosti počátečních podmínek (často zvolené nulové) počítat numerickými metodami v každém kroku kvaternion \bar{q} , a tedy i Eulerovy úhly. K tomu byla v programu *Inertial Center for PC* použita Eulerova metoda.

Nicméně protože se používá numerická integrace, není toto řešení časově stabilní. Stačí aby byl senzor i jen v malé míře nepřesně zkalibrován a nenulový offset způsobí nárůstání chyby. Tato metoda je tudíž přesná pouze pro krátkodobé určení změny polohy. Pro použitou IMU jednotku lze samotný gyroskop pro určení orientace použít v řádu jednotek sekund, jinak začínají být vypočtené úhly zatíženy chybou přesahující 1° .

6.5 Určení orientace z výstupu akcelerometru a magnetometru – kompas

Pokud na akcelerometr působí pouze tíhové zrychlení $\mathbf{g} = [0 \ 0 \ 1]g$, pak jeho výstupem $\mathbf{a} = [a_x \ a_y \ a_z]$ jsou složky tohoto tíhového zrychlení, ale vyjádřené v tělesových souřadnicích, které závisí na orientaci jednotky v prostoru. Tuto situaci vyjádříme pomocí rotační matice (1) (podle [8]):

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \begin{pmatrix} \cos \psi \cos \theta & \cos \theta \sin \psi & -\sin \theta \\ \cos \psi \sin \varphi \sin \theta - \cos \varphi \sin \psi & \cos \varphi \cos \psi + \sin \varphi \sin \psi \sin \theta & \cos \theta \sin \varphi \\ \sin \varphi \sin \psi + \cos \varphi \cos \psi \sin \theta & \cos \varphi \sin \psi \sin \theta - \cos \psi \sin \varphi & \cos \varphi \cos \theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Tedy pitch θ a roll φ mohou být spočítány podle následujících vztahů:

$$\begin{aligned} \theta &= \arcsin(-a_x), \\ \varphi &= \arcsin \frac{a_y}{\cos \theta} \end{aligned}$$

Znamená to ovšem, že úhel yaw ψ nelze spočítat pouze za použití akcelerometru. Z tohoto důvodu je při jeho výpočtu nutno použít magnetometr. Předpokládáme, že je už zkompenzovaný a jeho výstupní hodnoty jsou navíc normalizované, tedy norma naměřeného vektoru je vždy rovna jedné. Naměřené hodnoty označíme $\mathbf{B}_B = [B_{Bx} \ B_{By} \ B_{Bz}]$.

V inerciální vztažné soustavě I má vektor magnetického pole Země $\mathbf{B}_I = [B_{Ix} \ 0 \ B_{Iz}]$ nulovou y-ovou složku, tj. složku ukazující na východ, neboť magnetické pole Země vždy směřuje k magnetickému severu. Tento vektor dosadíme do

rovnosti s rotačními maticemi:

$$R_\psi \begin{pmatrix} B_{Ix} \\ 0 \\ B_{Iz} \end{pmatrix} = R_{-\theta} R_{-\varphi} \begin{pmatrix} B_{Bx} \\ B_{By} \\ B_{Bz} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} B_{Ix} \cos \psi \\ -B_{Ix} \sin \psi \\ B_{Iz} \end{pmatrix} = \begin{pmatrix} B_{Bx} \cos \theta + B_{By} \sin \theta \sin \varphi + B_{Bz} \sin \theta \cos \varphi \\ B_{By} \cos \varphi - B_{Bz} \sin \varphi \\ -B_{Bx} \sin \theta + B_{By} \cos \theta \sin \varphi + B_{Bz} \cos \theta \cos \varphi \end{pmatrix} \quad (4)$$

Z rovnice (4) pak pro ψ plyne:

$$\tan \psi = \frac{B_{Bz} \sin \varphi - B_{By} \cos \varphi}{B_{Bx} \cos \theta + B_{By} \sin \theta \sin \varphi + B_{Bz} \sin \theta \cos \varphi} = \frac{M_1}{M_2}$$

Při následném výpočtu ψ je potřeba korigovat znaménka v jednotlivých kvadrantech:

$$\psi = \begin{cases} \arctan \frac{M_1}{M_2}, & M_1 \geq 0 \wedge M_2 > 0 \\ 180^\circ + \arctan \frac{M_1}{M_2}, & M_2 < 0 \\ 360^\circ + \arctan \frac{M_1}{M_2}, & M_1 \leq 0 \wedge M_2 > 0 \\ 90^\circ, & M_1 < 0 \wedge M_2 = 0 \\ 270^\circ, & M_1 > 0 \wedge M_2 = 0 \end{cases}$$

V programovacím jazyce C (a mnoha jiných) výpočet funkce \arctan včetně korekce na kvadranty zajišťuje funkce $\text{atan2}(M_1, M_2)$.

Tato metoda tedy dokáže určit všechny tři Eulerovy úhly, tedy orientaci jednotky v prostoru, stejně jako výše uvedený algoritmus využívající pouze výstup z gyroskopu. Výhodou tohoto přístupu je, že nepoužívá integraci, takže je časově stálý, tj. pokud je jednotka v klidu, střední hodnota vypočteného úhlu se s časem nemění. Na druhou stranu, výstup z akcelerometru je zatížen poměrně výrazným šumem, způsobeným drobnými vibracemi a negravitačním zrychlením, který se přenesení i na vypočtené úhly.

Z těchto důvodů je tato metoda vhodná na určování orientace jednotky v prostoru v dlouhodobějším měřítku.

6.6 Fixed-gain observer

Jak bylo řečeno v předchozích sekcích, úhly určené pouze za pomoci gyroskopu se vyznačují časovým driftem a na druhou stranu výstupy akcelerometru a magnetometru jsou zatíženy šumem, což je dělá nepoužitelné pro krátkodobé měření. Proto je potřeba výstupy všech tří sensorů zkombinovat tak, aby vypočtené úhly nebyly zatíženy driftem ani šumem. Jedním z možných přístupů (podle [1]) je použití algoritmu s pozorovatelem:

- Fáze *odhadu*: použití gyroskopu pro odhad změny úhlu. Tato fáze se skládá z určení Eulerových úhlů, jak tomu bylo v sekci 6.4.
- Fáze *korekce*: použití akcelerometru a magnetometru pro opravu driftu gyroskopu. Korekce se provede podle následujících rovnic:

$$\psi_{new} = \psi + L(\hat{\psi} - \psi)$$

$$\theta_{new} = \theta + L(\hat{\theta} - \theta)$$

$$\varphi_{new} = \varphi + L(\hat{\varphi} - \varphi)$$

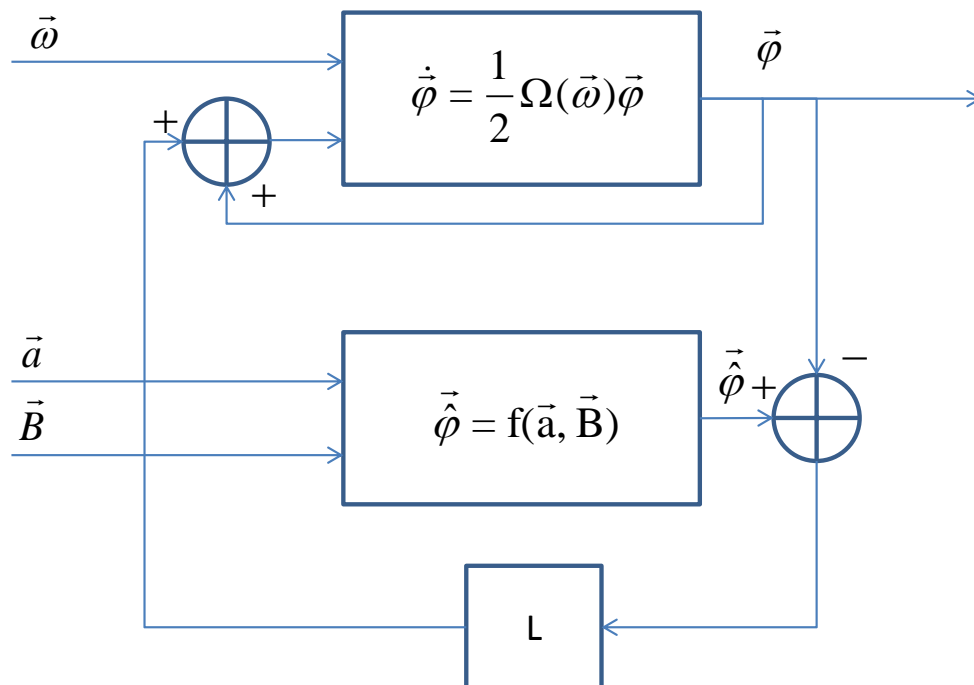
ψ_{new} , θ_{new} , φ_{new} jsou výsledné úhly po korekci, ψ , θ , φ představují úhly spočtené z údajů gyroskopu ve fázi odhadu, $\hat{\psi}$, $\hat{\theta}$, $\hat{\varphi}$ jsou úhly spočtené pomocí akcelerometru a magnetometru, jak je popsáno v sekci 6.5 a L je zvolená konstanta, $L \in \langle 0; 1 \rangle$. Hodnota L rozhoduje o tom, jestli model bude více zohledňovat data z gyroskopu nebo akcelerometru.

Pokud se L blíží k jedničce, věří se více akcelerometru a méně gyroskopu. To znamená, že odhadované úhly jsou méně ovlivněné driftem a šumem gyroskopu, ale jsou citlivější na vibraci a negravitační zrychlení akcelerometru, případně na rušivé magnetické pole.

Pokud se naopak L blíží k nule, více se věří gyroskopu než akcelerometru. Vypočtené úhly jsou poté více ovlivněny driftem a šumem gyroskopu, ale méně citlivé na šum akcelerometru.

Pokud je k dispozici kvalitní gyroskop, je obecně lepší nastavit nižší hodnotu zesílení L , viz [1]. O tom se lze prakticky přesvědčit za použití programu *Inertial Center for PC*, který umožňuje hodnotu L plynule měnit.

Pro přiblížení je algoritmus *Fixed-gain observer* znázorněn na obrázku 6.1.



Obrázek 6.1. Fixed-gain observer.

Fixed-gain observer je velmi podobný níže popsané Kalmanově filtraci. Hlavním rozdílem je, že Kalmanův filtr určuje zesílení L optimálně podle známých charakteristik systému. Navíc ze znalosti systému není potřeba před korekcí převádět naměřená data na úhly.

6.7 Kalmanův filtr

Kalmanův filtr (angl. Kalman filter, KF) je adaptivní rekurzivní filtr sloužící k odhadům stavů dynamického systému. Byl vyvinut v 60. letech 20. století pro účely filtrace šumu v elektrických systémech, ale našel své uplatnění i v aplikacích sledování objektů. Výhodou tohoto filtru je, že se jeho koeficienty v každém kroku dynamicky upravují podle přijaté informace tak, aby určil odhad příštího stavu optimálně.

Originální verze Kalmanova filtru uvažuje diskrétní lineární systém, například ve tvaru (5) (podle [13]).

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{F}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k.\end{aligned}\tag{5}$$

Kde

- k je krok diskrétního systému.
- \mathbf{x}_k je stavový vektor systému.
- \mathbf{u}_k je vektor vstupů.
- \mathbf{F}_k je matice dynamiky systému.
- \mathbf{B}_k je vstupní matice.
- \mathbf{H}_k je výstupní matice.
- \mathbf{y}_k je vektor výstupů.
- \mathbf{w}_k se nazývá systémový šum. Pochází z normálního rozdělení s kovarianční maticí \mathbf{Q}_k , tj. $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$.
- \mathbf{v}_k se nazývá šum měření. Pochází z normálního rozdělení s kovarianční maticí \mathbf{R}_k , tj. $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$.
- Pro \mathbf{w}_k a \mathbf{v}_k musí platit, že jsou to bílé šумы, jsou gaussovské s nulovou střední hodnotou a jsou nezávislé.

Nicméně pro sledování objektů se často používají systémy nelineární, proto je v této práci použit tzv. *rozšířený Kalmanův filtr* [13] (*angl. extended Kalman filter, EKF*), který oproti standardnímu Kalmanovu filtru v každém kroku systém linearizuje. Nelineární systém může být obecně popsán rovnicemi (6).

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) + \mathbf{w}'(t) \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}, t) + \mathbf{v}'(t),\end{aligned}\tag{6}$$

kde \mathbf{f} je známá nelineární (vektorová) funkce stavu \mathbf{x} , signálu \mathbf{u} a času; \mathbf{h} je známá nelineární (vektorová) funkce stavu a času; \mathbf{w}' a \mathbf{v}' jsou spojité bílé šумы.

EKF tento systém linearizuje. V každém kroku se zvolí pracovní body $\mathbf{x}^*(t)$ a $\mathbf{y}^*(t)$, které jsou řešením rovnic (7).

$$\begin{aligned}\dot{\mathbf{x}}^*(t) &= \mathbf{f}(\mathbf{x}^*, \mathbf{u}, t) \\ \mathbf{y}^*(t) &= \mathbf{h}(\mathbf{x}^*, t)\end{aligned}\tag{7}$$

Pak chyba stavového vektoru $\Delta \mathbf{x}(t)$ a vektoru výstupních hodnot $\Delta \mathbf{y}(t)$ je definována vztahem (8), resp. (9).

$$\Delta \mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}^*(t)\tag{8}$$

$$\Delta \mathbf{y}(t) = \mathbf{y}(t) - \mathbf{y}^*(t)\tag{9}$$

Linearizovaný systém popisující vývoj odchylek je pak následující:

$$\begin{aligned}\Delta \dot{\mathbf{x}}(t) &= \mathbf{F}(t) \Delta \mathbf{x}(t) + \mathbf{w}'(t) + e_x(t), \\ \Delta \mathbf{y}(t) &= \mathbf{H}(t) \Delta \mathbf{x}(t) + \mathbf{v}'(t) + e_y(t),\end{aligned}\tag{10}$$

kde $e_x(t)$ a $e_y(t)$ jsou chyby linearizace a $\mathbf{F}(t)$ a $\mathbf{H}(t)$ jsou Jakobiho matice funkcí \mathbf{f} a \mathbf{h} :

$$\mathbf{F}(t) = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}^*}, \mathbf{H}(t) = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}^*}$$

Protože Kalmanův filtr pracuje s diskrétními systémy, je nezbytné lineární spojité systém popsany rovnicemi (10) převést na jeho diskrétní podobu (11).

$$\begin{aligned} \Delta \mathbf{x}_{k+1} &= \mathbf{\Phi}_k \Delta \mathbf{x}_k + \mathbf{w}_k, \\ \Delta \mathbf{y}_k &= \mathbf{H}_k \Delta \mathbf{x}_k + \mathbf{v}_k, \end{aligned} \quad (11)$$

kde k je index a platí $t_k = kT_s$, kde T_s je vzorkovací perioda. Dále $\mathbf{H}_k = \mathbf{H}(t_k)$ a

$$\mathbf{\Phi}_k = e^{\mathbf{F}(t_k)T_s}.$$

Při popisu průběhu Kalmanova filtru se používá značení:

- \hat{x} pro odhad x .
- $\hat{x}_{j|i}$ pro odhad stavu x v čase t_j , který používá všechna měření až do času t_i .

Samotný algoritmus se skládá ze tří částí: inicializace, odhadu a korekce (podle [13]).

■ 6.7.1 Inicializace

Předpokládáme, že první měření proběhne v čase t_1 . Počáteční hodnoty odhadu stavu $\hat{\mathbf{x}}_{0|0}$, chyby stavu $\Delta \hat{\mathbf{x}}_{0|0}$ a kovarianční matice $\mathbf{P}_{0|0}$ se nastaví na hodnoty:

$$\hat{\mathbf{x}}_{0|0} = E[\mathbf{x}_0]$$

$$\Delta \hat{\mathbf{x}}_{0|0} = 0$$

$$\mathbf{P}_{0|0} = cov[\mathbf{x}_0]$$

■ 6.7.2 Odhad

Odhad se skládá z následujících fází:

1. Řešení rovnice $\dot{\mathbf{x}}^*(t) = \mathbf{f}(\mathbf{x}^*, \mathbf{u}, t)$ pro \mathbf{x}^* , například numerickým integrováním přes čas $t \in \langle t_k, t_{k+1} \rangle$ s počáteční podmínkou $\mathbf{x}^*(t_k) = \hat{\mathbf{x}}_{k|k}$. Řešení označíme jako $\hat{\mathbf{x}}_{k+1|k} = \mathbf{x}^*(t_{k+1})$.
2. Výpočet Jakobiho matice $\mathbf{F}(t)$ pro $t \in \langle t_k, t_{k+1} \rangle$.
3. Výpočet matice přechodu $\mathbf{\Phi}_k$ a kovarianční matice systémového šumu \mathbf{Q}_k .
4. Odhad kovarianční matice $\mathbf{P}_{k+1|k}$ podle vztahu:

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^T + \mathbf{Q}_k.$$

5. Inkrementování indexu $k \rightarrow k + 1$.

6.7.3 Korekce

1. Linearizace výstupní matice v bodě $\mathbf{x}^*(t_k)$

$$\mathbf{H}_k = \mathbf{H}(t_k) = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}}$$

2. Výpočet Kalmanova zisku

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T [\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k]^{-1}$$

3. Výpočet chyby stavu

$$\Delta \hat{\mathbf{x}}_{k|k} = \mathbf{K}_k \Delta \mathbf{y}_k$$

4. Korekce odhadu stavu

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \Delta \hat{\mathbf{x}}_{k|k}$$

5. Aktualizace kovarianční matice

$$\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k] \mathbf{P}_{k|k-1}$$

6.7.4 Použitý model

Pro potřeby určování orientace jednotky v prostoru jsem použil model popsáný v [14].

Spojité nelineární systém je popsán rovnicemi:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \omega) + \mathbf{w} \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}) + \mathbf{v}, \end{aligned} \quad (12)$$

kde

- $\mathbf{x} = [\bar{q} \quad \mathbf{b}_\omega]^T$ reprezentuje stav systému složený z kvaternionu $\bar{q} = [q_1 \quad q_2 \quad q_3 \quad q_4]$ a vektoru offsetů gyroskopu $\mathbf{b}_\omega = [b_{\omega_x} \quad b_{\omega_y} \quad b_{\omega_z}]$.
- $\omega = [\omega_x \quad \omega_y \quad \omega_z]^T$ je vektor úhlové rychlosti.
- $\mathbf{y} = [\mathbf{a} \quad \mathbf{m}]^T = [a_x \quad a_y \quad a_z \quad m_x \quad m_y \quad m_z]^T$ je vektor složený z naměřeného zrychlení a velikosti magnetického pole.
- \mathbf{w}, \mathbf{v} je systémový, resp. měřicí šum.

Pokud kromě gravitačního zrychlení zanedbáme všechna ostatní, lze IMU jednotku popsat modelem (13).

$$\begin{aligned} \omega &= \mathbf{G}_g \omega_r + \mathbf{b}_\omega + \mathbf{v}_g \\ \mathbf{a} &= \mathbf{G}_a [\mathbf{R}_I^B(\bar{q}) \mathbf{g}] + \mathbf{v}_a \\ \mathbf{m} &= \mathbf{G}_m [\mathbf{R}_I^B(\bar{q}) \mathbf{H}] + \mathbf{v}_m \end{aligned} \quad (13)$$

kde

- $\mathbf{G}_g, \mathbf{G}_a, \mathbf{G}_m$ jsou po řadě kompenzační matice gyroskopu, akcelerometru a magnetometru.
- $\mathbf{v}_g, \mathbf{v}_a, \mathbf{v}_m$ jsou po řadě šumy gyroskopu, akcelerometru a magnetometru.
- $\omega_r = [\omega_{rx} \quad \omega_{ry} \quad \omega_{rz}]^T$ jsou hrubá data gyroskopu.

- $\mathbf{R}_I^B(\bar{q})$ je rotační matice z inerciální do tělesové soustavy vyjádřená pomocí složek kvaternionu \bar{q} , viz (2).
- $\mathbf{g} = [0 \ 0 \ g]^T$ je vektor tíhového zrychlení.
- $\mathbf{H} = [H_x \ 0 \ H_y]^T$ je vektor magnetické indukce zemského magnetického pole v inerciální soustavě.

Z modelu (12) a z rovnice (3) plynou vztahy pro nelineární funkce $\mathbf{f}(\mathbf{x}, \omega)$ a $\mathbf{h}(\mathbf{x})$:

$$\mathbf{f}(\mathbf{x}, \omega) = \begin{bmatrix} \frac{1}{2} \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \cdot \begin{bmatrix} \omega_{rx} - b_{\omega x} \\ \omega_{ry} - b_{\omega y} \\ \omega_{rz} - b_{\omega z} \end{bmatrix} \\ \mathbf{b}_{\omega}^T \end{bmatrix}$$

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \mathbf{R}_I^B(\bar{q})\mathbf{g} \\ \mathbf{R}_I^B(\bar{q})\mathbf{H} \end{bmatrix}$$

Kovarianční matice \mathbf{R}_k a \mathbf{Q}_k byly určeny podle [15]:

$$\mathbf{R}_k = \begin{bmatrix} \mathbf{R}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_m \end{bmatrix}$$

$$\mathbf{Q}_k = \begin{bmatrix} (T_s/2)^2 \Xi_k \Sigma_g \Xi_k^T & \mathbf{0} \\ \mathbf{0} & {}^g \Sigma_k \end{bmatrix}$$

kde

- $\mathbf{R}_a = \text{diag}(\sigma_{ax}^2, \sigma_{ay}^2, \sigma_{az}^2)$ je diagonální kovarianční matice měřeného zrychlení. Prvky na diagonále jsou pak rozptyly zrychlení v jednotlivých osách.
- $\mathbf{R}_m = \text{diag}(\sigma_{mx}^2, \sigma_{my}^2, \sigma_{mz}^2)$ je diagonální kovarianční matice měřené magnetické indukce. Prvky na diagonále jsou pak rozptyly mag. indukce v jednotlivých osách.
- $\Sigma_g = \text{diag}(\sigma_{gx}^2, \sigma_{gy}^2, \sigma_{gz}^2)$ je diagonální kovarianční matice měřené úhlové rychlosti. Prvky na diagonále jsou pak rozptyly úhlové rychlosti v jednotlivých osách.
- ${}^g \Sigma_k = \text{diag}(T_s \sigma_{gk1}^2, T_s \sigma_{gk2}^2, T_s \sigma_{gk3}^2)$ je diagonální procesní kovarianční matice. Prvky na diagonále je většinou nutné nalézt empiricky.
- Matice Ξ_k je definována vztahem (14).
- T_s je vzorkovací perioda.

$$\Xi_k = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \quad (14)$$

6.7.5 Kompenzace zrychlení

Protože model (13) nepočítá s jiným než tíhovým zrychlením, je pro zpřesnění měření výhodné kompenzovat zrychlení související s pohybem lidského těla. Byly vyzkoušeny a ohodnoceny dva rozdílné přístupy.

První metoda, popsaná v [14], R-adaptivní algoritmus, zavádí měřící okno o N vzorcích, pomocí něhož se aktualizuje hodnota rozptylů podle vzorce:

$$\sigma_k^2 = \sigma_{ax}^2 = \sigma_{ay}^2 = \sigma_{az}^2 = \frac{1}{N+1} \sum_{i=k-N+1}^k (||a_i|| - ||a_{i-1}||)^2,$$

kde $\|a_i\|$ je modul naměřeného zrychlení.

Druhá metoda, podle [15], nastavuje určitou mez pro rozdíl mezi normou naměřeného zrychlení a normou tíhového zrychlení. Pokud je tato mez překročena, je rozptyl nastaven na velmi vysokou hodnotu, a tak se EKF začne spoléhat pouze na informace z magnetického senzoru.

6.8 Experimentální ověření

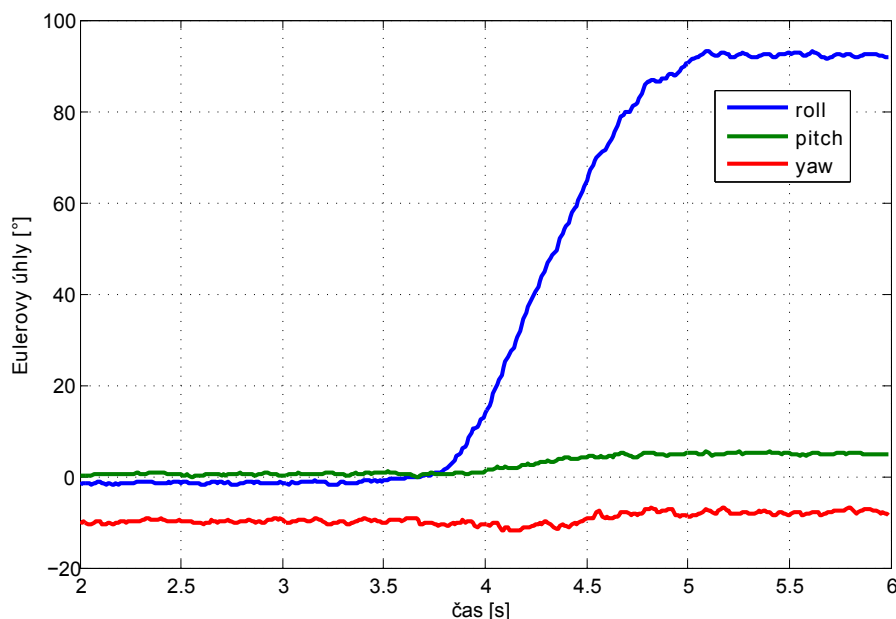
Algoritmus rozšířeného Kalmanova filtru byl implementován nejprve v *Matlabu*.

Dále se lze o funkčnosti přesvědčit z implementace EKF v programu *Inertial Center for PC*, který je více popsán v příloze B. Program umožňuje on-line vyhodnocování dat z jednotky a jejich vykreslení do grafu. Kromě toho, jedna z jeho součástí je také 3D krychle, která se otáčí v prostoru podle vypočtených Eulerových úhlů, a je tak možné názorně porovnat polohu krychle se skutečnou polohou IMU jednotky.

Všechny použité skripty a příslušná zdrojová data jsou nahrána na příloženém CD.

6.8.1 Experiment – otáčení kolem jedné osy

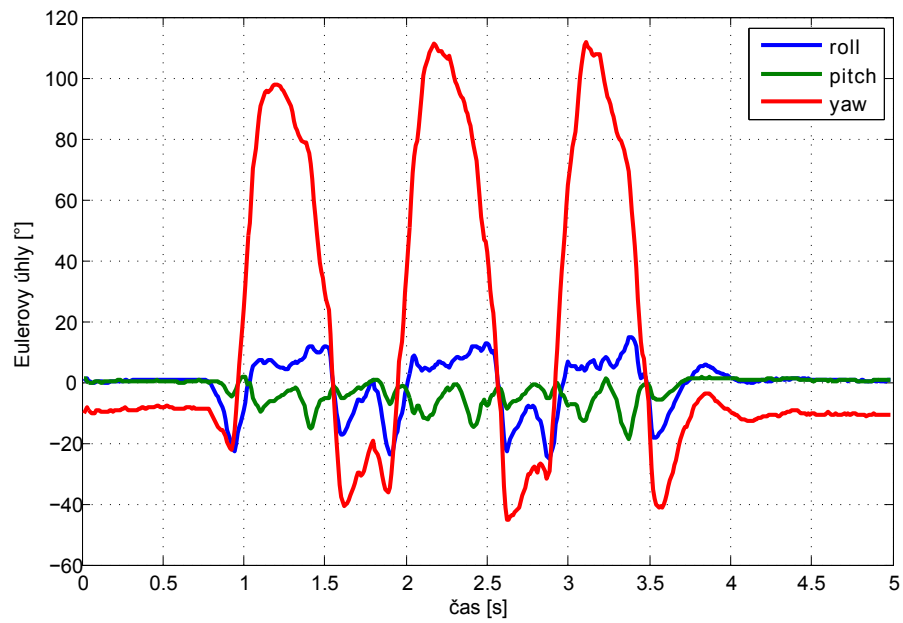
Prvním jednoduchým pokusem bylo určení orientace jednotky při pomalém otočení o 90° kolem osy x. Výsledek (převedený z kvaternionů na Eulerovy úhly) je uveden na obrázku 6.2.



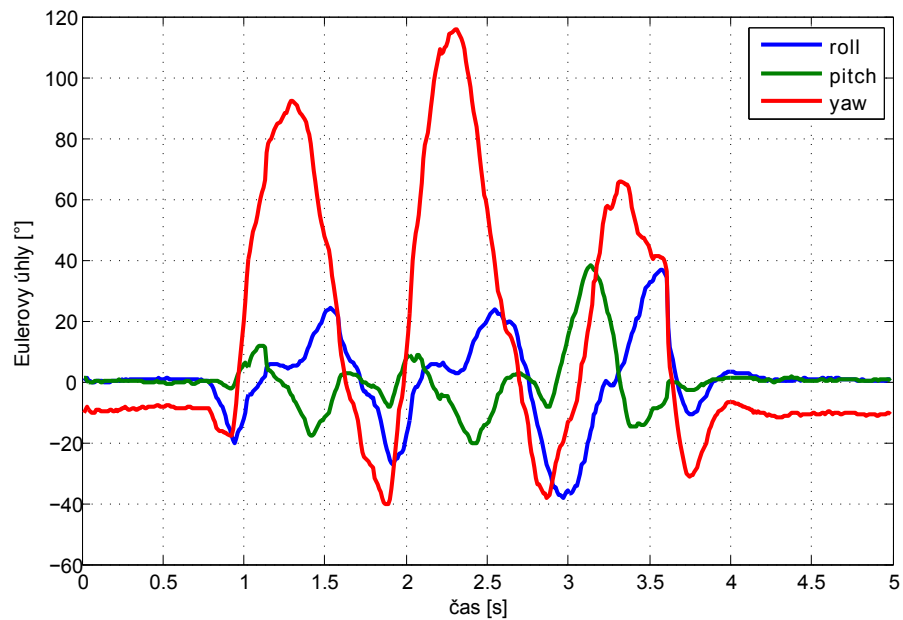
Obrázek 6.2. Rotace jednotky kolem jedné osy.

6.8.2 Experiment – data se zrychlením

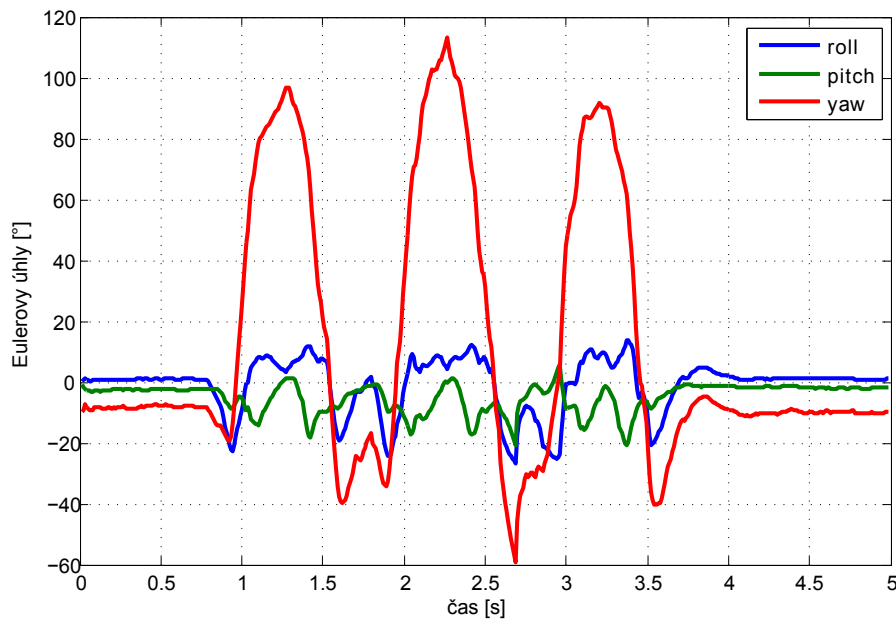
Dalším experimentem bylo vyhodnocení pohybu se zrychlením. Jednotka byla umístěna na stole a otočena rychle třikrát za sebou okolo osy z cca o 100° a zase zpět do výchozí polohy. Poté byla data zpracována originálním EKF, EKF s R-adaptivním algoritmem a EKF s prahováním zrychlení. Výsledky lze porovnat z grafů 6.3, 6.4 a 6.5.



Obrázek 6.3. Rychlé rotace jednotky.



Obrázek 6.4. Rychlé rotace jednotky (R-adaptivní algoritmus).



Obrázek 6.5. Rychlé rotace jednotky (prahování zrychlení).

Na grafu 6.3 je vidět, že samotný proces otáčení je vyhodnocen vcelku přesně. Poté byla jednotka rychle zastavena, takže začalo působit poměrně velké zrychlení (zpomalení). To je na grafu patrné z toho, že se úhly okamžitě nevrátily na svoji původní úroveň, ale nastal jistý přechodný děj. Úhel yaw totiž nikdy ve skutečnosti nedosáhl -40° .

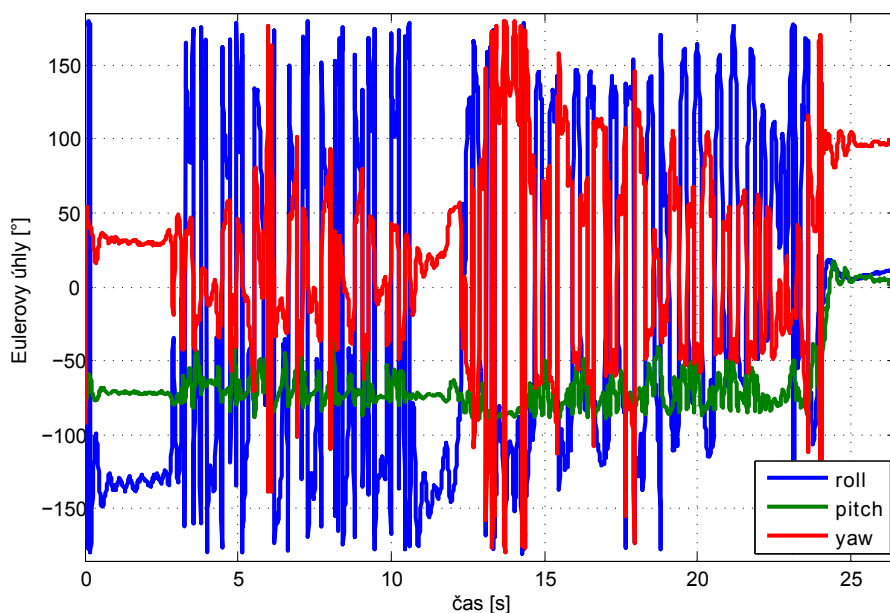
Graf 6.4 zobrazuje výsledek EKF s R-adaptivním algoritmem. Ačkoli jsem se pokusil tento algoritmus co nejlépe vyladit, artefakty způsobené přidáním zrychlení ve výsledku zůstávají. Dokonce se znehodnotily i některé části grafu, které nepotřebovaly korekci. R-adaptivní algoritmus se tedy neukázal jako přesvědčivý prostředek k řešení problému přídatného zrychlení.

Třetí graf 6.5 ukazuje spočtený průběh po aplikaci EKF s prahováním zrychlení. Podle doporučení z [15] byl práh nastaven na 20% normy tíhového zrychlení. Můžeme si všimnout, že algoritmus začal prahovat vždy při změně směru rotace, tedy v těchto bodech detekoval zrychlení správně. Je to zřejmé z tvaru ostrých špiček, které na grafu 6.3 nejsou přítomné. Přestože zrychlení bylo detekováno na správných místech, tuto změnu průběhu považuji za nevhodnou, neboť vzniklé špičky působí uměle a mohou případně další výpočty poškodit více než originální EKF.

Ačkoli se v několika pracích ([14], [15]) objevují algoritmy, které se snaží kompenzovat přídatné zrychlení změnou kovarianční matice \mathbf{R} , ukázalo se, že tento přístup zřejmě není úplně vhodný a nedokáže se zbavit artefaktů vzniklých nežádoucím zrychlením. Naopak se domnívám, že mohou signál více poškodit, než vylepšit. Proto ani tyto metody nejsou použity v implementaci EKF v programu *Inertial Center for PC*.

6.8.3 Experiment – určení orientace při chůzi

Dalším experimentem bylo připevnění jednotky na opasek a chůze s ní. Na obrázku 6.6 je znázorněna orientace jednotky během pohybu – relativně pomalá chůze sedm metrů na západ, otočení a chůze zpět, tedy sedm metrů směrem na východ. Ze záznamu jsou patrné jednotlivé kroky, které by zcela jistě bylo možné vhodným nástrojem detekovat, dále otočka a následně chůze jiným směrem.



Obrázek 6.6. Záznam chůze (orientace).

6.8.4 Experiment – určení polohy při chůzi

Jednotkou naměřené zrychlení je součtem zrychlení gravitačního a zrychlení způsobeného pohybu lidského těla. Z principu by tedy bylo možné podle následujících vztahů určit pouze zrychlení pohybů lidského těla a z něj integrací rychlost v jednotlivých osách a také polohu jednotky v prostoru:

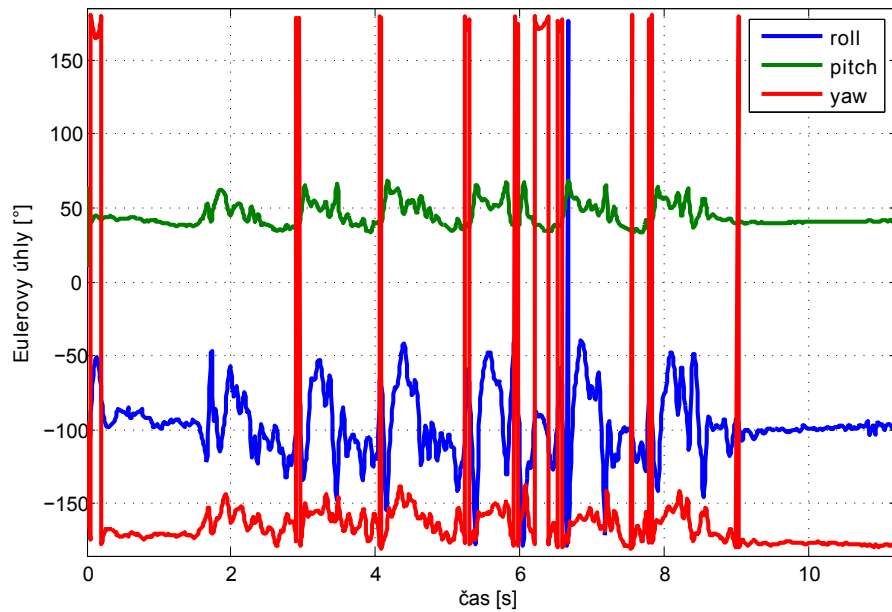
$$\mathbf{v}_k = \mathbf{v}_{k-1} + (\mathbf{R}_B^I \mathbf{a} - \mathbf{g}) T_s$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + (\mathbf{v}_k + \mathbf{v}_{k-1}) \frac{T_s}{2} \quad (15)$$

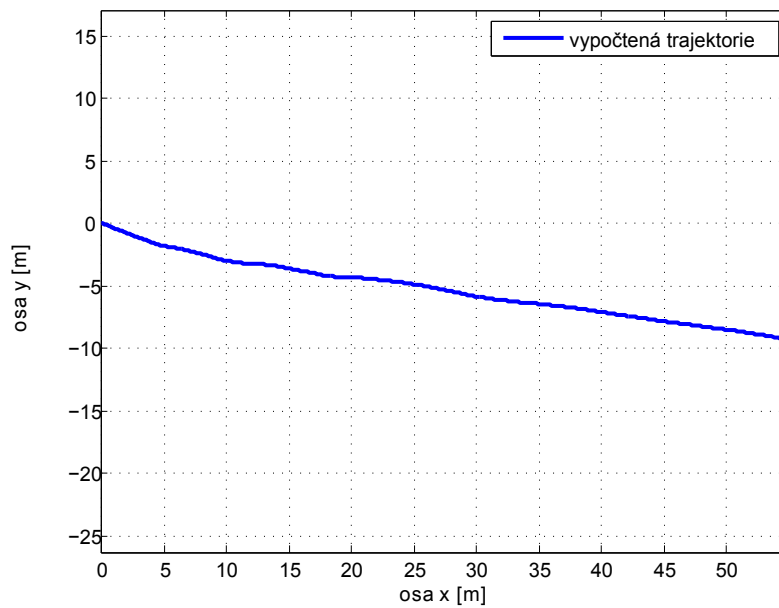
Kde \mathbf{v}_k je vektor rychlosti v k-tém kroku, \mathbf{R}_B^I je rotační matice z tělesových do inerciálních souřadnic, \mathbf{a} je vektor naměřeného zrychlení, \mathbf{g} je vektor tíhového zrychlení v inerciálních souřadnicích, T_s je vzorkovací perioda a \mathbf{x}_k je poloha v k-tém kroku.

Bohužel rovnice (15) dávají smysluplné výsledky pouze v ideálním případě. V praxi není dokonalá ani kalibrace, ani určení orientace, tedy matice \mathbf{R}_B^I . To pak vytváří lineární chybu při určování rychlosti a dokonce chybu kvadratického řádu při určování polohy. Proto není příliš běžné, aby se IMU jednotky samotné používaly k určování polohy.

Druhý experiment spočíval v chůzi po vyznačených značkách, jejichž poloha je přesně známa. Ušlá dráha měřila 5 m. Obrázek 6.7 ukazuje výsledek Kalmanovy filtrace, obrázek 6.8 pak následně vypočtenou polohu, jež je evidentně zatížena chybou. Z grafu lze odhadnout, v tomto konkrétním případě, přibližný nárůst chyby o 0,2 m/s. To je sice o řád méně, než bylo odhadnuto v úvodu, pořád se ale jedná o značnou chybu znemožňující další zpracování.

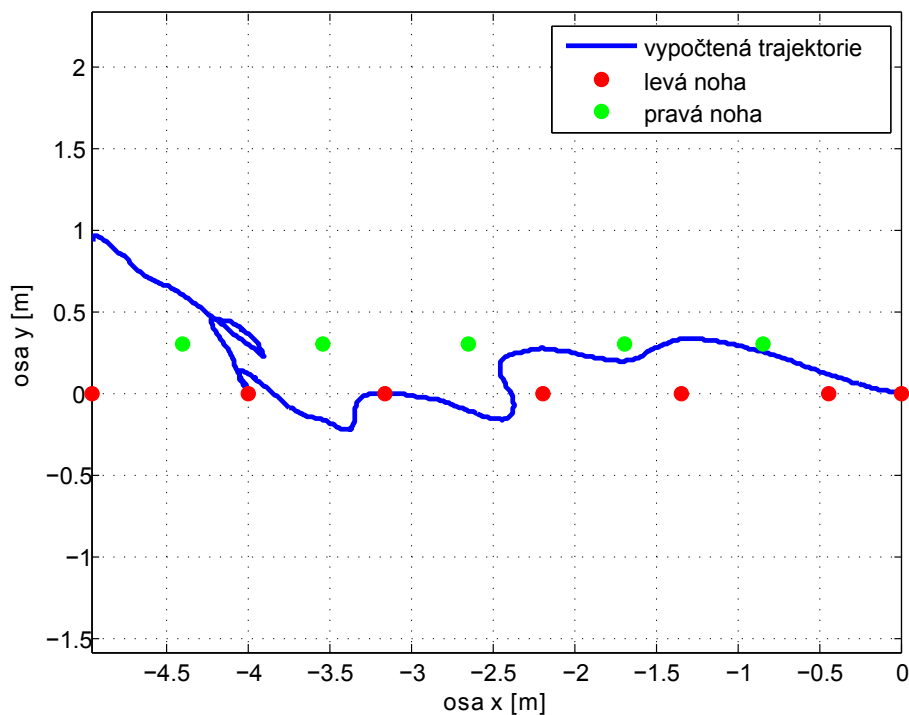


Obrázek 6.7. Záznam chůze č. 2 (orientace).

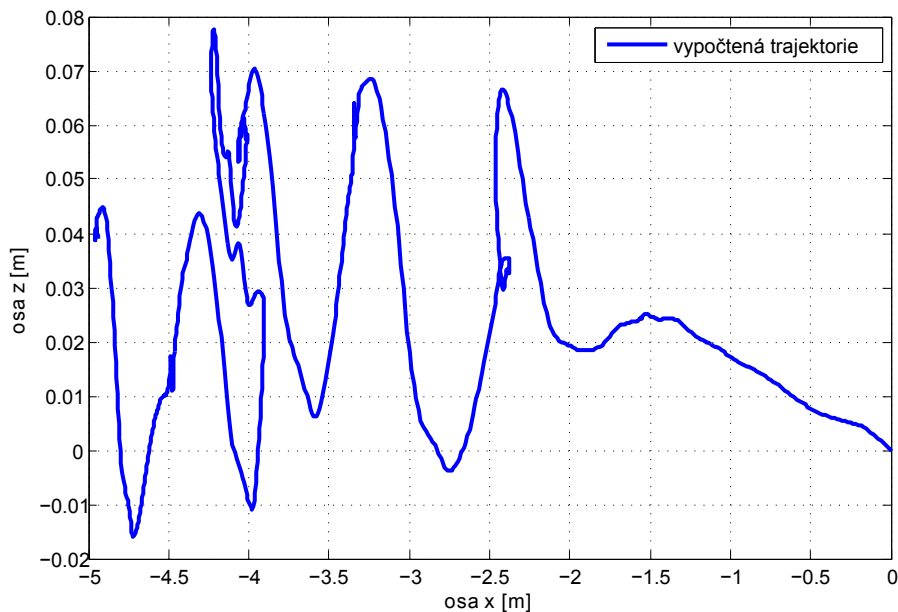


Obrázek 6.8. Záznam chůze č. 2 (neupravená trajektorie).

Pro odstranění narůstající chyby byla rychlost, spočtená podle vzorce (15), vyfiltrována v Matlabu digitálním filtrem s nulovou fází, tj. příkazem `filtfilt(b, a, signal)`. Koeficienty filtru byly zvoleny tak, aby filtr fungoval jako horní propust, tedy např. $a = [1, -0.995]$, $b = [1, -1]$. Na grafu 6.9 je znázorněna spočtená trajektorie v osách X-Y včetně vyznačených skutečných stop. Na grafu 6.10 je pak tatáž trajektorie v osách X-Z.



Obrázek 6.9. Záznam chůze č. 2 (trajektorie X-Y, po filtraci).



Obrázek 6.10. Záznam chůze č. 2 (trajektorie X-Z, po filtraci).

Z grafu 6.9 plyne, že ačkoli byla provedena filtrace, trajektorie je jen přibližná a po velmi krátkém čase, cca 7 s, se odchyluje od skutečné polohy už o 1 m. Tato metoda samotná tedy není vhodná pro delší pozorování, nicméně v kratším časovém měřítku může dávat relativně uspokojivé výsledky. Pokud by se navíc doplnila o značkovače umístěné na lidském těle, na jejichž základě by se vždy při průchodu označované části těla určitou polohou výpočty zkorigovaly, mohla by tato metoda být použitelná i pro dlouhodobější měření. Při měření pohybů nohou by se například jednalo o magnetická

čidla umístěná na botách, která by po přiblížení, tedy na začátku každého kroku, vyslala do řídicí jednotky nulovací impuls, který by zresetoval EKF a začalo by se integrovat od nuly. Tím by se mohla získat poměrně kvalitní informace o sledovaném pohybu, neboť jeden krok netrvá příliš dlouho a nestihne se tak naakumulovat velká chyba.

Další možností je připojit k naměřeným datům ještě hodnoty z navigace GPS. Inerciální navigace by určovala polohu v krátkých okamžicích, GPS by ji pak jednou za čas korigovala. S klesající cenou GPS přijímačů je tato metoda dobře realizovatelná. Nicméně, je nutné myslet na to, že například v budovách či zastavěných územích je přesnost GPS poměrně nízká.

Graf 6.10 ukazuje polohu jednotky v osách X-Z, to znamená pohyby těla, v tomto případě nohy, v sagitální rovině (dělí tělo na levou a pravou stranu). Z grafu je vidět, přes některé chyby zpracování, jasný sinusoidální trend pohybu nohy. Z tohoto záznamu by bylo jistě možné vyčíst např. počet kroků, výšku zdvihu kyčle nebo také popsat celkový průběh kroku a porovnat jej u více osob.

Kapitola 7

Závěr

V rámci této bakalářské práce jsem se seznámil prostřednictvím kitu *Primer2* s procesory s jádrem ARM a jejich programováním. Na základě těchto znalostí vzniknul program *Inertial Center 1.0.1* pro vyčítání dat z IMU jednotky, jejich ukládání na paměťovou kartu a současné posílání do PC po sběrnici USB. Dále jsem navrhnul pro jednotlivé senzory jednotky jednoduché kalibrační postupy, které jsem následně implementoval do téhož programu.

Druhá polovina práce pojednává o zpracování již kompenzovaných dat pro účely tzv. inerciální navigace. Bylo uvedeno a prakticky vyzkoušeno několik přístupů k určení orientace jednotky a porovnány klady a zápory jednotlivých metod. Jako nejpokročilejší metoda, která se snaží minimalizovat nedostatky jednotlivých senzorů, byla stanovena Kalmanova filtrace. Následně bylo provedeno několik pokusů, od těch nejjednodušších, které se zabývaly jen pohyby v jedné ose, až po pokusy se skutečným monitorováním lidské chůze. Na základě všech popsaných pozorování jsem dospěl k závěru, že inerciální navigace s relativně levnými MEMS senzory dokáže určit poměrně přesně orientaci jednotky v prostoru. Nicméně stačí, aby určení orientace vykazovalo malou chybu a ta se značně promítne do výpočtu rychlosti a polohy jednotky. Proto bylo v této práci navrženo filtrování, po jehož aplikaci určená poloha v krátkém časovém měřítku, cca do 10 s, poměrně dobře odpovídá skutečně uražené trajektorii. Chyba mnou použitého přístupu je dokonce o řád menší, než jsem odhadoval v úvodní kapitole. Dále byly navrženy pomocné metody, které by umožnily dlouhodobé měření: využití markerů a přidružené družicové navigace.

Pro praktickou demonstraci většiny postupů z této bakalářské práce vzniknul program *Inertial Center for PC*, který pracuje s on-line daty z *Primeru* přenášených po USB. Návod k jeho použití se nachází v příloze B.

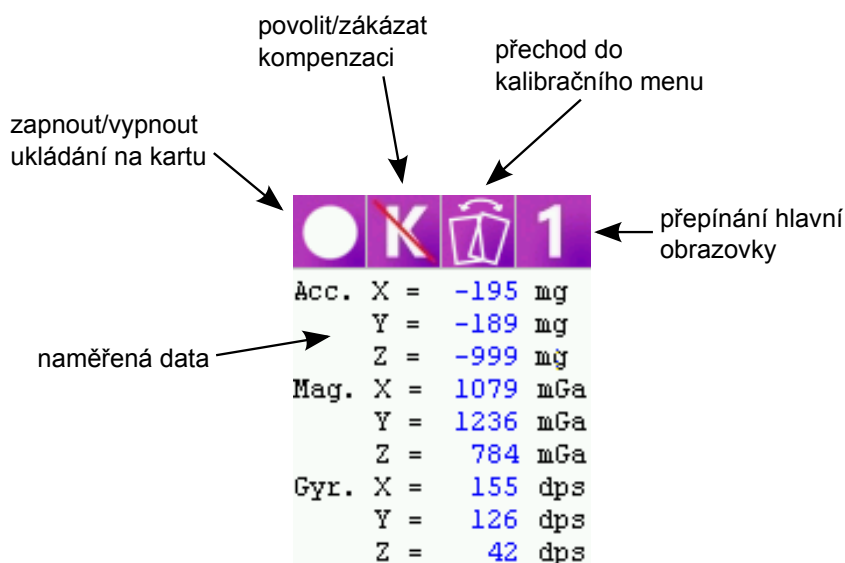
Literatura

- [1] CH Robotics. *The CH Robotics Content Library*.
<http://www.chrobotics.com/library>.
- [2] STMicroelectronics. *Sensor module: 3-axis accelerometer and 3-axis magnetometer datasheet*, 2011.
https://www.pololu.com/file/download/LSM303DLM.pdf?file_id=0J514.
- [3] STMicroelectronics. *MEMS motion sensor: ultra-stable three-axis digital output gyroscope datasheet*, 2010.
http://www.st.com/web/catalog/sense_power/FM89/SC1288/PF250373.
- [4] STMicroelectronics. *MEMS pressure sensor: 300 - 1100 mbar absolute digital output barometer*, 2010.
<http://www.stm32circle.com/resources/Datasheets/LPS001D.pdf>.
- [5] Raisonance SAS. Extension board open4 - inertial moving system, 2011.
http://www.stm32circle.com/resources/Open4/Schematics/PU_Inertial_v1.1.pdf.
- [6] Raisonance SAS. *STM32-Primer2 User Manual*.
<http://www.stm32circle.com/resources/download.php?STM32-Primer2-Manual.pdf>.
- [7] Mohinder S. Grewal, Angus P. Andrews, and Chris G. Bartone. *Global Navigation Satellite Systems, Inertial Navigation, and Integration*. Wiley, third edition, 2013.
- [8] STMicroelectronics. *Application note-AN3192: Using LSM303DLH for a tilt compensated electronic compass*.
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/application_note/CD00269797.pdf.
- [9] Martin Šipoš et al. Analyses of triaxial accelerometer calibration algorithms. *IEEE Sensors Journal*, 12(5), 2012.
- [10] STMicroelectronics. *Everything about STMicroelectronics 3-axis digital MEMS gyroscopes*, 2011.
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/technical_article/DM00034730.pdf.
- [11] Mark Looney. A simple calibration for mems gyroscopes. *EDN Europe*, 2010.
- [12] Nicolas Trawny and Stergios I. Roumeliotis. *Indirect Kalman Filter for 3D Attitude Estimation. A Tutorial for Quaternion Algebra*. 2005.
- [13] Frank L. Lewis et al. *Autonomous Mobile Robots. Sensing, Control, Decision Making and Application*. Taylor & Francis, 2006.
- [14] Z. Lin, M. Zecca, et al. Development of an ultra-miniaturized inertial measurement unit wb-3 for human body motion tracking. *SI International*, 2010.
- [15] Angelo M. Sabatiny. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *Biomedical Engineering, IEEE Transactions*, 53(7), 2006.

Příloha A

Inertial Center 1.0.1

Aplikace *Inertial Center 1.0.1* vyvinutá pro *Primer* dovoluje kalibrovat připojenou IMU jednotku, číst změřená data na displeji, ukládat je na microSD kartu a posílat je do PC prostřednictvím sběrnice USB.



Obrázek A.1. Popis úvodní obrazovky Inertial Center 1.0.1.

A.1 Zapnutí aplikace

- Dlouze stisknout oranžové tlačítko (dále jen tlačítko). Tím se zapne samotný *Primer*.
- Stisknout znovu tlačítko a z menu pomocí joysticku vybrat aplikaci *Inertial*.

A.2 Hlavní obrazovka

Většinu hlavní obrazovky zabírá výpis měřených hodnot. Pokud v pravém horním rohu je indikována číslice 1, pak se zobrazují změřené hodnoty zrychlení, magnetické indukce a úhlové rychlosti.

Stisknutím fialového tlačítka 1 se přeneseme na obrazovku s výpisem měřeného tlaku a teploty. Stisknutím tlačítka 2 se dostaneme zpět na výchozí obrazovku.

A.3 Záznam dat na paměťovou kartu

Pokud je v zařízení přítomna microSD karta, je možné na ni ukládat naměřené hodnoty. Záznam se zahájí stiskem prvního tlačítka z horní nabídky – bílým kolečkem. Toto kolečko se během nahrávání přepne na bílý čtverec značící aktivní záznam. Po stisknutí tlačítka se čtvercem se záznam ukončí. Na paměťovou kartu se data uloží do souboru *IMUdata.csv* ve snadno přenosném formátu CSV.

A.4 Kompenzace

Po spuštění aplikace jsou z paměťové karty načteny do paměti soubory *kalibACC.txt*, *kalibGYRO.txt* a *kalibMAG.txt* s kalibračními maticemi jednotlivých senzorů. Tyto soubory jsou vygenerovány po kalibraci jak přímo v *Primeru*, tak i v aplikaci pro PC (viz příloha B).

V úvodním nastavení jsou po USB posílána nekompenzovaná data. Tento stav je vyznačen druhým tlačítkem v horní liště – přeškrtnutým písmenem *K*. Kompenzaci podle načtených matic je nejdříve potřeba povolit, a to stiskem tohoto tlačítka. Opětným stiskem se kompenzace zakáže.

A.5 Kalibrace

Stiskem třetího tlačítka zleva vstoupíme do kalibračního menu. Horní nabídka se změní na tlačítka s označením *A*, *M* a *G*, která po vybrání zahájí kalibraci po řadě akcelerometru, magnetometru a gyroskopu. Stisk oranžového tlačítka nás přenese zpět na úvodní obrazovku.

Implementované kalibrační metody jsou zjednodušenými verzemi kalibračních metod z *Inertial Center for PC*. Jejich výstupem je matice offsetů a matice škálovacích faktorů. Tato kalibrace neumí určit matici neortogonalit z důvodu omezených výpočetních kapacit *Primeru*!

A.6 Kalibrace akcelerometru

Pro kalibraci akcelerometru stiskněte *A* v kalibračním menu. Pokračujte podle následujících pokynů:

- Položte *Primer* na vodorovnou podložku s osou *Z* mířící směrem dolů, stiskněte tlačítko a počkejte 5 s. K orientaci os viz obr. 5.3.
- Položte *Primer* na vodorovnou podložku s osou *Z* mířící směrem nahoru, stiskněte tlačítko a počkejte 5 s.
- Položte *Primer* na vodorovnou podložku s osou *Y* mířící směrem dolů, stiskněte tlačítko a počkejte 5 s.
- Položte *Primer* na vodorovnou podložku s osou *Y* mířící směrem nahoru, stiskněte tlačítko a počkejte 5 s.
- Položte *Primer* na vodorovnou podložku s osou *X* mířící směrem dolů, stiskněte tlačítko a počkejte 5 s.
- Položte *Primer* na vodorovnou podložku s osou *X* mířící směrem nahoru, stiskněte tlačítko a počkejte 5 s.
- Kalibrační matice se uloží na paměťovou kartu do souboru *kalibACC.txt*. Pokud už na kartě soubor se stejným jménem existuje, bude přepsán.

A.7 Kalibrace magnetometru

Pro kalibraci magnetometru stiskněte *M* v kalibračním menu. Je nutné, aby kalibrace byla prováděna v magneticky čistém prostředí. Pokračujte podle následujících pokynů:

- Položte *Primer* na vodorovnou položku a po stisku tlačítka s ním otáčejte okolo osy *Z*, kterou orientujte směrem kolmo do podložky. Po několika plných rotacích stiskněte znovu tlačítko.

- Postavte *Primer* na vodorovnou položku a po stisku tlačítka s ním otáčejte okolo osy Y, kterou orientujte směrem kolmo do podložky. Po několika plných rotacích stiskněte znovu tlačítko.
- Postavte *Primer* na vodorovnou položku a po stisku tlačítka s ním otáčejte okolo osy X, kterou orientujte směrem kolmo do podložky. Po několika plných rotacích stiskněte znovu tlačítko.
- Stiskněte tlačítko a rotujte *Primerem* volně v prostoru po dobu 5 s.
- Kalibrační matice se uloží na paměťovou kartu do souboru *kalibMAG.txt*. Pokud už na kartě soubor se stejným jménem existuje, bude přepsán.

A.8 Kalibrace gyroskopu

Pro kalibraci gyroskopu stiskněte *G* v kalibračním menu. Pokračujte podle následujících pokynů:

- Položte *Primer* na vodorovnou položku, stiskněte tlačítko a nechte jednotku v klidu po dobu 5 s.
- Stiskněte tlačítko a během nejvýše 5 s otočte *Primer* o $+90^\circ$ okolo osy X. K orientaci os a směrů rotace viz obr. 5.3.
- Stiskněte tlačítko a během nejvýše 5 s otočte *Primer* o -90° okolo osy X.
- Stiskněte tlačítko a během nejvýše 5 s otočte *Primer* o $+90^\circ$ okolo osy Y.
- Stiskněte tlačítko a během nejvýše 5 s otočte *Primer* o -90° okolo osy Y.
- Stiskněte tlačítko a během nejvýše 5 s otočte *Primer* o $+90^\circ$ okolo osy Z.
- Stiskněte tlačítko a během nejvýše 5 s otočte *Primer* o -90° okolo osy Z.
- Kalibrační matice se uloží na paměťovou kartu do souboru *kalibGYRO.txt*. Pokud už na kartě soubor se stejným jménem existuje, bude přepsán.

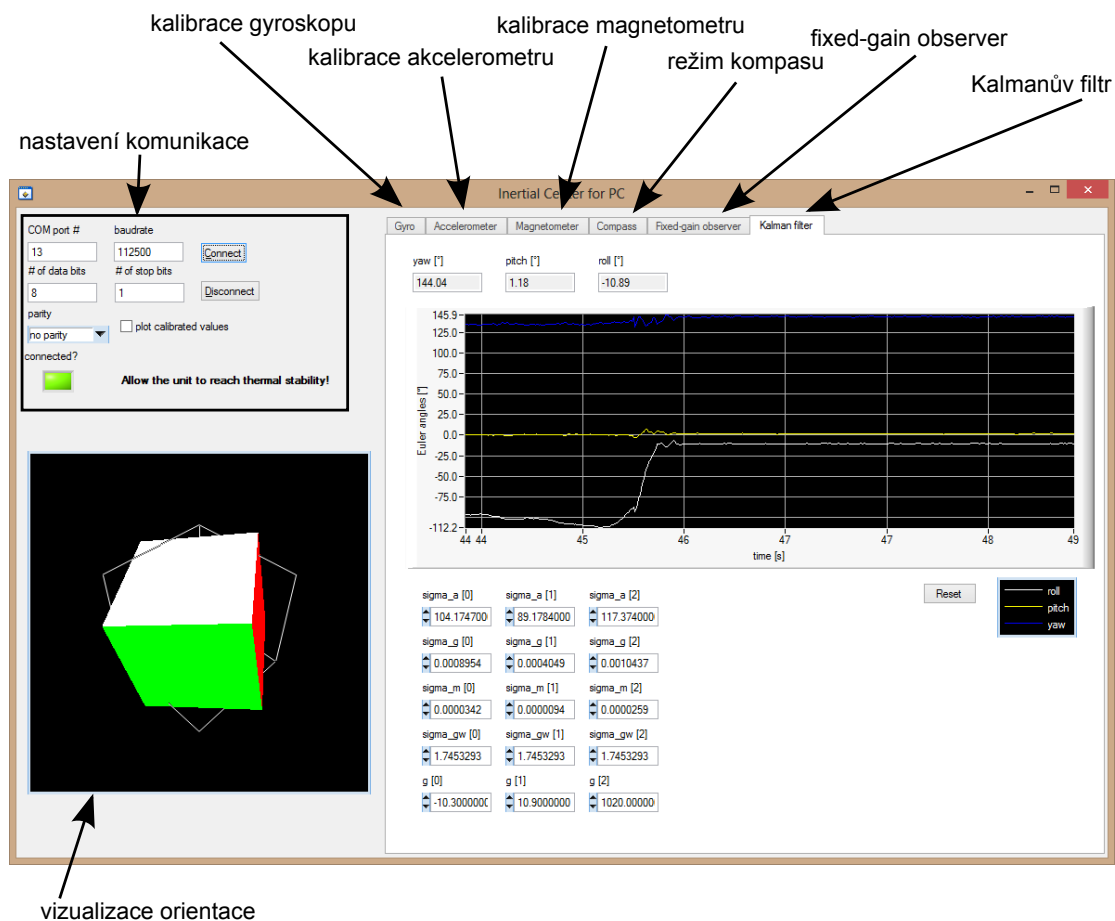
Příloha B

Inertial Center for PC

Program *Inertial Center for PC* vzniknul za účelem kalibrace IMU jednotky a implementace Kalmanovy filtrace v reálném čase.

Jak je vyznačeno na obr. B.2, *Inertial Center* se skládá z osmi hlavních částí:

1. nastavení komunikace s *Primerem*
2. vizualizace orientace pomocí 3D kostky
3. kalibrace gyroskopu
4. kalibrace akcelerometru
5. kalibrace magnetometru
6. režim kompasu
7. implementace algoritmu fixed-gain observer
8. Kalmanův filtr



Obrázek B.2. Základní popis Inertial Center for PC.

B.1 Komunikace

Inertial Center komunikuje s *Primerem* po sběrnici USB. Nejprve je nutné v *Primeru* zapnout aplikaci *Inertial Center 1.0.1* a propojit ho pomocí USB kabelu s počítačem. K tomu slouží na *Primeru* pravý USB port, označený jako „STM32“.

V levém horním rohu programu *Inertial Center for PC* je poté potřeba nastavit korektně komunikaci:

- COM Port # – nastavení čísla portu, ke kterému je připojen *Primer*. Lze ho snadno dohledat ve *Správci zařízení*.
- baudrate – nastavení znakové rychlosti. Nastavit na hodnotu 112500 Bd.
- # of data bits – počet datových bitů. Nastavit na 8 bitů.
- # of stop bits – počet stop bitů. Nastavit na 1 bit.
- parity – parita. Vybrat „no parity“.

Po správném nastavení, připojení *Primeru* a stisknutí tlačítka „Connect“ se zahájí přenos dat do aplikace. Funkční komunikaci indikuje LED označená „connected?“.

Před fyzickým odpojením *Primeru* je potřeba ukončit komunikaci. Provedete tak stiskem tlačítka „Disconnect“.

B.2 Vizualizace orientace pomocí 3D kostky

Levý dolní roh obrazovky zabírá 3D kostka využívající grafický engine OpenGL. Automaticky se zapíná při aktivních záložkách: „Gyro“, „Compass“, „Fixed-gain observer“ a „Kalman filter“ a natáčí se podle Eulerových úhlů vypočtených podle těchto metod. S kostkou lze pomocí myši volně rotovat a nastavit si ji tak, aby její orientace odpovídala skutečné orientaci *Primeru*.

B.3 Kalibrace gyroskopu

První záložka označená jako „Gyro“ slouží především ke kalibraci gyroskopu.

- Scale matrix, Offset matrix a Threshold matrix – odpovídají kalibračním maticím uvedeným v sekci 5.5.
- Values – zobrazuje naměřené nebo kompenzované hodnoty podle toho, jestli je vybrána možnost „plot calibrated values“ nacházející se v sekci nastavení komunikace.
- Threshold on/off – povoluje a zakazuje použití prahovacího algoritmu ze sekce 5.5.
- Calibrate X/Y/Z – zahajuje kalibraci příslušné osy gyroskopu podle zjednodušeného algoritmu popsaného v sekci 5.5. Při kalibraci postupujeme podle pokynů na obrazovce.
- Rotation span – umožňuje nastavit velikost úhlu, o který se otáčí jednotkou při kalibraci. Výchozí hodnota je 90°.
- Reset – nuluje úhly počítané z úhlové rychlosti.
- Model – zobrazí kompenzační model gyroskopu, se kterým program počítá.
- Load – načte kalibrační matice ze souboru.
- Save – uloží kalibrační matice do souboru.
- LP filtr – uplatní Butterworthův filtr 3. řádu (dolní propust) na přijaté úhlové rychlosti. Na výběr jsou zlomové frekvence 2 Hz, 5 Hz a 10 Hz.

B.4 Kalibrace akcelerometru

Druhá záložka označená jako „Accelerometer“ slouží především ke kalibraci akcelerometru.

- Scale matrix a Offset matrix – odpovídají kalibračním maticím uvedeným v sekci 5.3.
- Values – zobrazuje naměřené nebo kompenzované hodnoty podle toho, jestli je vybrána možnost „plot calibrated values“ nacházející se v sekci nastavení komunikace.
- Calibrate – zahajuje kalibraci podle algoritmu popsaného v sekci 5.3. Při kalibraci postupujeme podle pokynů na obrazovce. Po dokončení se spočtené koeficienty zobrazí v maticích Scale matrix a Offset matrix.
- Model – zobrazí kompenzační model akcelerometru, se kterým program počítá.
- Load – načte kalibrační matice ze souboru.
- Save – uloží kalibrační matice do souboru.
- LP filtr – uplatní Butterworthův filtr 3. řádu (dolní propust) na přijaté hodnoty zrychlení. Na výběr jsou zlomové frekvence 2 Hz, 5 Hz a 10 Hz.

B.5 Kalibrace magnetometru

Třetí záložka označená jako „Magnetometer“ slouží především ke kalibraci magnetometru.

- Scale matrix a Offset matrix – odpovídají kalibračním maticím uvedeným v sekci 5.4.
- Values – zobrazuje naměřené nebo kompenzované hodnoty podle toho, jestli je vybrána možnost „plot calibrated values“ nacházející se v sekci nastavení komunikace.
- Calibrate – zahajuje kalibraci podle algoritmu popsaného v sekci 5.4. Při kalibraci postupujeme podle pokynů na obrazovce. Po dokončení se spočtené koeficienty zobrazí v maticích Scale matrix a Offset matrix.
- Model – zobrazí kompenzační model magnetometru, se kterým program počítá.
- Load – načte kalibrační matice ze souboru.
- Save – uloží kalibrační matice do souboru.
- LP filtr – uplatní Butterworthův filtr 3. řádu (dolní propust) na přijaté hodnoty velikosti mag. indukce. Na výběr jsou zlomové frekvence 2 Hz, 5 Hz a 10 Hz.

B.6 Režim kompasu

Po vybrání záložky s označením „Compass“ se spustí algoritmus počítání orientace jednotky v prostoru za pomoci akcelerometru a magnetometru, viz sekce 6.5. Pro správnou funkci je nejprve potřeba správně nakalibrovat akcelerometr a magnetometr v příslušných záložkách nebo přímo v *Primeru*.

V polích yaw, pitch a roll se zobrazují vypočtené úhly, v grafu pak jejich časový vývoj.

B.7 Implementace algoritmu fixed-gain observer

Pátá záložka „Fixed-gain observer“ aktivuje výpočet orientace jednotky pomocí algoritmu popsaného v sekci 6.6. Pro správnou funkci je nutné mít nejdříve nakalibrovány všechny tři senzory.

Indikační prvky jsou shodné s těmi v záložce „Compass“. Přibyla ale možnost pomocí posuvného prvku „Gain“ nastavit zesílení L , viz sekce 6.6.

B.8 Kalmanův filtr

V rámci poslední, šesté, záložky s označením „Kalman filter“ je implementován algoritmus rozšířeného Kalmanova filtru. Pro správnou funkci je nutné nejprve nakalibrovat všechny tři senzory.

- pole yaw, pitch a roll – zobrazují vypočtené Eulerovy úhly.
- graf zobrazuje časový vývoj Eulerových úhlů.
- Reset – provede inicializační krok EKF. Vhodné například při rozkmitání filtru.
- sigma_a [0–2] – rozptyly zrychlení v jednotlivých osách.
- sigma_g [0–2] – rozptyly úhlové rychlosti v jednotlivých osách.
- sigma_m [0–2] – rozptyly mag. indukce v jednotlivých osách.
- sigma_gw [0–2] – prvky σ_{gki}^2 diagonální procesní kovarianční matice ${}^g\Sigma_k$, viz podsekcce 6.7.4.
- g [0–2] – prvky vektoru tíhového zrychlení v inerciálních souřadnicích. Implicitně v jednotkách mg .