ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Určení polohy objektu pomocí kombinace sensorů mobilního telefonu

květen 2014

Bakalant: **Filip Sušánek**

Vedoucí práce: **Ing. Bc. Marek Neruda**

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

Datum: 23. 5. 2014

.....

podpis bakalanta

Anotace

Tato bakalářská práce se zabývá problematikou inerciálních polohových systémů a možností jejich sloučení s daty z družicového polohového systému GPS a integrace v mobilním operačním systému Android. Předmětem výzkumu byla naměřená data získána pomocí moderního zařízení Google Nexus 7 s operačním systémem Android ve verzi 4.1.2.

Cílem práce bylo vyhodnotit výsledky měření poskytované gyroskopem a senzorem lineární akcelerace. Vytvořená aplikace kombinuje výstupní data z vytvořené inerciální polohové jednotky a změřenou polohu z GPS pro zvýšení přesnosti zaznamenané trasy.

Klíčová slova

Android, inerciální navigační systém, GPS, gyroskop, lineární akcelerace, senzorová fúze

Abstract

This bachelor thesis covers the topic of Inertial Navigation Systems and their possible fusing with data from global positioning system GPS in Android operating system. Goal was to research possible use of linear acceleration sensor in device Google Nexus 7 for implementation of inertial measurement unit.

During research I developed an Android application dedicated for process of acquiring sensor data and their mutual fusion in order to achieve higher accuracy in determining user's position.

Klíčová slova

Android, Inertial Navigation System, GPS, gyroscope, linear acceleration sensor, sensor fusion

Obsah

1 Úvod

Tato bakalářská práce se zabývá problematikou určování polohy zařízení pomocí kombinace inerciálního a globálního pozičního systému GPS v systému Android. Na základě konzultací s vedoucím práce byly stanoveny následující body práce. Hlavní náplní bylo osvojení problematiky inerciální navigace a senzorové fúze, osvojení základů programování pro systém Android, vytvoření demonstrační lokalizační aplikace a prozkoumání možnosti využití vytvoření vlastní implementace *Inertial Measurement Unit*, neboli inerciální poziční jednotky, do systému pomocí dostupných senzorů lineární akcelerace a gyroskopu.

Většina moderních inerciálních polohových jednotek se skládá ze senzorů měřící zrychlení, neboli akcelerometrů, senzorů měřících úhlovou rychlost pro výpočet orientace systému a doplňujících senzorů pro kompenzaci chyb a určení referenční pozice.

Motivací pro výběr této problematiky byly vlastní zkušenosti s různými aplikacemi zaznamenávající trasu pro systém Android. Vzhledem k tomu, že všechny tyto aplikace využívají k určení polohy výhradně data z GPS, již z podstaty fungování globálního pozičního systému nelze dosáhnout vysoké přesnosti a věrnosti dat odpovídající skutečnému pohybu. Určité zpřesňující algoritmy jsou implementovány například v polohových aplikacích využívaných v automobilové dopravě. Aplikace využívá dat z implementovaného senzoru akcelerace pro úpravu rychlosti v případě ztráty družicového signálu.

Při návrhu vlastní inerciální polohové jednotky v systému Android jsem se pokusil prozkoumat výhody a možnosti již implementovaného senzoru lineární akcelerace. Dále bylo nutné prozkoumat vlastnosti výstupních dat z gyroskopu a možnosti filtrace. Na základě analýzy byl v prototypu aplikace implementován algoritmus vypočítávající polohu zařízení.

V dalším kroku byla výstupní data z jednotky IMU slučována s daty poskytnutými senzorem GPS pro výpočet doplňujících bodů změřené trasy.

Vytvořený prototyp aplikace je schopen měřit a ukládat celou škálu dat odvozených z nefiltrovaných dat senzoru lineární akcelerace, gyroskopu a GPS v různých formátech.

Práci lze dle jejího obsahu rozdělit na tři části: teoretickou, praktickou část a závěr. První část se zabývá seznámením se systémem Android, jeho historií a strukturou aplikací. V dalších kapitolách je detailně rozebrána problematika určování polohy zařízení pomocí vnější reference a sledování změn vnitřního stavu. Problematika inerciální navigace obsahuje historii, principy realizace a použité senzory. V neposlední řadě lze v teoretické části nalézt kapitolu věnované problematice senzorové fúze a příklady využití inerciálních polohových systémů v reálném životě.

Praktická část obsahuje popis prototypu první měřící aplikace, návrhy a implementace algoritmů a popis vyvinuté lokalizační aplikace. Dále obsahuje analýzu a metody filtrování výstupních dat ze senzorů a navržené filtry. V příslušné kapitole je nastíněn průběh výpočtu pozice z inerciální jednotky a fúzování s daty z GPS.

V závěru práce jsou shrnuty a diskutovány dosažené výsledky a návrhy dalšího rozšíření práce.

2 Teoretický rozbor

2.1 Android

2.1.1 Historie operačního systému Android

Počátky vývoje operačního systému Android sahají již do roku 2003, kdy byla v říjnu založena společnost Android Inc. Zakladateli byla čtveřice lidí v čele s Andy Rubinem [1]. Dalšími zakladateli byli Rich Miner, Nick Sears a Chris White. Společnost se zabývala vývojem softwaru pro mobilní telefony a digitální fotoaparáty. Vývojáři pracovali v zásadě bez zájmu veřejnosti a výsledky své práce nijak veřejně neprezentovali [2].

V srpnu 2005 společnost Android Inc. kupuje Google a udělá z ní jednu ze svých dceřiných společností [2]. Celý vývojový tým přechází pod křídla tohoto giganta a šéfem vývoje zůstává Andy Rubin, který tuto pozici zastává až do roku 2013, kdy se v rámci společnosti Google začne věnovat jiným projektům[3].

V roce 2007 získal Google několik patentů v segmentu mobilních přístrojů a odborná veřejnost začala spekulovat, zda-li se nechystá vstoupit na trh „chytrých" mobilních telefonů [4]. Za chytré zařízení se označuje to, které využívá pokročilý operační systém umožňující instalaci aplikací přinášejících doplňkové funkce a úpravu systému.

Na začátku listopadu téhož roku, konkrétně 5. listopadu, byla vytvořena tzv. Open Handset Alliance, zkráceně OHA [5]. Jde o alianci společností zabývajících se výrobou hardwaru a softwaru pro mobilní zařízení. V současnosti má aliance několik desítek členů, mezi něž patří například mobilní operátoři, výrobci čipů, výrobci mobilních telefonů či softwarové společnosti [5]. Cílem sdružení bylo vyvinout otevřený standard pro mobilní zařízení.

Ve stejný den byl oznámen první společný projekt OHA a Googlu, linuxový operační systém Android [6]. Výkonný ředitel Googlu Eric Schmidt ve svém prohlášení uvedl, že představují mobilní systém budoucnosti, který má přinést svěží vítr do mobilního segmentu a věří, že v budoucnu bude tisíce různých zařízení poháněno právě tímto systémem [7]. Ve stejném měsíci byl uvolněn první *Android Software Development Kit* (*SDK*) pro vývojáře mobilních aplikací.

O rok později, v říjnu 2008, došlo k uvedení prvního telefonu poháněného Androidem ve verzi 1.0 a uveřejnění systému jako *open-source* [4]. Telefon vyrobila tchajwanská společnost HTC a prodával se pod označením HTC Dream. Známější je však brandovaná verze T-Mobile G1.

Aktualizace Androidu na verzi 1.5 označovaná jako *Cupcake* nastartovala zájem široké veřejnosti o nový systém a sama je někdy označována za první plnohodnotnou verzi systému [1].

Další úspěšným telefonem bylo zařízení od společnosti Motorola, Motorola Droid z roku 2009. Uveden byl společně s novou verzí Androidu 2.0 označovanou jako *Eclair*. Telefon zaznamenal velký úspěch a započal tažení proti společnosti Apple a jejich iPhony běžících na vlastním systému iOS [1].

Prvním vlastním telefonem se Google pyšnil začátkem roku 2010. Byl prvním v produktové řadě Nexus a označoval se jako Nexus One. Bohužel vysoká cena stanovená na 530$ zapříčinila, že po šesti měsících byl prodej zastaven [1].

Naštěstí se Google počátečním neúspěchem nenechal odradit a ve spolupráci se společností Samsung představil Nexus S, který byl v podstatě klonem úspěšného modelu Samsung Galaxy S, lišil se pouze designem. Byl představen zároveň s novou verzí systému označovanou jako *Gingerbread*, číselně 2.3 [1]. Tato verze systému se rychle stala velmi oblíbenou a dominovala trhu ještě v roce 2012 [8].

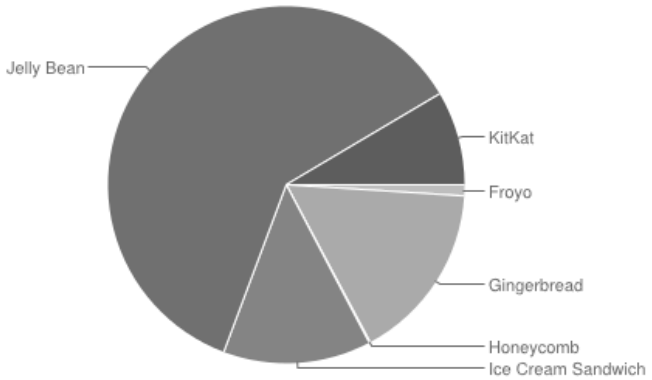
Rok 2010 byl také rokem, kdy podíl Androidu na trhu překonal podíl iOS a iPhonu v USA. Podíl systému od společnosti Blackberry byl překonán na jaře následujícího roku a od té doby Android dominuje americkému trhu [1].

Nejvýraznějším posunem ve vývoji systému byla beze sporu aktualizace označována jako *Ice Cream Sandwich* (4.0). Přinesla sjednocení telefonní a tabletové verze systému, vylepšené GUI a multitasking [4]. Předcházející čistě tabletová verze označována jako *Honecomb* (3.0) se ve větší míře neprosadila a brzo byla nahrazena právě „zmrzlinovým sendvičem".

Další představenou verzí systému byl tzv. *Jelly Bean* (4.1). Tato aktualizace přinesla rychlejší grafické prostředí a celkovou optimalizaci běhu systému [4].

Navzdory každoročnímu představování nových verzí se výrobci hardwaru zrovna nepředháněli ve vytváření aktualizací pro již existující zařízení. Jak jinak vysvětlit fakt, že dva měsíce po představení *Jelly Beanu*, byla stále dominantní verzí systému *Gingerbread*, představený téměř o dva roky dříve. *Gingerbread* měl 60% podíl mezi zařízeními běžící na Androidu, *Ice Cream Sandwich* po roce života jen 16% a čerstvě představený *Jelly Bean* pouhých 0,8% [8].

Dnes je situace výrazně lepší, a přestože nedávno došlo k vydání další verze označované jako *KitKat* (4.4), podíl zařízení běžících na *Jelly Beanu* je již přes 50% a *Gingerbread* má pouhých 10% [9].



Obrázek 1: Poměr verzí systému Android v zařízeních ke dni 22. 5. 2014, převzato z [9]

V roce 2013 Android již dominuje trhu s mobilními zařízeními. Android v současnosti ovládá téměř 80 % trhu, zatímco hlavní konkurent iOS od Applu pouhých 13 %. Třetí nejvýraznější platforma Windows Phone patří pouhá necelá 4 % světového trhu a dříve velmi oblíbený systém Blackberry používá jen necelá 3 % segmentu [10].

Top Smartphone Operating Systems, Shipments, and Market Share, 2013 Q3 (Units in Millions)

Operating System	2Q13 Unit Shipments	2Q13 Market Share	2Q12 Unit Shipments	2Q12 Market Share	Year-over-Year Change
Android	187.4	79.3%	108	69.1%	73.5%
iOS	31.2	13.2%	26	16.6%	20.0%
Windows Phone	8.7	3.7%	4.9	3.1%	77.6%
BlackBerry OS	6.8	2.9%	7.7	4.9%	-11.7%
Linux	1.8	0.8%	2.8	1.8%	-35.7%
Symbian	0.5	0.2%	6.5	4.2%	-92.3%
Others	N/A	0.0%	0.3	0.2%	-100.0%
Total	**236.4**	**100.0%**	**156.2**	**100.0%**	**51.3%**

Source: IDC Worldwide Mobile Phone Tracker, August 7, 2013

Obrázek 2: Podíl operačních systémů v segmentu mobilních zařízení k 7. srpnu 2013, převzato z [10]

2.1.2 Architektura systému Android

Operační systém Android je rozdělen do vrstev jako všechny linuxové systémy. Vrstev je zde celkem pět. Každá z nich plní určitou funkci důležitou pro chod celého systému [11].

Nejnižší vrstvou je kernel, neboli jádro systému, které je založeno na kernelu dospělého operačního systému Linux. Tato vrstva zprostředkovává komunikaci mezi hardwarem zařízení a softwarem, který na něm běží. Zajišťuje například správu paměti, sítě, ovladače nebo například přidělování procesů procesoru. Implementace této vrstvy obecně zvyšuje stabilitu a odolnost systému [12].

Druhou vrstvou jsou knihovny. Ty obsahují funkce pro práci s různými komponentami systému od přehrávání médií, přes knihovny prohlížeče až po vykreslování grafiky. Kód většiny knihoven je psán v C nebo C++ [12]

Třetí vrstva se nazývá Android Runtime a obsahuje virtuální stroj pro spouštění aplikací napsaných v jazyce Java. *Runtime* se označuje jako *DVM*, neboli *Dalvik Virtual Machine*. Tento stroj je unikátní pro Android, neboť knihovny původního virtuálního stroje *Java Virtual Machine* (*JVM*) nejsou volně šiřitelné a proto byl vytvořen *Dalvik*. Každá spuštěná aplikace běží ve vlastním procesu s vlastní instancí stroje *Dalvik* [12].

Čtvrtou vrstvu tvoří aplikační rámec, neboli *Application Framework*. Tato vrstva je z pohledu vývojáře nejdůležitější. Poskytuje přístup k službám a funkcím, které mohou poté zpřístupnit ve své aplikaci. Patří sem například knihovny pro vytváření uživatelského rozhraní, přístup k senzorům, řízení životního cyklu aplikace a práce se soubory [12].

Poslední a nejvyšší vrstvou jsou samotné aplikace spouštěné uživatelem. Zde již běžný uživatel vidí jen hotovou aplikaci a provádí s ní interakci. V zájmu co největší stability a zabezpečení, každá aplikace běží ve svém vlastním procesu s unikátním identifikátorem a vlastní instancí virtuálního stroje *Dalvik* [13]. Díky tomuto způsobu spouštění má aplikace přístup pouze k součástem systému, které jsou nezbytně nutné pro korektní běh aplikace. Toto omezení lze však v určitých případech obejít [13].

Obrázek 3: Struktura systému Android, převzato z [11]

2.1.3 Architektura a struktura aplikací pro systém Android

Vývoj aplikací běžících v systému Android může probíhat v několika jazycích. Jde o Javu, C a C++. Nejrozšířenější způsob vytváření programů je pomocí jazyka Java a na něj navázaného Android *SDK* [13]. Nicméně Google nabízí i kompilační nástroj Android *NDK* schopný vytvářet spustitelné programy i v jazyce C a C++. Jde však pouze o okrajový způsob vývoje a spíše než k tvorbě celých aplikací se Android *NDK* využívá pro překlad již vytvořených knihoven [14].

V této práci se zabývám vývojem v jazyce Java pomocí vývojového prostředí *Eclipse IDE*, neboť již vytvořené *API* značně ulehčuje celý proces.

Instalace aplikace probíhá pomocí archivačního balíku s příponou *apk*. Tento archiv obsahuje všechny potřebné komponenty pro samotný běh aplikace [13].

Mezi základní součásti funkční aplikace patří:

- *Aktivities* (aktivity)
- *Services* (služby)
- *Content Providers* (poskytovatelé obsahu)
- *Broadcast Receivers*

Aktivita odpovídá jedné obrazovce uživatelského prostředí zobrazené uživateli. Pro vytvoření uživatelského rozhraní (*UI*) aktivita načítá layout definovaný v *xml* souboru. Aplikace obvykle obsahuje alespoň dvě aktivity, výchozí obrazovku s informacemi a nastavení aplikace. Mezi aktivitami může uživatel pomocí prvků uživatelského rozhraní přepínat. Při přepínání obrazovek si lze předávat data pomocí tzv. *Intents*. Přestože vícero aktivit společně vytváří kompletní aplikaci, jsou na sobě nezávislé. O životní cyklus aktivit se stará *Activity Manager* [13].

Service, je služba představují pouze proces běžící na pozadí bez uživatelského rozhraní. Využívá se k úkonům, které jsou náročné na výpočet nebo pro přístup ke vzdáleným

zdrojům, kde není jistá okamžitá odezva. Služba se spouští dvěma způsoby, kdy ji komponenta aplikace spustí v režimu závislosti či nezávislosti. Režim nezávislosti je vyvolán metodou *startService*. Služba je spuštěna nezávisle na ostatních částech aplikace a musí být ukončena metodou *stopService*. Režim závislosti se vyvolá pomocí metody *bindService*, kdy se služba „sváže“ s volající komponentou. V tomto případě službu může ukončit pouze svázaná komponenta. Službu lze svázat s více klienty a k ukončení dojde po odpojení posledního klienta [13].

Content Providers, neboli poskytovatelé obsahu, poskytují *interface* pro přístup k datům aplikace nebo ke sdílení dat mezi ostatními součástmi systému. Poskytovatel obsahu zároveň určuje, zda k datům aplikace budou mít přístup i jiné aplikace. Data můžou být uložena ve formě souborů na disku, *SQL* databáze, na webu, nebo kdekoliv, kam bude mít aplikace přístup. Oddělení dat od aplikací pomocí *content provideru* umožňuje například nahradit výchozí aplikace systému aplikacemi novými [13].

Broadcast Receiver, neboli „naslouchač“, je komponenta aplikace, která vytváří rozhraní pro zpracování upozornění a *intentů* systému a podle nastavení na ně reaguje například spuštěním nějaké komponenty aplikace. Stejně jako služby a poskytovatelé obsahu nemá *broadcast receiver* uživatelské rozhraní [13].

Intent je vyslaný požadavek nebo zpráva oznamující událost, která při zachycení odpovídající komponentou vykoná určitou akci. Může sloužit i k předání dat mezi komponenty aplikace [12].

2.2 Určování polohy zařízení

Již od nepaměti měl člověk potřebu orientace v prostoru a určování polohy. Vývoj probíhal od učení světových stran podle Slunce, přes vynález kompasu až po dnešní nejmodernější způsob určení polohy pomocí satelitů.

Určování polohy zařízení, potažmo člověka, lze nejobecněji rozdělit na dva způsoby. Zjištění polohy na základě vnější reference a zjištění polohy na základě změny vnitřního stavu objektu [15].

Způsoby určení polohy na základě vnější reference:

- *Podle orientačních bodů v okolí (Pilotage)*
  - Orientace probíhá díky určení polohy orientačních bodů a své pozice vzhledem k nim.
  - Jde o způsob starší než samo lidstvo, neboť je známo, že i divoká zvířata se tímto způsobem orientují.
- *Na základě známého kurzu a předchozí pozice (Dead reckoning)*
  - Při učení polohy se spoléhá na to, že předchozí pozice je známa a lze určit směr a odhadnout rychlost.
  - Ve středověku využití v mořeplavectví.
- *Navigace podle hvězd (Celestial navigation)*
  - Neboli astronavigace, jde o určení vlastní polohy na základě doby měření a úhlu určitých nebeských těles vzhledem k horizontu v daném čase.
  - Využití převážně v mořeplavectví.
- *Na základě rádiových signálů*
  - Určení polohy probíhá na základě známé pozice zdroje rádiové frekvence, např. radiomaják, a měří se doba od vyslání signálu.
  - Tímto způsobem lze změřit pozici, směr a pomocí Dopplerova efektu i rychlost.
  - Do této skupiny patří globální družicové polohové systémy (GNSS), tedy GPS, GLONASS, Galileo a jiné.

Jako způsob určení polohy bez vnější reference se označuje:

- *Inerciální navigace*
  - Jde o způsob určení polohy pomocí měření změny stavu vlastního zařízení vůči výchozí pozici.

2.2.1 Global Positioning System

Americký poziční systém GPS je jedním ze systémů patřících do skupiny *Global Navigation Satellite System* (*GNSS*), neboli globální navigační satelitní systémy. Další platformy satelitních pozičních systémů vznikly pod záštitou jiných států či organizací. Hlavními zástupci jsou ruský GLONASS *(Global Orbiting Navigation Satellite System),* jehož počátky sahají do éry studené války a existence SSSR, čínský Compass (známý také pod svým kódovým označením *Beidou-2*) a evropský navigační systém Galileo. Projekt Galileo je zaštiťovaný Evropskou unií a Evropskou vesmírnou agenturou ESA [15].

Všechny tyto systémy operují na podobných principech. Jde o vypuštění většího počtu družic, které se pohybují po předem určených drahách. V systému GPS není žádná z jeho 32 družic geostacionární, na rozdíl od jiných *GNSS* [15].

Systém GPS byl původně plánován pro provoz 24 družic, ale dnes jich je již 32. Satelity se pohybují po šesti orbitálních drahách, na kterých jsou rovnoměrně rozmístěny. Orbita má přibližně kruhový tvar se sklonem *55°* k rovníku a jsou od sebe vzájemně vzdálené o *60°* zeměpisné délky. Doba oběhu jednoho satelitu je přibližně dvanáct hodin. Dráhy a satelity na nich jsou vypočteny tak, aby alespoň tři a více družic bylo viditelných v jakémkoliv okamžiku z jakéhokoliv místa na Zemi [16].

Každá družice nese své atomové hodiny pro přesné měření času, který je pro správnou funkčnost systému klíčový. Celý systém družic a pozemních stanic a přijímačů musí být synchronizován s co největší přesností. Satelity komunikují mezi sebou, dohledovým centrem a uživateli. Řídící a dohledová centra jsou rozmístěny po zeměkouli a jejich pozice je přesně známa. Jejich úkolem je sledovat pozici satelitů a synchronizaci hodin v síti. V případě ukončení činnosti tohoto dohledového centra, je systém schopen fungovat ještě po dobu následujících šesti měsíců [16]

Princip určení polohy funguje tak, že družice vyšle signál, jenž obsahuje časovou značku a informace o poloze satelitu, z nichž přijímač vypočte polohu satelitu v době vyslání a dobu zpoždění přijetí signálu. Z těchto informací je vypočten poloměr kružnice s poloměrem rovným vzdálenosti přijímače od družice. Pro určení přesné pozice na Zemi stačí teoreticky signál ze tří družic, ale v zájmu zvýšení přesnosti se jich použije více [15].

Přesnost určení pozice je ovlivněna mnoho faktory. Mezi hlavní patří množství viditelných satelitů, přesnost hodin přijímače a meteorologické jevy v atmosféře [15].

2.2.2 Inerciální navigace

Inerciální navigace umožňuje odhadovat polohu zařízení bez použití vnější reference. Využívá vestavěné senzory pohybu a rotace na určení orientace, zrychlení a přítomnosti magnetického pole. Zařízení obsahující tyto senzory a vykonávající navigaci se označuje jako inerciální navigační jednotka, neboli IMU *(Inertial Measurement Unit)*. Navigační inerciální systémy se označují INS *(Inertial Navigation System)* [17].

Navigace obvykle vychází z předpokladu, že výchozí podmínky, jako poloha, rychlost a orientace jsou známé. Výchozí polohu lze získat pomocí jedné nebo více referenčních jednotek, například pomocí *GPS*.

Obrovská výhoda tohoto způsobu spočívá v tom, že je nezávislá na vnějších podmínkách, tudíž nelze „zmást“ rušením, radiací, či nepříznivými meteorologickými podmínkami. Nevýhodou je, že s časem roste nepřesnost systému. Malé chyby v navigaci se s časem sčítají a způsobují větší chyby. Tato chyba se nazývá *drift*. Vojenské námořní systémy patřící k nejdokonalejším *INS* také trpí *driftem* rovnajícím se *1,8 km* za den. To znamená, že pokud by zařízení stálo v naprostém klidu, tak kvůli nepřesnostem v kalibraci a měření bude změřená poloha po 24 hodinách od skutečné pozice vzdálená o *1,8 km*. Takovouto chybu ovšem není těžké kompenzovat na základě pravidelných aktualizací polohy pomocí dalších navigačních systémů, jmenovitě např. *GPS* [15].

INS dříve převážně využívaly vojenské složky ve svých vozidlech, plavidlech a letadlech [15]. Ovšem díky neustálému pokroku v oblasti miniaturizace došlo ke zlevnění výroby potřebných senzorů do takové míry, že v dnešní době velká většina chytrých

mobilních telefonů tyto senzory také obsahuje. Jen chybí komplexní softwarová implementace, která ovšem není jednoduchá [18].

2.2.2.1 Historie inerciální navigace

Inerciální navigace má za sebou poměrně krátký, ale o to intenzivnější vývoj. Počátek těchto navigačních systémů je svázán s vývojem raketových zbraní dlouhého doletu během druhé světové války [18]. Ovšem první aplikace gyroskopů jako inerciálních stabilizačních systémů se datuje do 20. let 20. století [15]. Šlo o použití gyroskopů k ovládání torpéd a v letectví o reprezentaci horizontu. Tyto systémy byly navrženy americkým inženýrem Elmerem Sperrym [15].

Ovšem pravé využití našly *IMU* v raketových zbraních. Vývojem systému řízení a navigace raket během druhé světové války se na americké straně zabýval americký inženýr Robert Goddard a na německé straně německý inženýr Wernher von Braun [18].

Německé rakety V-2 byly vybaveny jednoduchým analogovým počítačem, dvěma gyroskopy a akcelerometrem. Dva gyroskopy, horizontální a vertikální, sloužili pro boční stabilizaci rakety a akcelerometr integrující zrychlení po dosažení určité rychlosti odpojil motor [15].

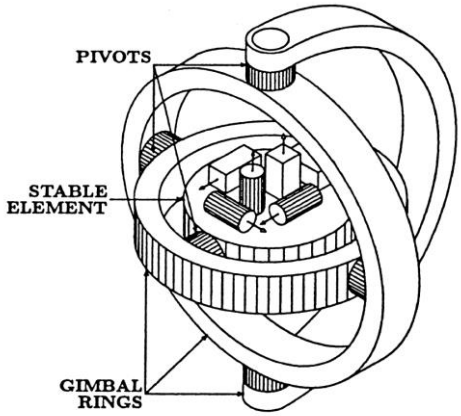
Další nekomerční uplatnění nalezly INS v kosmonautice. V 60. letech NASA během programu Apollo spolupracovala na navigačním systému s MIT *(Massachusetts Institute of Technology)* [19] a divizí General Motors zvanou Delco. Výsledkem této spolupráce bylo vytvoření *IMU*, které pomáhaly s navigací řídícího a lunárního modulu [19].

S postupem miniaturizace došlo k implementaci těchto senzorů i do mobilních zařízení. Jejich primárním účelem je sledovat natočení zařízení a uzpůsobovat tomu uživatelské rozhraní. Další využití nacházejí ve hrách nebo v měřících aplikacích.

2.2.2.2 Způsob realizace IMU

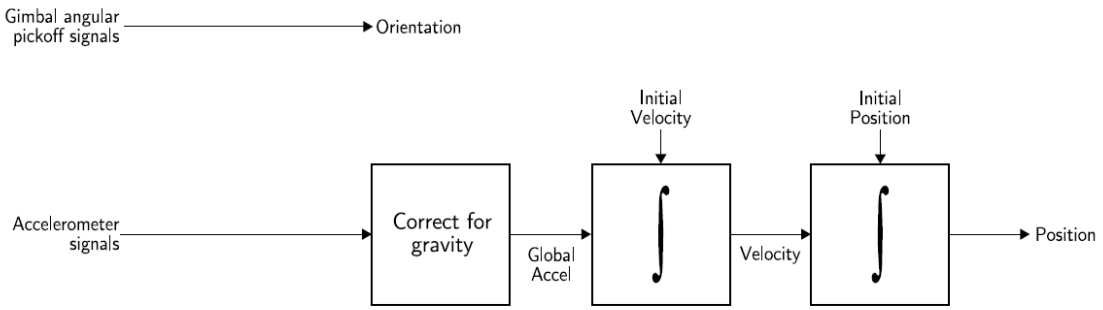
Jednotky *IMU* podle způsobu své konstrukce využívají dva principy pro zpracování polohy. Rozlišujeme, zda je senzorová část připevněna pevně k měřenému objektu, nebo je v konstrukci umožňující volný pohyb. Odborné termíny pro tyto soustavy jsou *Gimbaled INS* a *Strapdown INS* [20].

Gimbaled INS, neboli *INS* se stabilní platformou, je starší a dříve hojně využívaný typ *INS*. Jde o mechanické zařízení, jež ve svém středu obsahuje stabilní měřicí senzorovou jednotku. Díky této konstrukci na senzory v jejím středu nepůsobí rotační pohyb objektu, a proto akcelerometry měří zrychlení vzhledem k Zemi. Měří tedy v globálním systému souřadnic, kde osa *Y* míří na sever, osa *X* na východ a osa *Z* je kolmá k povrchu Země a míří vzhůru [21].



Obrázek 4: Konstrukce Gimbaled INS, převzato z [15]

První takovéto zařízení se používaly již při vynálezu gyroskopu, kdy sloužili pro izolaci gyroskopu během rotace připevněné podložky. Mezi nevýhody těchto systémů patří zejména velikost, náročnost výroby a tím způsobená vysoká cena. Výhodou je naopak pouze nízký výpočetní výkon potřebný pro zpracování dat. Až do 70. let, kdy došlo k vynálezu *INS* s pevnou montáží, se používaly výhradně mechanické systémy. V 50. letech se tyto systémy využívaly v raketových zbraních, ponorkách, letadlech a lodích americké armády a někde stále slouží jejich modernější verze [15].

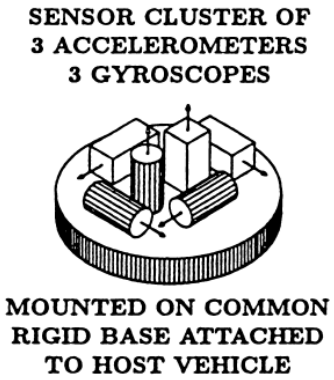


Obrázek 5: Postup měření pozice v Gimbaled INs, převzato z [20]

Je třeba zmínit, že tyto systémy mohou trpět jevem zvaným *Gimbal Lock*. Jde o jev, kdy je ztracena informace o jedné ose v třídimenzionálním prostoru. Tato situace nastává, pokud dvě osy v systému mají takovou polohu, že měří stejnou veličinu. Podrobněji je *gimbal lock* vysvětlen v části o Eulerovo úhlech [21].

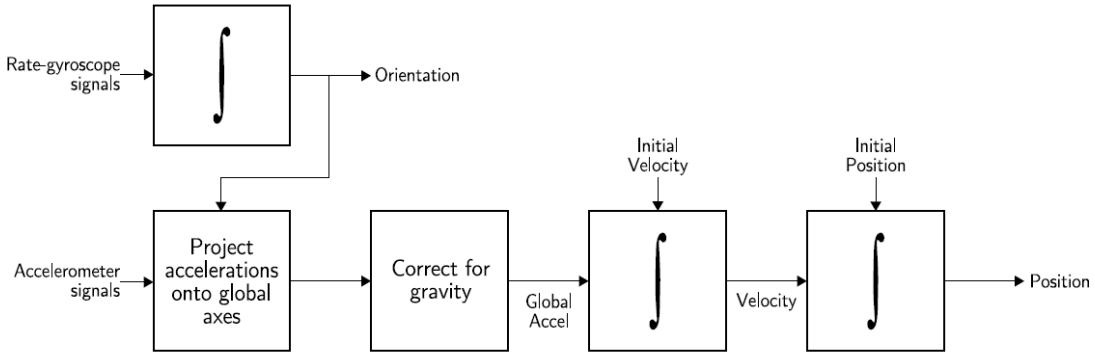
Strapdown INS, neboli k objektu pevně připevněné *INS*, vznikly začátkem 70. let a díky miniaturizaci došlo ke snížení ceny a rozšíření i do komerční sféry. Vzhledem k tomu, že v tomto způsobu je snímací platforma pevně připevněna k pohybujícímu se objektu, senzory měří změny natočení a zrychlení vzhledem k výchozí pozici tělesa. Neměří skutečný pohyb po Zemi, ale pohyb vzhledem k výchozí pozici. Měří tedy v lokální soustavě souřadnic.

Systém vyžaduje více výpočetního výkonu, než *Gimbaled INS*, neboť je třeba naměřený pohyb přepočítat do globální soustavy souřadnic. Pomocí údajů z gyroskopu je vypočtena orientace objektu a pomocí ní se přepočte naměřené zrychlení na zrychlení vůči Zemi [20].



Obrázek 6 Konstrukce Strapdown INS, převzato z [15]

Výhodou těchto systémů je menší velikost, nižší cena a spolehlivost. Naopak nevýhodou je potřeba vyššího výpočetního výkonu pro zpracování vstupních dat, což ovšem v dnešní době již není výraznější problém [15].



Obrázek 7: Postup měření pozice v Strapdown INs, převzato z [20]

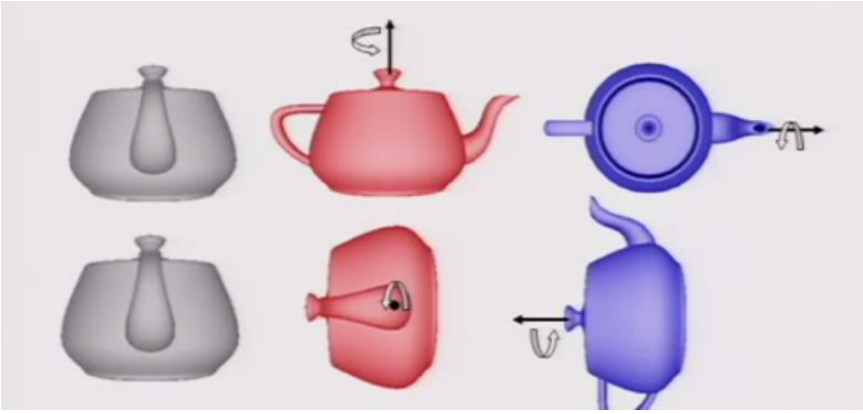
2.2.2.3 Způsoby výpočtu orientace objektu v strapdown systémech

Předně při určování orientace je třeba si uvědomit, že pro orientaci neplatí komutativita jako pro zrychlení. Zrychlení je totiž vektor a výsledný vektor je součtem jeho složek. Pro jeho získání nezáleží na pořadí kroků, viz ilustrace. Toto ovšem neplatí pro rotaci.

Natáčíme-li objekt postupně podél osy *Z* a *Y* o +90 stupňů, skončíme s jinou orientací, než kdybychom rotaci prováděli v pořadí os Y a *Z*. Tento jev je tím výraznější, s čím většími úhly operujeme [22].

Komutativita zrychlení:

$$a = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{bmatrix} = \begin{bmatrix} b_1 + a_1 \\ b_2 + a_2 \\ b_3 + a_3 \end{bmatrix} \quad (1)$$



Obrázek 8: Příklad vlivu pořadí rotace, převzato z [22]

Existuje několik způsobů výpočtu orientace objektu v prostoru. Lze použít jednu z těchto metod: *změna úhlů, Eulerovy úhly, quaterniony* nebo *rotační matice* [22]. Každá z nich má své výhody a nevýhody. Podrobněji jsem ve své práci zkoušel pracovat s Eulerovými úhly a *quaterniony*.

2.2.2.3.1 Změna úhlů

Pomocí této metody se měří změna natočení okolo každé osy od předchozí pozice. Díky měření pouze malých změn úhlů vzniká pouze malá chyba při rotaci podél každé z os. Ovšem z dlouhodobého hlediska se chyby kumulují, proto tento způsob není vhodný pro přesné aplikace [22].

2.2.2.3.2 Eulerovy úhly

Zavádí pojmy rotace okolo každé osy, kdy rotace okolo osy *Z* je *Yaw* (neboli azimut), okolo osy *X Pitch* (neboli elevace) a okolo osy *Y Roll* (neboli natočení). Pozitivní směr otáčení je proti směru hodinových ručiček. Soustava souřadnic je pravotočivá a osy jsou uspořádány tak, že osa *X* prochází bokem zařízení, osa *Y* předkem a osa *Z* horem zařízení či objektu [22].

Jak je výše zmíněno, pro určení orientace zařízení nezáleží jen na rotaci podél os, ale zároveň na jejich pořadí. Proto je pořadí rotace definováno. Pro rotaci z lokálního systému souřadnic do globálního systému je pořadí rotací dáno jako *yaw*, *pitch* a *roll*. Při opačné rotaci postupujeme v opačném pořadí [17].

Významnou nevýhodou tohoto způsobu určení orientace je, že zde může nastat jev zvaný *Gimbal Lock*. Stane se tak v případě, že rotace okolo jedné osy bude blízká *90°*, potom vyvstane jev, že dvě osy budou měřit stejnou veličinu [22]. Lze si to představit tak, že *roll* (rotace podél osy *X*) bude *90°*. Pak osa měřící *pitch* (elevace na ose *Y*) je na místě původní osy *Z* měřící *yaw* (azimut). V této situaci přestane být jasné, která těchto os měří správně jakou hodnotu. S tímto je třeba počítat a přizpůsobit případnou aplikaci [17].

2.2.2.3.3 Quaterniony

Pojmem *quaternion* je myšlen 4-dimenzionální vektor, jehož složky jsou rovny:

$$q = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \cos\frac{\varphi}{2} \\ x * \sin\frac{\varphi}{2} \\ y * \sin\frac{\varphi}{2} \\ z * \sin\frac{\varphi}{2} \end{bmatrix} \tag{2}$$

První složka *a* značí úhel, o jaký je třeba soustavu otočit. Další tři složky definují vektor, okolo kterého se bude soustava otáčet [17]. Fungují na myšlence, že transformace jakékoliv jedné orientace ve druhou lze provést pouze jednou rotací podél vektoru představující novou osu rotace [22]. *Quaternion* lze vyjádřit jako součet reálné části *a* a imaginárních částí *b,c,d*.

$$q = a + ib + jc + kd \tag{3}$$

Vektor $a^b = (x, y, z)$ v lokální soustavě souřadnic může být vyjádřen jako:

$$a_q^b = 0 + ix + jy + kz \tag{4}$$

Převedení do globální soustavy souřadnic se vypočte pomocí Hamiltonova produktu [23] jako:

$$a^g = q * a_q^b * q^* \tag{5}$$

kde q^* je komplexně sdružené číslo k číslu q.

Způsob výpočtu orientace pomocí *quaternionu* netrpí fenoménem *gimbal lock,* neboť zde měříme pouze jednu rotaci. Další výhodou je, že výpočet je méně náročný na výpočetní výkon než výpočet pomocí rotačních matic.

2.2.2.3.4 Rotační matice

Převod orientace z lokální do globální soustavy souřadnic lze provést i pomocí rotačních matic. Jako v případě Eulerových úhlů jde o tři po sobě jdoucí rotace, kde každý Eulerův úhel je reprezentován jednou maticí [17].

Matice reprezentující *yaw* (azimut), neboli rotaci okolo osy *Z*:

$$C_1 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Matice reprezentující *pitch* (elevaci), neboli rotaci okolo osy *Y*:

$$C_2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta\ 0 & 0 & \cos\theta \end{bmatrix} \tag{7}$$

Matice reprezentující *roll* (natočení), neboli rotaci okolo osy *X*:

$$C_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{8}$$

Všechny tyto dílčí transformace lze vyjádřit společnou rovnicí. Pro přepočet z globálního do lokálního systému souřadnic se používá tento výpočet:

$$C_n^b = C_3 * C_2 * C_1 \tag{9}$$

Pro transformaci v opačném směru má rovnice tvar:

$$C_n^g = C_3^T * C_2^T * C_1^T \tag{10}$$

Tím vznikne rotační matice *C*, kterou násobíme vektor, který chceme transformovat. Výhodou oproti Eulerovým úhlům má tato reprezentace v tom, že zde nenastává *gimbal lock*. Naopak nevýhodou jsou vyšší výpočetní nároky [17].

2.3 Senzory používané v INS

Základ funkční inerciální jednotky (IMU) v navigačním systému obvykle tvoří dva typy senzorů: akcelerometry a gyroskopy. V přesnějších *strapdown* systémech jsou přítomny ještě magnetometry pro zpřesnění a kompenzace *driftu* orientace [24]. Skupina gyroskopů měří úhlovou rychlost, na jejímž základě se vypočítává orientace v prostoru. Druhý typ senzorů, akcelerometry, zaznamenávají síly působící na zařízení, ze kterých se výpočtem získá rychlost a pozice v prostoru. Jelikož inerciální navigační systémy trpí určitým stupněm nepřesnosti, který v čase způsobuje *drift* pozice, existuje v dražších a přesnějších jednotkách ještě další skupina senzorů, magnetometry. Ty měří sílu magnetického pole a slouží pro kompenzaci *driftu* orientace způsobené nepřesností gyroskopů [24].

Většina moderních INS systémů slouží pro měření v prostoru, z toho důvodu každá jednotka obsahuje po třech akcelerometrech a gyroskopech. Každý člen z trojice má na starosti měření na jedné z os prostorových kartézských souřadnic.

Takovéto zařízení se označuje termínem 6DOF *(Degrees of Freedom)*, což znamená, že zařízení je schopno na každé ose soustavy souřadnic měřit sílu a změnu úhlu. Tato schopnost určuje naprostou volnost pohybu, neboť jakýkoliv pohyb je zaznamenán bez ztráty informace. Existují však zařízení s menším počtem senzorů, např. 3DOF, které obsahuje dva akcelerometry a jeden gyroskop, ale lze je využít pouze pro speciální případy, kdy máme pohyb zařízení v prostoru předem omezen.

Jednotky IMU označené zkratkou 9DOF nezaznamenávají pohyb na dalších osách zařízení, nýbrž obsahují společně se třemi akcelerometry a třemi gyroskopy ještě skupinu tří magnetometrů. Ty slouží ke měření úrovně magnetického pole na každé ze tří os a slouží ke kompenzaci *driftu* orientace způsobené chybami gyroskopu.

2.3.1 Gyroskop

Tradiční mechanický gyroskop je schopen měřit orientaci, tedy úhly. Naproti tomu moderní gyroskopy měří úhlovou rychlost a úhel je pomocí výpočetní jednotky získán výpočtem. V této kapitole jsou stručně shrnuty typy a konstrukce gyroskopů.

2.3.1.1 Mechanický gyroskop

Tento typ gyroskopického senzoru využívá vlastnosti rychle rotující kotouče zavěšeného v mechanické konstrukci. Rotující kotouč díky uchovávání svého úhlového momentu nepodléhá změnám v orientaci a zůstává ve výchozí pozici. Změnu v orientaci lze odečíst ze změny konstrukce vůči ose rotujícího kotouče [20].

Mechanický gyroskop jako jediný typ neměří úhlovou rychlost, ale měří úhly neboli skutečnou orientaci přístroje. Hlavní nevýhoda tohoto typu gyroskopu je, že obsahuje pohyblivé části. Mechanické části nejsou jen náročnější na výrobu, ale díky tření, které nelze úplně eliminovat, vzniká nezanedbatelná chyba, která s časem roste, čímž vytváří *drift*. Gyroskop také potřebuje čas pro uvedení do chodu, což nemusí být vhodné pro aplikace, kde je vyžadována okamžitá připravenost [20].

2.3.1.2 Optický gyroskop

FOG (*Fibre Optic Gyroscope*), neboli optický gyroskop, se skládá z kotouče optického vlákna a zdroje světelného paprsku. Při měření úhlové rychlosti se využívá tzv. Sagnacova efektu. Do obou konců vlákna je vyslán paprsek. Pokud vlákno podléhá rotaci, paprsek v jejím směru musí překonat delší vzdálenost než paprsek v opačném směru. Na

výstupu z vlákna vzniká interferenční obrazec, jehož intenzita závisí na úhlové rychlosti [20].

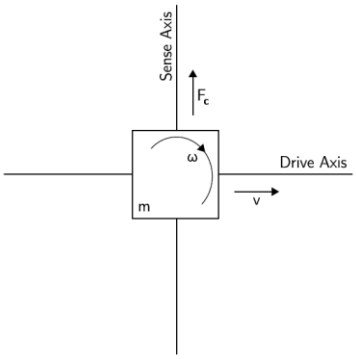
Výhodami tohoto typu gyroskopu je absence pohyblivé části a vysoká přesnost. Naopak nevýhodou je cena optického vlákna. S rostoucí délkou optického vlákna roste i přesnost gyroskopu [20].

2.3.1.3 Vibrační gyroskop

Tento typ konstrukce gyroskopu využívá ke měření úhlové rychlosti Coriolisovu sílu. Vibrující část senzoru se pohybuje v rovině osy otáčení senzoru. Při rotaci se pohybující část snaží zůstat v původní poloze a působí silou na snímač. Tato síla je způsobena právě Coriolisovou silou.

Coriolisova síla je setrvačná síla působící na hmotu v rotující neinerciální soustavě. *Proof mass* gyroskopu vibruje kolmo na osu otáčení a pří rotaci působí silou F_c na snímač gyroskopu [20].

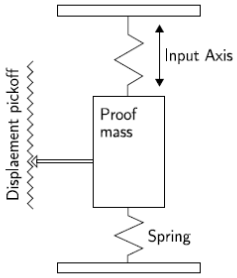
$$F_c = -2m(\omega \times v) \tag{11}$$



Obrázek 9: Princip vibračního gyroskopu, převzato z [20]

2.3.2 Akcelerometr

Senzor akcelerace měří zrychlení působící na zařízení včetně působení gravitačního pole Země. Senzor funguje na principu měření síly působící na pohyblivou část senzoru. Princip senzoru je, že měřící hmota (proof mass) je upevněna na pružině a v klidovém režimu je udržována na nulové hodnotě [20].



Obrázek 10: Princip mechanického akcelerometru, převzato z [20]

Působením akcelerace na zařízení dojde k vychýlení *proof mass* z nulové hodnoty a dojde k změření působící síly. Akcelerace je poté vypočtena jako:

$$a = -\sum F_s/mass \tag{12}$$

kde *mass* je hmotnost pohyblivé složky, F_s je změřená síla působící na pružinu.

2.3.2.1 Mechanický akcelerometr

Skládá se z měřící hmoty zavěšené na pružinách. Při změně vychýlení měřící hmoty se mění výstupní signál senzoru, který je úměrný síle působící na hmotu v měřicí ose. Výpočtem odvozeným z druhého Newtonova zákona se vypočte akcelerace působící na zařízení v měřící ose (viz rovnice 12). Tento typ akcelerometru se také označuje jako *open-loop* [20].

2.3.2.2 Akcelerometry bez mechanických částí

Existuje více typů konstrukcí senzoru akcelerace. Akcelerometry nevyužívající mechanických částí se označují termínem *Solid State Accelerometers*. Tato konstrukce se také označuje termínem *close-loop*. Lze je rozdělit do různých dílčích kategorií, kam patří například SAW (*Surface Acustic Wave*), vibrační a piezoelektrické akcelerometry [20].

2.3.3 MEMS technologie

Technologie MEMS (*Micro-Electro-Mechanical System*), neboli technologie společné integrace mechanických a elektronických struktur, přinesla nové možnosti konstrukce a aplikace v oblasti senzorové techniky. Technologie MEMS integruje mechanický senzor a řídící jednotku na jeden čip. Tím dojde k výraznému zjednodušení konstrukce a zlevnění výroby [17].

Čip může obsahovat mnoho pohyblivých částí nebo naopak žádnou. MEMS zařízení může mít velikost řádově jednotek mikrometru až několik milimetrů.

Mezi výhody této výrobní technologie patří zejména nízká cena, malé rozměry, nízká spotřeba a široké možnosti aplikace. Ovšem cenou za rozměry a cenu je nižší přesnost a výraznější šum [17].

2.3.3.1 MEMS gyroskop

MEMS gyroskopy fungují na podobném principu jako vibrační gyroskopy. Nemají žádnou rotující část a k měření úhlové rychlosti využívají Coriolisova efektu [17]. Coriolisův efekt je jev, při kterém v rotujícím systému souřadnic s úhlovou rychlostí ω se pohybuje hmota rychlostí v a působí na ní síla F_c ze vzorce (12).

Gyroskopy typu MEMS obsahují vibrující prvek pro měření Coriolisovy síly, ze které se vypočte úhlová rychlost. Pohyblivá část vibruje kolmo na osu rotace. Existuje mnoho druhů vibrujících struktur, některé mohou mít podobu ladičky nebo kruhu [20].

2.3.3.2 MEMS akcelerometr

Elektromechanické mikročipové akcelerometry využívají stejné principy jako jejich větší příbuzní. Dělí se opět na dvě kategorie podle použité konstrukce.

Mechanické MEMS akcelerometry měří posun měřící struktury. Při působení zrychlení na zařízení dojde k vychýlení z rovnovážného stavu a ke změně kapacity mezi elektrodami integrovaného obvodu [17].

S*olid state* MEMS akcelerometry využívají vibrující hmotu na určité frekvenci. Působí-li vnější síla na zařízení, detekují změnu frekvence a přepočtou na odpovídající zrychlení [17].

2.4 Senzory v systému Android

Systém Android díky své přizpůsobitelnosti v základu podporuje celou škálu senzorů. Navíc díky své otevřenosti je možné, s odpovídající softwarovou implementací, připojení dalších speciálních senzorů.

Většina současných zařízení se systémem Android obsahuje implementované senzory, které měří pohyb, orientaci a rozdílné podmínky v okolí přístroje. Nejrůznější aplikace společně s grafickým rozhraním se přizpůsobují podle údajů z těchto senzorů. Například natáčení obrazu, náklon ve hrách či zatřesení jako znamení pro akci.

Všechny druhy senzorů lze roztřídit do tří základních skupin [25].

- snímání pohybu
- snímání pozice telefonu
- snímání okolních podmínek

Samostatnou kategorií senzory poskytující lokalizační služby. Využívají GNSS nebo lokalizaci pomocí sítí Wi-Fi. Základní implementací satelitní navigace, která je dnes již standardem v chytrých mobilních telefonech, je americká GPS.

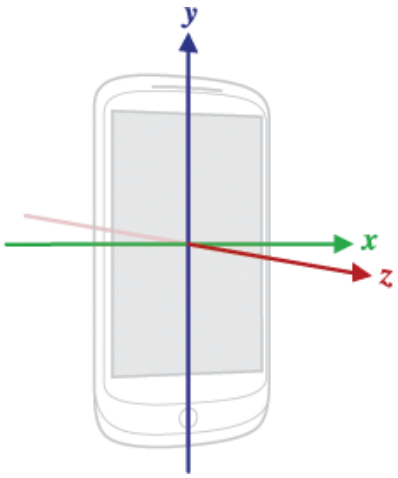
Dále se senzory dělí na hardwarové a softwarové. Hardwarové jsou fyzicky v zařízení přítomny a předávají měřenou veličinu přímo aplikaci. Softwarové nejsou fyzicky přítomny, i když se tak v systému prezentují. Získávají data z jednoho či více hardwarových senzorů a vyhodnocují je. Někdy se také nazývají virtuálními senzory [25].

Ve svém projektu jsem pro určování polohy vyhodnocoval data z těchto senzorů: senzor lineární akcelerace, gyroskopu a z GPS.

2.4.1 Systém souřadnic mobilního telefonu

Definice standartního systému souřadnic se liší podle typu zařízení i podle používaného senzoru. Mobilní telefon je standardně držen ve vzpřímené poloze, označené termínem *Portrait*. Proto jsou osy pravotočivého systému souřadnic definovány takto:

- Osa X míří ven z pravé (delší) strany přístroje.
- Osa Y vychází z horní (kratší) strany telefonu
- Osa Z míří z displeje k uživateli.



Obrázek 11: Systém souřadnic mobilního telefonu, převzato z [25]

Pro zařízení typu tablet, u kterého se předpokládá standartní držení v režimu „naležato“, neboli *Landscape*, jsou osy definovány takto:

- Osa X míří ven z pravé (kratší) strany přístroje
- Osa Y vychází z horní (delší) strany tabletu.
- Osa Z je pak definována stejně jako v případě mobilního telefonu a vychází z displeje směrem k uživateli.

Toto rozdělení není absolutní, neboť opět záleží na výrobci, který určí standartní držení zařízení a podle toho nastaví výstup ze senzorů. Příkladem budiž testovací zařízení Nexus 7, které se svojí konstrukcí řadí do kategorie tabletů, ovšem výrobce preferuje ovládání přístroje v režimu *Portrait* a tomu odpovídá i nastavení os, které je stejné jako v případě mobilních telefonů [25].

2.4.2 Senzor lineární akcelerace

Senzor lineární akcelerace je ve většině případů softwarového typu, ale může být přítomen i fyzicky. Tento senzor měří zrychlení ve směru každé ze tří os přístroje bez působení gravitace. Dodává údaje o zrychlení v jednotkách [ms^{-2}] z akcelerometru po vyfiltrování vlivu gravitačního pole Země [26]. Údaje a způsoby měření najdete v příslušné kapitole.

Implementace tohoto filtru se může zařízení od zařízení velmi lišit, neboť algoritmus a filtr pro získání dat implementuje výrobce a není to standartní součástí systému Android. Standartní schéma pro získání lineární akcelerace je však [25]:

$$\textit{linear acceleration} = \textit{acceleration} - \textit{acceleration due to gravity} \tag{13}$$

2.4.3 Gyroskop

Další senzor, který ve své bakalářské práci využívám, je gyroskop. Gyroskop měří změnu úhlové rychlosti v jednotkách [$rads^{-1}$] okolo každé ze tří os telefonu [25]. Pro výpočet úhlu se změna úhlové rychlosti integruje podle vzorce:

$$\theta(t) = \int_0^t \dot{\theta}(t)dt \approx \sum_0^t \dot{\theta}(t)T_s \tag{14}$$

Systém souřadnic je stejný jako v případě akcelerometru a senzoru lineární akcelerace. Rotací okolo osy proti směru hodinových ručiček získáme kladné hodnoty, zatímco v opačném směru záporné [25].

Na začátku měření dojde k nastavení výchozí polohy a úhly se vypočítávají vůči této orientaci. Výchozí orientace nemusí odpovídat globálnímu systému souřadnic, kdy osa Y míří na sever, osa X na východ a osa Z je kolmá k povrchu Země a míří vzhůru. Nicméně ve své práci doporučuji zařízení položit na vodorovný povrch, aby došlo alespoň k částečnému zarovnání.

2.4.4 GPS

Senzor GPS v dnešní době obsahuje téměř každý telefon s běžícím systémem Android. Tento poslední senzor využitý v mé práci jako jediný neměří polohu v lokálním, ale naopak v globálním systému souřadnic.

Senzor má velmi dobrou integraci do systému Android, neboť byl přítomen již od raných verzí systému. Je dostupný prostřednictvím dvou různých knihoven. Základní API *android.location* poskytuje přístup k senzoru GPS a získání výstupních dat. Pokročilejší

API *google.android.gms.location* je doplněno o pokročilé služby jako je hlavně optimalizace spotřeby energie a kombinace dat z Wi-Fi pro rychlejší určení polohy [27].

Ve svém projektu jsem se zaměřil na práci s API (*google.android.gms.location*) [27]. Mezi užitečná data získávaná ze senzoru GPS patří zejména zeměpisná délka, šířka a nadmořská výška. Tyto údaje slouží k určení pozice v prostoru. Mezi další data patří vzdálenost od předchozího bodu, rychlost, azimut a přesnost dat v metrech.

2.5 Senzorová fúze

S vývojem vojenské techniky došlo na zvýšení požadavků na přesnost a relevantnost senzory naměřených dat. Postupem času našla senzorová fúze využití i v civilním životě, například v robotice, medicíně či výzkumu [28].

Sloučení dat z více senzorů pro vyšší přesnost a relevanci se označuje termínem senzorová fúze, neboli *sensor fusion*. Existuje více druhů zpracování dat. *Data Fusion* je termín označující sloučení dat z více senzorů různých, či stejných typů, zatímco *Information Fusion* označuje kombinaci dat ze senzorů a informací poskytnutých uživatelem [28].

Princip slučování dat a informací z více zdrojů není pro lidstvo a zvířectvo ničím novým. Z evolučního hlediska tělo poskytuje informace z pěti smyslů. Schopnost vyhodnocovat tato data společně pomáhá člověku k přežití. Například je mnohem výhodnější a efektivnější vyhodnotit poživatelnost ovoce kombinací vizuálních, hmatových, čichových a chuťových vjemů, než použitím pouze jednoho. Z toho vyplývá, že pro poznávání svého okolí je proces senzorové fúze zcela zásadní a pokud chceme pokročit ve vývoji senzorové techniky, jde o zcela logický krok.

Při měření jedné veličiny s použitím více senzorů stejného typu můžeme pomocí senzorové fúze zpřesnit měřený odhad [28].

Použijeme-li více typů senzorů měřících různé veličiny, z nichž každý má určité vlastnosti, můžeme vyhodnocováním dat vzájemně tyto vlastnosti kompenzovat. Příkladem budiž senzory v 9DOF jednotce IMU, kde tříosý magnetometr slouží k potlačení gyroskopického *driftu* [28].

Metod zpracování dat pomocí senzorové fúze existuje celá řada. Mezi nedůmyslnější patří varianty tzv. *Kalmanova* filtru [29]. Ve své práci jsem se však zabýval implementací kombinace jednoduchých typů filtrů. Typy filtrů jsou podrobněji popsány v praktické části této práce.

2.5.1 Využití senzorové fúze v IMU

Pro realizaci jednotky IMU v *strapdown* systémech se využívá senzorové fúze k vzájemnému přepočítávání výstupů ze senzorů.

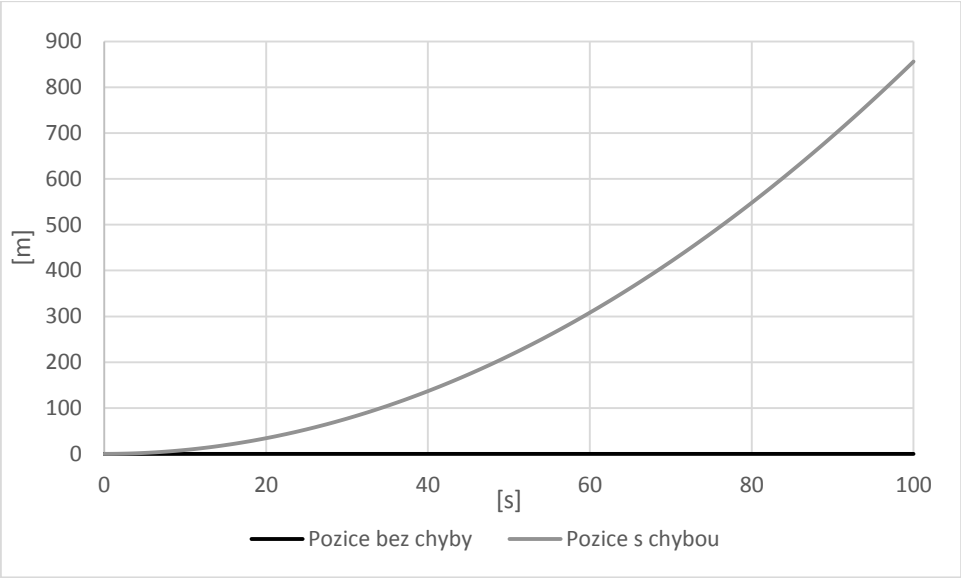
Pro získání lineárního zrychlení v rámci globální soustavy souřadnic se výstup z akcelerometru transformuje pomocí změřené orientace pomocí gyroskopu. V dalším kroku dojde k odečtení gravitačního vlivu. Po získání lineární akcelerace je možné vypočítat rychlost a pozici zařízení vzhledem k počátečním podmínkám [20].

Je tedy zřejmé, že pro co nejpřesnější určení pozice je nutná vysoká přesnost použitého gyroskopu. Při chybném určení orientace nemůže dojít ke korektní kompenzaci vektoru gravitace.

Na příkladu lze takovou skutečnost dobře ilustrovat. Při chybném určení orientace o pouhý 1°, vzniká vinou odečtení gravitačního zrychlení o velikosti *9,81 ms*$^{-}$2 chyba

$$\varepsilon = 0{,}1712\ ms^{-2}$$

Kvůli působení pouze této chyby dojde během 1 minuty k chybnému určení pozice o *300 m*.



Obrázek 12: Ukázka vlivu chybného určení orientace o pouhý 1°.

2.6 Příklady využití inerciální navigace v civilní sféře

2.6.1 Systém zvyšující přesnost GPS

Výzkumníci ze španělské Univerzity Karla III. V Madridu (*UC3M*) v roce 2012 vyvinuly systém, který zvyšuje přesnost určení polohy automobilu pomocí GPS v zástavbě až o 90 % [30].

Systém využívá standardní poziční čip GPS a jednotku IMU obsahující akcelerometr a gyroskop. Pomocí senzorové fúze se snaží o zvýšení přesnosti polohy vozidla. V husté městské zástavbě chyba určení pozice GPS vlivem odrazu signálu mezi budovami může být i větší než 50 m. Může docházet i ke chvilkovým ztrátám signálu, např. v tunelu. Ovšem díky kombinaci dat z GPS a IMU lze polohu ve městě zpřesnit až na chybu řádově jednotek metrů. Navíc IMU dokáže určovat pozici automobilu krátkou dobu i po ztrátě signálu z GPS [30].

Hardware instalovaný ve výzkumném vozidle se skládá z jednotky GPS, jednotky IMU a výpočetního zařízení. Software pro zpracování dat využívá modifikovaný *Kalmanův* filtr. Tento filtr pro zpracování dat využívá model chování automobilu [30].

Pozice jednotky IMU je taková, že osa *X* směřuje ve směru pohybu vozidla, osa *Z* je vertikální a osa *Y* s nimi tvoří pravotočivou bázi, tedy míří z boku vozidla ven. Systém měří pouze polohu ve dvou dimenzích, tedy polohu *XY* [30].

V tiskové zprávě výzkumná laboratoř prohlašuje, že dalším krokem je výzkum využití senzorů v moderním chytrém mobilním telefonu [30].

2.6.2 Projekt Tango

Jedno z využití inerciální navigace předvedla společnost Google, když v únoru letošního roku uveřejnila výsledky vývoje nové generace mobilního zařízení schopného vnímat okolní prostor [31]. Projekt nese kódové označení Tango a ukazuje jeden z možných směrů vývoje v segmentu chytrých mobilních zařízení. Na vývoji trvajícím rok se podílelo několik spolupracujících výzkumných laboratoří univerzit a robotických společností [32].

Jde o snahu vytvořit zařízení, jenž bude schopné rozumět okolnímu prostoru podobným způsobem jako člověk. Bude vnímat okolního prostor a svoji polohu v něm. Prototyp je osazen na míru řešeným hardwarem. Obsahuje inerciální senzory pro záznam pohybu v prostoru, infračervený snímač hloubky prostoru a několik kamer, z nichž jedna slouží pro sledování objektů v obraze [33].

Při snímání inerciální polohy zpracuje procesor každou vteřinu čtvrt milionu vzorků. Google nezpřístupnil veřejnosti podrobnosti o použitých senzorech, ale z ukázkových videí lze zjistit, že zařízení využívá data z gyroskopu a akcelerometru. Pro kompenzaci chyb z IMU se pravděpodobně využívají údaje o prostoru z dostupných kamer [33].

Přesnost tohoto zařízení je velmi působivá. Při pohybu v budově, kdy měřící člověk prošel okruh o délce 380 m, překonal pět podlaží vzhůru a zpět, výsledná chyba od skutečné pozice činila pouhé 1 % [33].

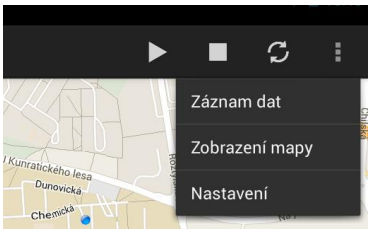
Možnosti využití jsou široké a záleží pouze na lidské fantazii. Mezi pár příkladů uveďme možnost 3D skenování objektů a prostor, navigace uvnitř budov či pomocník zrakově postižených [33].

3 Praktická část

Při realizaci vlastní implementace inerciální polohové jednotky jsem zkoumal vlastnosti použitých senzorů a metody vhodného filtrování. Dále jsem vytvořil prototyp aplikace, která může dále v budoucnu sloužit jako platforma pro další vývoj.

3.1 Návrh a realizace první měřící aplikace

Během předcházejícího semestru byla vytvořena velmi jednoduchá aplikace sloužící pro prostý sběr dat z vybraných senzorů. Následné zpracování probíhalo v počítači pomocí výpočetního softwaru Matlab. Při práci na této bakalářské práci jsem původní aplikaci rozvinul a zapracoval do finální aplikace. Tato původní aplikace je nyní přístupná přes odkaz Záznam dat v roletkovém menu.



Obrázek 13: Přístup k první měřicí aplikaci

Aplikace má velmi jednoduchou strukturu, neboť se skládá pouze z jedné měřicí aktivity. V aktivitě probíhá záznam vybraných dat ze senzoru GPS, senzoru lineární akcelerace a gyroskopu. Aktivita má velmi jednoduché uživatelské rozhraní využívající základní rozložení uživatelských prvků nazvané *Linear Layout*.

Obrázek 14: UI měřicí aplikace

Ve svém prostředí zobrazuje časový údaj po spuštění testu v sekundách, dále data z GPS jako je zeměpisná délka a šířka, nadmořská výška a přesnost údaje v metrech. Dalšími zobrazenými údaji jsou data ze senzoru lineární akcelerace a data o úhlové rychlosti z gyroskopu pro každou měřenou osu.

K ovládání záznamu dat slouží dvě tlačítka. Jedno spouští záznam a druhé ho zastaví a provede zápis do textového souboru, který uloží do kořene externího úložiště.

Data ukládá do objektu jednorozměrného pole a tyto objekty po dobu měření ukládá do pole objektů. Po ukončení měření provede strukturovaný zápis do textového souboru, který poté může být dále zpracován.

3.2 Návrh implementace vlastní měřicí jednotky

Pro vytvoření standardního inerciálního pozičního systému se používají akcelerometry a gyroskopy. Ve vlastním návrhu *strapdown* systému jsem se ovšem pokusil využít výhody v systému již implementovaného senzoru lineární akcelerace.

Použití tohoto senzoru nám umožní vyhnout se problémovým krokům při realizaci. Předně, pokud získáme lineární zrychlení přímo ze senzoru, není již třeba odečítat vliv gravitačního zrychlení. Díky této vlastnosti již není kladen takový důraz na přesnost určení orientace zařízení (viz kapitola o realizaci inerciální navigaci). Ze senzoru lineární akcelerace tedy po příslušné filtraci dat získáme hodnoty zrychlení působících na zařízení.

K určení orientace slouží naměřená data z gyroskopu. Na začátku měření umístíme zařízení do výchozí polohy, v ideálním případě na vodorovný povrch, a poté určujeme orientaci přístroje vzhledem k výchozí poloze pomocí integrace dat gyroskopu. Úhel okolo každé z os lze určit jako integraci úhlové rychlosti podle času.

$$\theta = \int \omega \, dt \tag{14}$$

Kombinací dat ze senzoru lineární akcelerace a orientace z gyroskopu získáme transformované zrychlení vzhledem k výchozí orientaci přístroje. Z takto přepočtených dat lze ze známých počátečních podmínek vypočítat odpovídající rychlost podle vzorce:

$$v = \int a \, dt \tag{15}$$

Naměřená pozice se vypočte podle vzorce:

$$s = \int v \, dt = \iint a \, dt \tag{16}$$

3.3 Návrh a struktura měřící aplikace

Aplikace se skládá ze tří aktivit a jedné služby. Hlavní aktivitou aktivovanou po spuštění je *MapActivity*. Hlavní aktivita spouští službu běžící na pozadí a vykonávající záznam a zpracování dat nazvanou *DataService*.

Uživatelské rozhraní *MapAcitivity* se skládá ze dvou fragmentů. Jeden zobrazuje mapu s využitím API *google.android.gms.maps*, kde uživatel může sledovat svůj pohyb a polohu na mapě určenou pomocí senzoru GPS. Druhý fragment zobrazuje v textové podobě data nasbírána a vypočtena ze všech využívaných senzorů. Množství dat je rozděleno do tří sloupců. Levý zobrazuje získaná data z GPS, prostřední naměřené a zpracované zrychlení a v pravém sloupci lze nalézt údaje o orientaci zařízení vůči výchozí poloze, vypočtenou rychlost a pozici z transformovaného zrychlení.

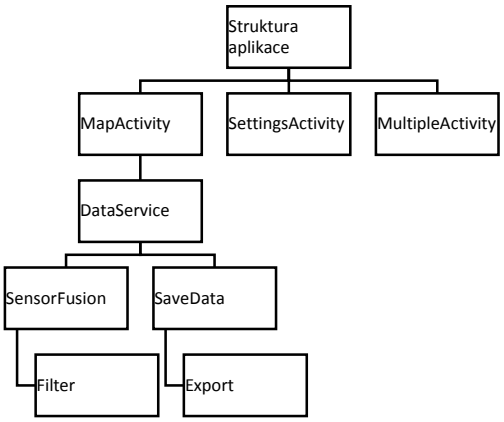
Druhou aktivitou je prvotní měřící aplikace, která vznikla v rámci semestrálního projektu. Tato komponenta je dostupná z roletkového menu a při aktivaci nespouští žádnou službu. Aktivita provádí měření dat ze senzoru GPS, lineární akcelerace a gyroskopu s nejvyšší možnou frekvencí čtení rovnou, která v případě Nexus 7 je $f = 203$ *Hz*. Zápis se provádí do textového souboru a uloží se do kořenového adresáře

externího úložiště. K ovládání aktivity slouží dvě tlačítka spouštějící a zastavující záznam dat.

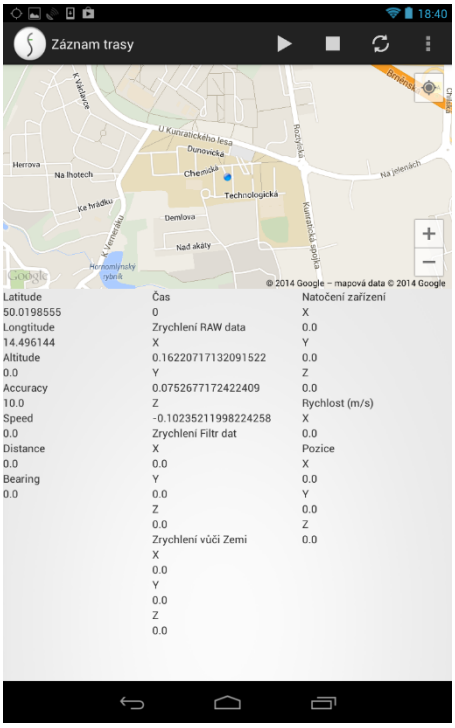
Třetí aktivita se jmenuje *SettingsActivity* a jak již název napovídá, slouží ke konfiguraci aplikace a sběru dat. Uživatel zde má možnost konfigurovat možnosti ukládání, nastavení filtrů jednotlivých senzorů a nastavení zobrazení uživatelského rozhraní *MapActivity*.

Služba provádějící sběr a zpracování dat se nazývá *DataService*. Nemá uživatelské rozhraní a je spouštěna při startu *MapActivity*, která ji však automaticky neukončuje a musí ji uživatel vypnout ručně. Důvod pro ruční vypínání je takový, aby bylo možné měřit i pokud není viditelná *MapActivity*, příkladem je třeba vypnutý displej. Služba řídí sběr dat ze senzorů a jejich následné zpracování předává třídě *SensorFusion*. Při spuštění záznamu dat, načte nastavení pro záznam z knihovny *SharedPreferences* a toto nastavení předá třídě zpracovávající data, *SensorFusion*. Během záznamu se stará o zobrazení informací o měření v uživatelském rozhraní *MapActivity*. Po ukončení a pro uložení záznamu *DataService* zavolá další třídu tentokrát nazvanou *SaveData*, která se postará o zápis požadovaných hodnot do textového souboru. V případě požadavku na uložení do formátu *kml*, třída *SaveData* předá data třídě *Export*, která provede zápis v požadované struktuře.

Mezi další implementované třídy, se kterými uživatel nepřijde do styku, patří třída *SensorFusion*, která řídí zpracování a filtraci dat pomocí dílčích tříd. Využívá služeb vytvořených instancí třídy *Filter*, která obsahuje metody pro filtraci vstupních dat. Procedury pro provádění senzorové fúze jsou implementovány přímo v třídě *SensorFusion*. Provádí se zde veškeré manipulace s daty.



Obrázek 16: Struktura vytvořené aplikace



Obrázek 15: Ukázka prostředí vytvořené aplikace

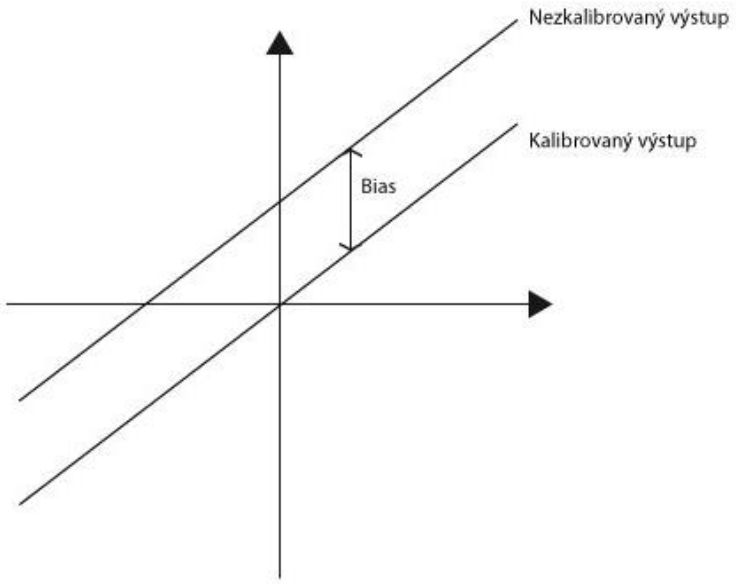
3.4 Metody filtrování dat

Senzory, kterými jsou osazovány moderní mobilní telefony, patří do levnější cenové kategorie. Nedisponují vysokou mírou přesnosti jako jejich pokročilejší a dražší příbuzní používané v profesionálních navigačních systémech. Z tohoto důvodu jsou výstupní data zatíženy různými faktory ovlivňující přesnost senzoru. Níže jsou uvedeny dva hlavní faktory ovlivňující použité senzory.

3.4.1 Faktory ovlivňující přesnost dat

3.4.1.1 Bias

Pokud na akcelerometr nepůsobí žádné zrychlení nebo na gyroskop žádná rotace, výstup ze senzoru by měl být roven nule. Pokud senzor v tomto případě dává na výstupu nenulové hodnoty, nazýváme tento jev termínem *bias*, neboli odstup signálu od skutečné hodnoty. Vliv tohoto faktoru lze rozdělit na statickou část zvanou *bias offset* a na dynamickou část *bias drift* [17].



Obrázek 17: Bias offset, převzato a upraveno z [17]

Konstantní část, neboli *bias offset*, lze snadno určit jako průměr naměřených hodnot, kdy zařízení nepodléhalo žádné rotaci. Chyba měření způsobená *bias offsetem* při integraci způsobí odchylku od skutečné hodnoty, která roste lineárně s časem [20].

$$\theta(t) = \epsilon \cdot t. \tag{17}$$

Kompenzace je po určení hodnoty tohoto posunutí velmi jednoduchá:

$$\textit{skutečná hodnota} = \textit{naměřená hodnota} - \textit{vypočtený offset} \tag{18}$$

Dynamickou část je velmi těžké určit bez bližších znalostí o daném senzoru a jeho charakteristik. Pokud nechceme mít měření významně ovlivněné touto chybou a nemůžeme vymodelovat funkční filtr, lze chybu minimalizovat měřením kratších časových intervalů.

V případě akcelerometru ovlivnění výsledků způsobené *bias driftem* je velmi malé a naopak podstatný vliv zde má *bias offset*. V případě gyroskopu je tomu opačně, větší

vliv na přesnost měření má *bias drift*. Tento drift gyroskopu je způsoben výrazným vlivem termo-mechanického bílého šumu [20].

3.4.2 Navržené filtrační metody

Ve své práci jsem se pokusil pomocí implementace jednoduchých filtrů dosáhnout relevantních výsledků. V případě použití *sensor fusion* se doporučuje využít pokročilejších filtračních metod, mezi něž patří varianty *Kalmanova* filtru.

3.4.2.1 Metoda plovoucího průměru

Anglický termín pro tuto filtrační metodu je *Moving Average Filter*. Princip tohoto filtru je takový, že nová hodnota se zprůměruje s určitým počtem hodnot, které jí předcházely. Vypočtená hodnota je zaznamenána.

Díky průměrování s předchozími hodnotami lze docílit hladšího průběhu signálu. Vlastnosti a výsledek filtrování ovlivňuje velikost filtračního okna, neboli počet hodnot *k*, které se průměrují.

Čím menší hodnota *k*, tím méně je signál filtrován. Při *k=1* je filtrovaný signál roven původnímu. Čím vyšší hodnota *k*, tím hladší průběh signálu. S rostoucím *k* však roste zpoždění za původním signálem a výpočetní náročnost filtru.

```
public double vypoctiPrumerVzorku(double[] vzorky) {

    double suma = 0;
    double prumer = 0;
    for (int i = 0; i < vzorky.length; i++) {
        suma += vzorky[i];
    }
    prumer = suma / vzorky.length;
    return prumer;
}
```

Obrázek 18:Příklad použitého algoritmu pro výpočet průměru ze zadaného souboru vzorků

3.4.2.2 Jednoduchý Low Pass Filter

Tato filtrační metoda opět slouží k vyhlazení průběhu signálu, princip je však odlišný od metody plovoucího průměru. Při filtraci se využívá určení váhy nového vzorku vzhledem k předchozímu. Implementace může mít několik podob [34]:

$$result = alpha * predchoziHodnota + (1 - alpha) * novaHodnota \quad (18)$$

$$result = predchoziHodnota + alpha * (novaHodnota - predchoziHodnota) \quad (19)$$

$$result = predchoziHodnota + (novaHodnota - predchoziHodnota)/alpha \quad (20)$$

Výsledky filtru opět ovlivňuje jediná proměnná, v tomto případě označená jako *alpha*. Pokud použijeme varianty implementující násobení, má *alpha* hodnotu z intervalu *<0;1>*. V případě použití dělení, *alpha* nabývá hodnot z intervalu *<1;∞>*.

Výhodou tohoto filtru je nízká výpočetní náročnost, nevýhodou opět vznikající zpoždění filtrovaného signálu při vysoké hodnotě *alpha*.

3.4.2.3 Metoda prahování (tresholding)

Jde o jeden z nejjednodušších filtrů. Princip filtru spočívá v porovnávání výstupních hodnot s nastavenou prahovou hodnotou. Existují dva typy *treshold* filtrace: *hard* a *soft*.

V případě *hard tresholdingu* se naměřená hodnota porovná s definovanou hranicí a pokud je menší, je nastavená nulová hodnota. V případě, že naměřená hodnota je větší než definovaný práh, je zaznamenána původní hodnota.

```
protected double [] pevnyFiltr(double [] hodnota){
    for(int i=0; i<hodnota.length;i++){
    if(hodnota[i]<filtrPevny && hodnota[i]>-filtrPevny){
        hodnota[i] = 0;
    }
    }
    return hodnota;
}
```

Obrázek 19: Příklad použitého kódu pro metodu pomocí prahování

Kód ukazuje implementaci *treshold* filtru. Pole hodnot určených k vyfiltrování se v cyklu porovná s nastavenou hranicí a v případě, že hodnota vzorku je menší než hranice, vzorek se zahodí.

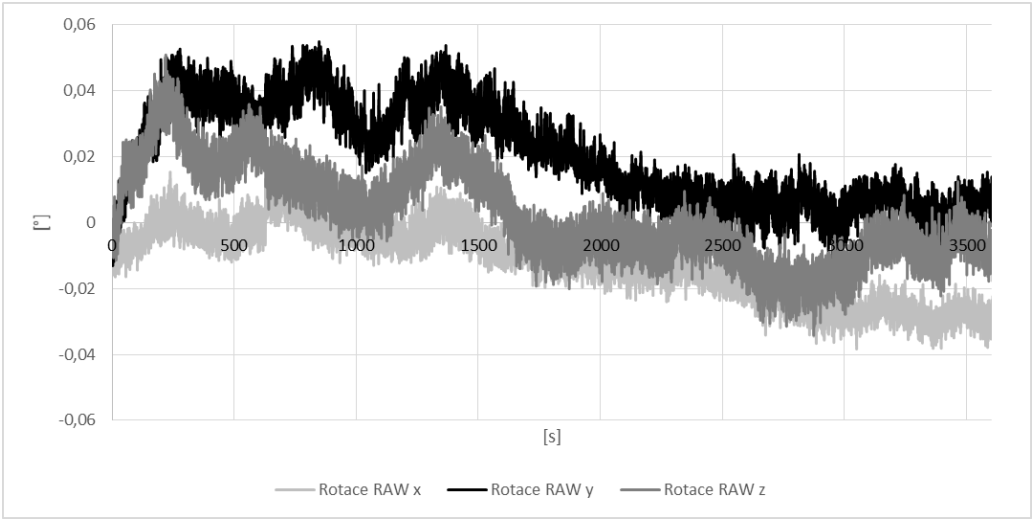
Soft tresholding se vyznačuje tím, že od hodnot větších než je nastavený práh ještě odečte hodnotu hranice, čímž dojde k posunutí hodnot signálu.

3.4.3 Implementace filtrů na výstup z gyroskopu

V této kapitole se nachází příklady použití a porovnání výstupů jednotlivých filtrů na data z gyroskopu. Na konci kapitoly lze nalézt navržený filtr, který kombinuje vícero filtračních metod.

3.4.3.1 Analýza výstupních dat z gyroskopu

Měřením reprezentativního pohybu získáme vhodný vzorek dat pro analýzu a navržení filtru. První měření bylo provedeno v klidu, kdy přístroj ležel na vodorovné podložce bez působení vnějších vlivů na časovém intervalu jedné hodiny.



Obrázek 20: Průběh vypočtené rotace v klidu

Pro vyhodnocení chyby způsobené *bias offsetem* vypočteme jednoduchý průměr hodnot na každé ose. Vypočtený offset je uveden v tabulce níže.

Tabulka 1: Vypočtený bias offset při měření v klidu

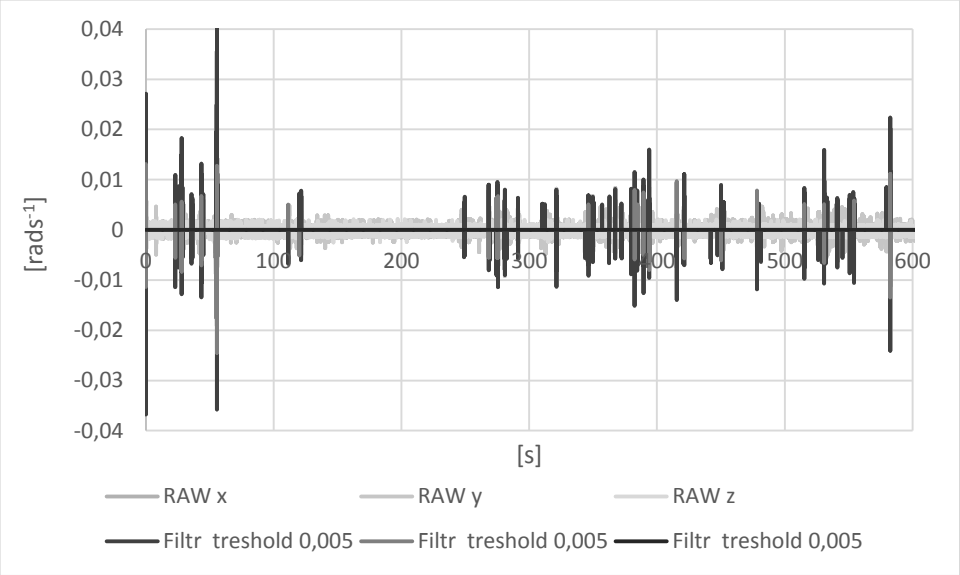
Osa	Vypočtený bias offset	Nejistota bias offsetu
X	1,52E-05	8,25E-06
Y	-1,1E-05	5,18E-06
Z	1E-07	3,53E-06

Z uvedených dat vyplývá, že chyba způsobená posunutím nuly je velmi malá a proto tento offset můžeme lze vyfiltrovat pouze pomocí jednoduchého *treshold filtru*.

Dynamická chyba způsobená kolísáním signálu má vliv na výsledek mnohem větší. Z grafu průběhu vypočtené orientace je projev této chyby patrný. Na začátku měření *bias drift* způsobí růst úhlu natočení, zatímco v polovině měřící doby naopak velikost úhlu klesá.

3.4.3.2 Treshold filter

Vhodným nastavením *treshold* filtru lze výše uvedené vlastnosti klidového stavu kompenzovat.

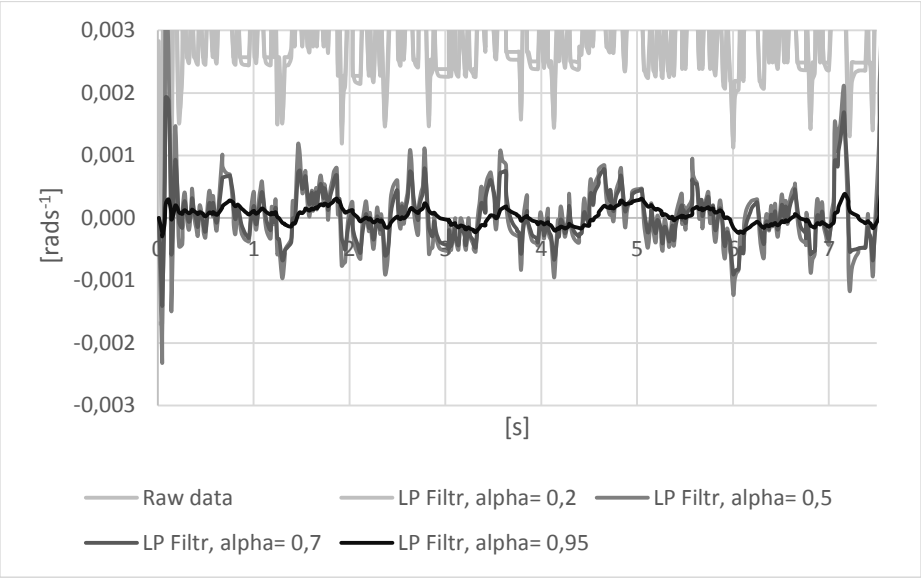


Obrázek 21: Ukázka použití treshold filtru při měření v klidu

Nevýhodou tohoto řešení je, že hodnoty menší než nastavená hranice jsou zahozeny a je tedy tak ztracena část informace o signálu. Další nevýhodou je, že nevyhlazuje průběh signálu a tedy původní citlivost senzoru je zachována.

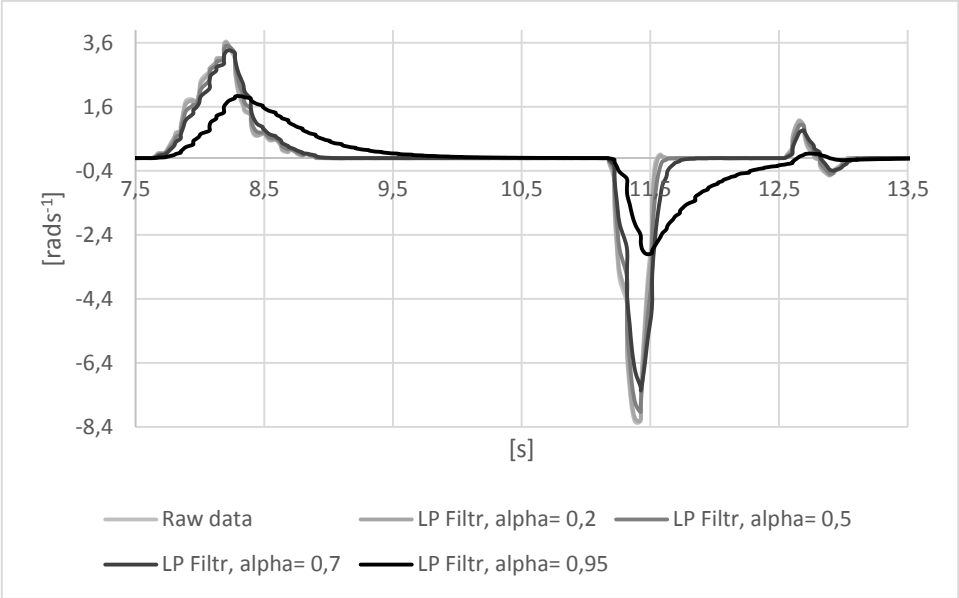
3.4.3.3 Simple Low Pass filter

Aplikace tohoto vyhlazovacího filtru dává na rozdíl od t*reshold* filtru mnohem lepší výsledky, neboť nedochází ke ztrátě informace o signálu. Filtrovaný signál je hladší než signál původní. Pokud je zařízení v klidu, požadovanému chování signálu se přiblížíme použitím vyšší hodnoty koeficientu *alpha*.

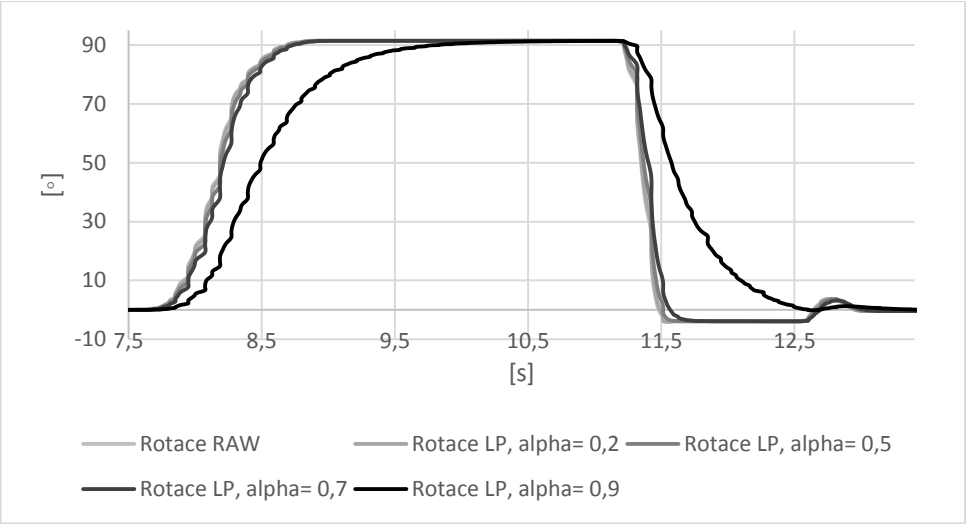


Obrázek 22: Ukázka aplikace Low Pass filtru při měření v klidu

Nevýhoda spočívá však v tom, že při těchto vyšších hodnotách koeficientu *alpha* dochází ke zpoždění signálu. V klidovém stavu, kdy chceme dostávat hodnoty co nejbližší nule, to nevadí, ovšem při záznamu pohybu je velké zpoždění signálu již nežádoucí.



Obrázek 23: Ukázka aplikace Low Pass filtru při měření úhlové rychlosti

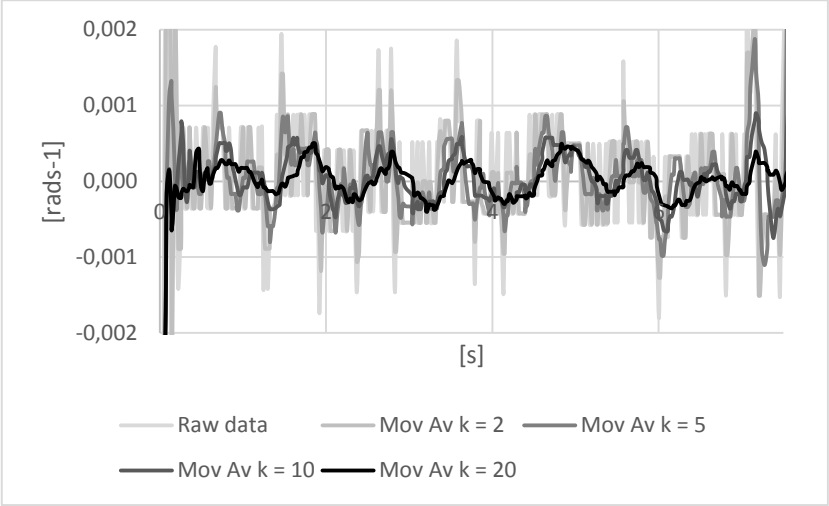


Obrázek 24: Ukázka aplikace Low Pass filtru při měření rotace

Podíváme-li se na průběh měřeného úhlu, zjistíme, že nejlepší výsledek, co se týče poměru vyhlazení a zpoždění, poskytuje nastavení hodnoty *alpha = 0,7*. *Při alpha < 0,7* je vyhlazení nižší, ale zpoždění signálu menší, pro *alpha > 0,7* je naopak vyhlazení roste, ovšem roste i zpoždění signálu.

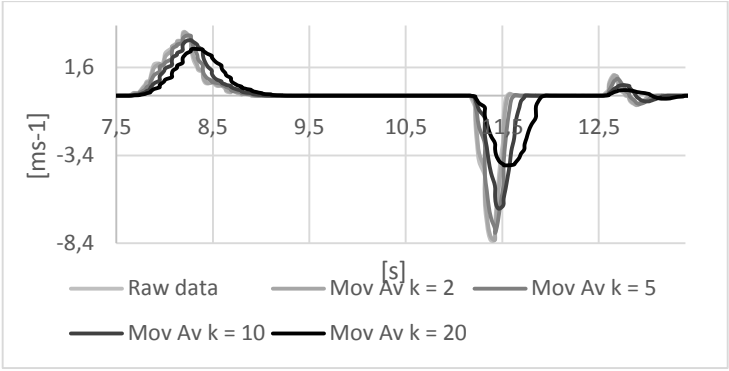
3.4.3.4 Moving Average filter

Aplikace tohoto typu filtru dává podobné výsledky jako předchozí *Low Pass filter*. Filtrovaný signál je opět hladší než signál původní. Ovládání výstupu filtru se provádí nastavením počtu hodnot k průměrování. Z grafu je patrné, že čím vyšší koeficient *k*, tím hladší signál, ale větší zpoždění za původním signálem.

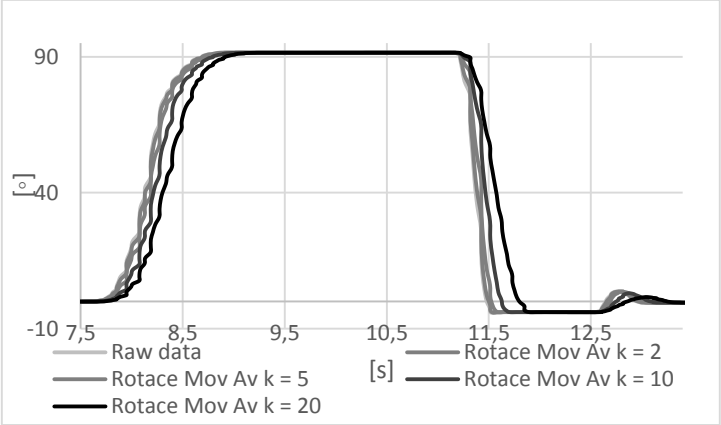


Obrázek 25: Ukázka použití filtru plovoucího průměru při měření v klidu

Během pohybu však díky průměrování dochází ke zpoždění signálu. V klidovém stavu, kdy se snažíme získat hodnoty blízké nule, to nevadí, ovšem při záznamu pohybu je již zpoždění nežádoucí.



Obrázek 26: Ukázka použití filtru plovoucího průměru při měření úhlové rychlosti

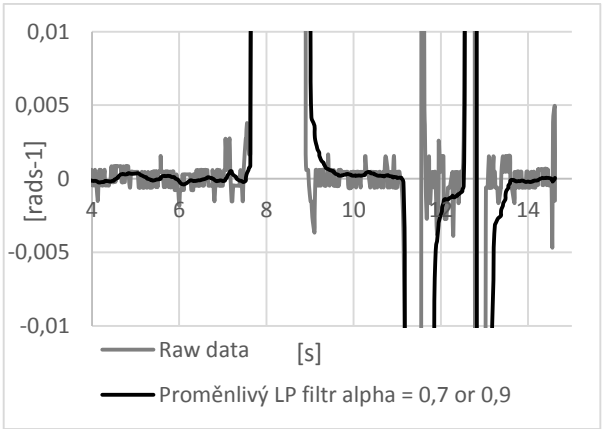


Obrázek 27: Ukázka použití filtru plovoucího průměru při měření rotace

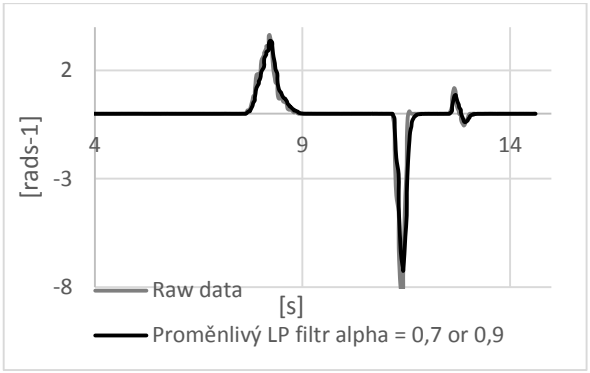
Z naměřených průběhů je patrné, že metoda plovoucího průměru dává podobné výsledky jako předchozí *Low Pass filter*. Nevýhodou je však vyšší výpočetní náročnost.

3.4.3.5 Navržený filtr

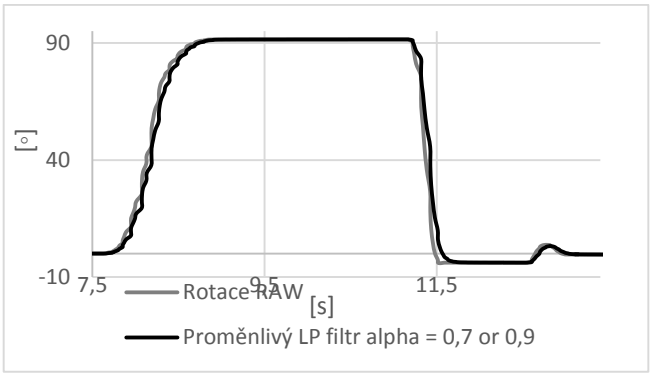
Pro optimální filtrování dat jsem navrhl filtr kombinující funkci *treshold* a *low pass* filtru. Signál je filtrován standardně pomocí *low pass* filtru s nastaveným koeficientem *alpha = 0,7* a zároveň probíhá porovnávání předem nastaveným intervalem pomocí hraniční hodnoty. Pokud leží výsledná dat v tomto intervalu, změní se *alpha* vyhlazovacího filtru na *alpha = 0,95*. Tím dojde k většímu vyhlazení hodnot, u kterých se předpokládá, že nedochází ke změně natočení a zároveň nedojde ke zvýšení zpoždění signálu.



Obrázek 28: Navržený filtr



Obrázek 29: Navržený filtr



Obrázek 30: Navržený filtr

3.4.4 Implementace filtrů na výstup ze senzoru lineární akcelerace

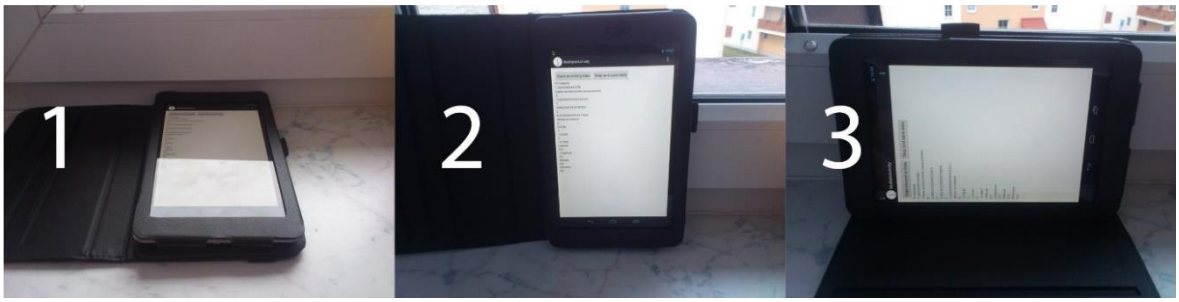
V této kapitole jsou uvedeny příklady použití a porovnání výstupů jednotlivých filtrů na data ze senzoru lineární akcelerace. Na konci kapitoly je popis navrženého filtru, který kombinuje vícero filtračních metod pro dosažení optimálních výsledků

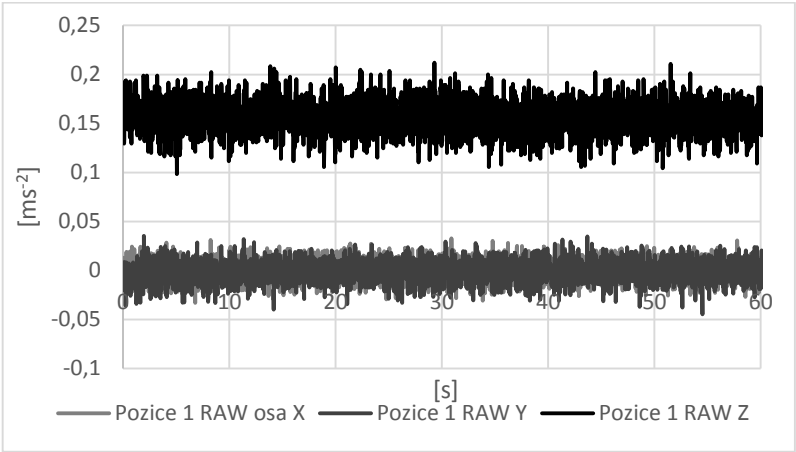
3.4.4.1 Analýza výstupních dat ze senzoru lineární akcelerace

Naměřená data získaná ze senzoru lineární akcelerace se mohou velmi lišit zařízení od zařízení. Jak již bylo zmíněno v kapitole 2.4, tento senzor je zpravidla v zařízení implementován softwarově a jeho implementace závisí na výrobcem použité metodě získání lineární akcelerace. Příkladem budiž testovací měření, které proběhlo s dvěma zařízeními za stejných podmínek.

3.4.4.1.1 Vlastnosti klidového stavu

První měření se týkalo zjištění offsetu výstupních hodnot, kdy zařízení bylo v naprostém klidu po dobu dvaceti minut. Měření bylo provedeno ve třech měřicích polohách. Poloha číslo 1 je, když zařízení leží na vodorovné ploše, displejem vzhůru. Poloha číslo 2 je, když zařízení stojí na levém boku, tudíž osa *X* je kolmá k Zemi a míří vzhůru. Třetí a poslední poloha je, když zařízení stojí na spodní hraně a osa *Y* je kolmá k Zemi a míří vzhůru.

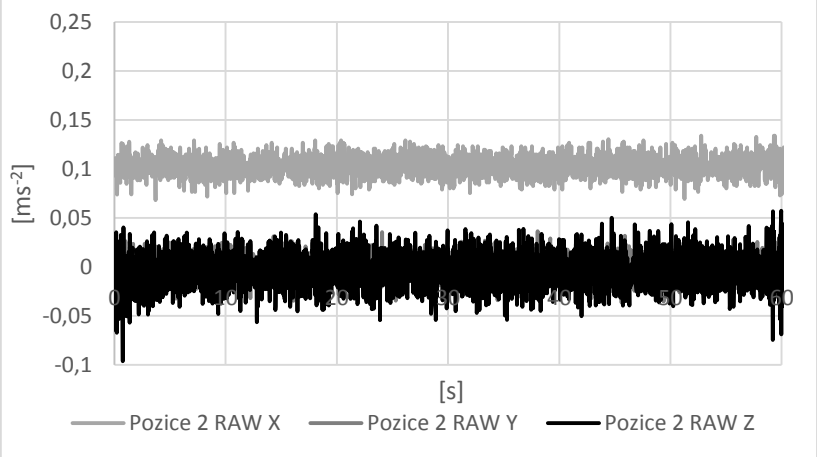




Obrázek 31: Průběh naměřených hodnot v klidu a poloze 1

Tabulka 2: Vypočtený bias offset v klidu a poloze 1

Osa	Vypočtený bias offset	Nejistota bias offsetu
X	0,000285	0,000177
Y	-0,0014	0,000196
Z	0,156335	0,000302

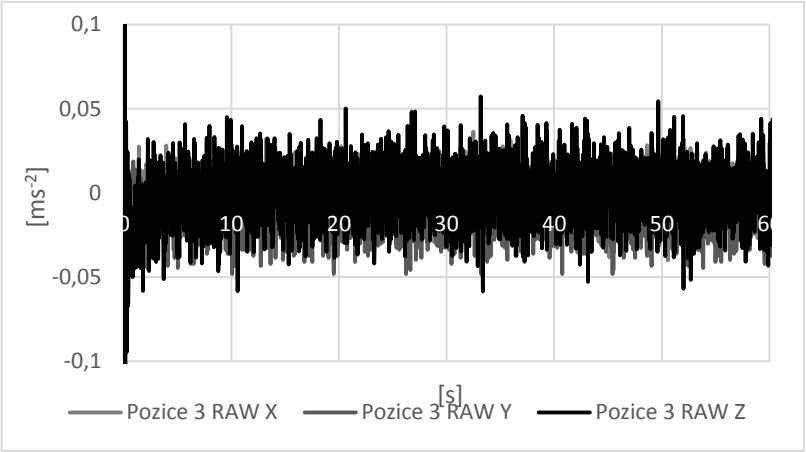


Obrázek 32: Průběh naměřených hodnot v klidu a poloze 2

Tabulka 3: Naměřený bias offset v klidu a poloze 2

Osa	Vypočtený bias offset	Nejistota bias offsetu
X	0,103326	0,001878
Y	-4,7E-05	0,000197

Z	-0,00364	0,000311



Obrázek 33: Průběh naměřených hodnot v klidu a poloze 3

Tabulka 4: Naměřený bias offset v klidu a poloze 3

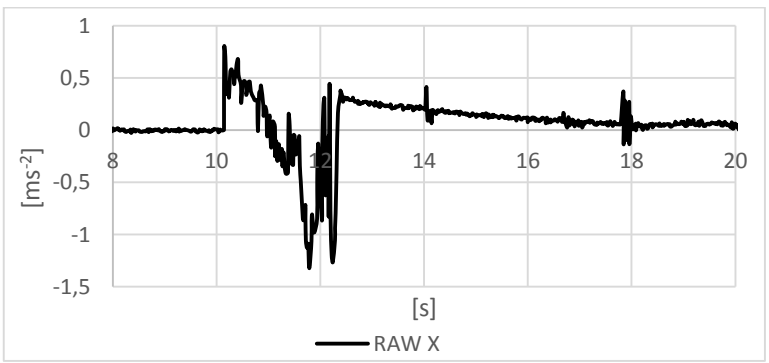
Osa	Vypočtený bias offset	Nejistota bias offsetu
X	-0,00056	0,000205
Y	-0,01274	0,000205
Z	-0,00065	0,000316

Jak je možné vidět z naměřených grafů, výstupní hodnoty ze senzoru lineární akcelerace jsou zatíženy *offsetem.* Tento *offset* vždy nejvýrazněji působí na tu osu, na kterou působí gravitační zrychlení. Je to dáno nedokonalostí kalibračních algoritmů implementovaných výrobcem. Tento fakt je třeba při implementaci zohlednit a navrhnout vhodný kompenzační algoritmus.

Při pohledu na výpočet offsetového posunutí lze spatřit, že nejistoty *biasu* os se vzájemně liší. Zatímco nejistota měření na ose *X* a *Y* je přibližně stejná, nejistota na ose *Z* je *1,5x* větší. To může být způsobeno zvýšenou citlivostí senzoru, či vyšším ovlivněním šumem.

3.4.4.1.2 Vlastnosti naměřených dat při pohybu

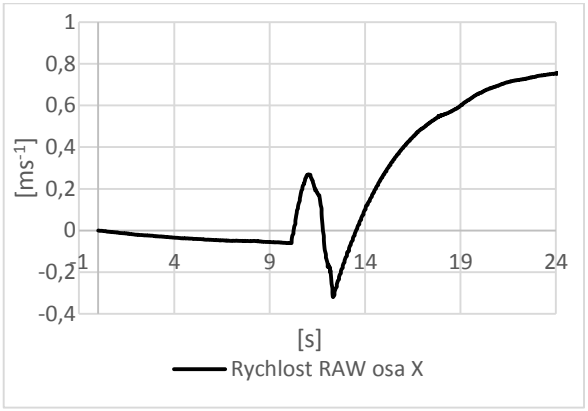
Během druhého měření byl prováděn pohyb reprezentující posun zařízení po vodorovné podložce ve směru osy *X* na vzdálenost 1 m.



Obrázek 34: Ukázka výstupu dat ze senzoru při záznamu posunu po vodorovné podložce na vzdálenost 1m.

Z grafu lze vyčíst, že po dokončení pohybu dochází k opětovnému ustalování nulové hodnoty. Tento návrat k nulové hladině zabere okolo 4 s. Doba je závislá na velikosti zrychlení. Čím větší zrychlení tím déle trvá ustálené hodnoty. Nastavení nulové hodnoty se nejvýrazněji projevuje na osách, na které v danou chvíli působí gravitační zrychlení nejméně a tedy *bias offset* je minimální. Důvodem vzniku tohoto jevu je pravděpodobně fakt, že ve výrobci naprogramovaném algoritmu získání dat z akcelerometru může být použit *Low Pass Filter*.

Na grafech níže je vidět vypočtená rychlost a pozice z těchto nefiltrovaných dat.



Obrázek 35: Vypočtená rychlost z nefiltrovaných dat

Obrázek 36: Vypočtená poloha z nefiltrovaných dat

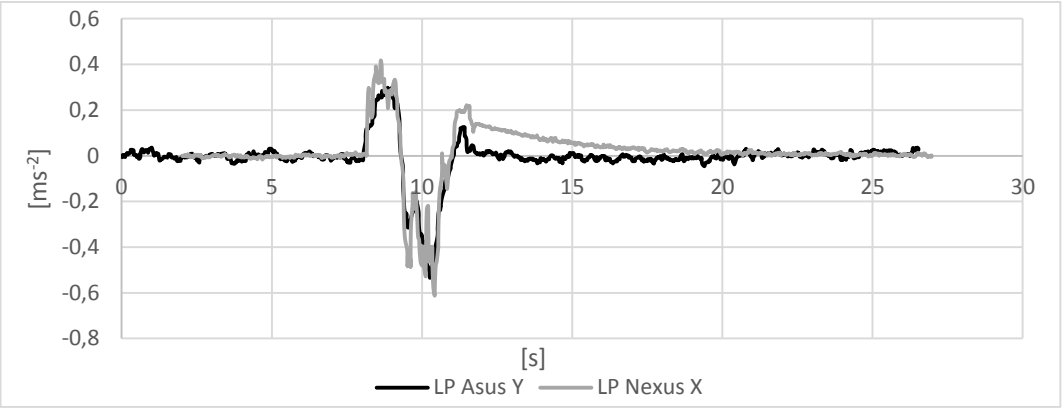
Na průběhu rychlosti je vidět výrazný vliv doběhu hodnoty po ukončení pohybu. Ten způsobuje výrazný drift rychlosti. Další zvláštností je, že při zastavení je změřené zrychlení větší než zrychlení při uvedení do pohybu. Z pohledu rychlosti se tedy zdá, že zařízení bylo uvedeno do pohybu a po zastavení se zase pohybovalo směrem zpět.

Získání pozice z takového průběhu zrychlení je velmi obtížné. Kvůli dvojité integraci je chyba určení pozice značná a bez implementace filtrovacích algoritmů je takto vypočtená rychlost a pozice nepoužitelná.

3.4.4.1.2.1 Srovnání průběhu zrychlení se zařízením Asus Transformer

Během zkoumání vlastností výstupních dat při pohybu se mi dostalo možnosti provést měření i se zařízením jiného výrobce. Konkrétním zařízením byl tablet Asus Transformer TF300. Tablet je vybaven stejným typem senzorů, jen implementace lineární akcelerace je odlišná a proto jsou výsledky rozdílné.

Měření oběma zařízeními bylo provedeno současně a za stejných podmínek. Zařízení z naprostého klidu byla posunuta o 1 m ve směru jedné osy.



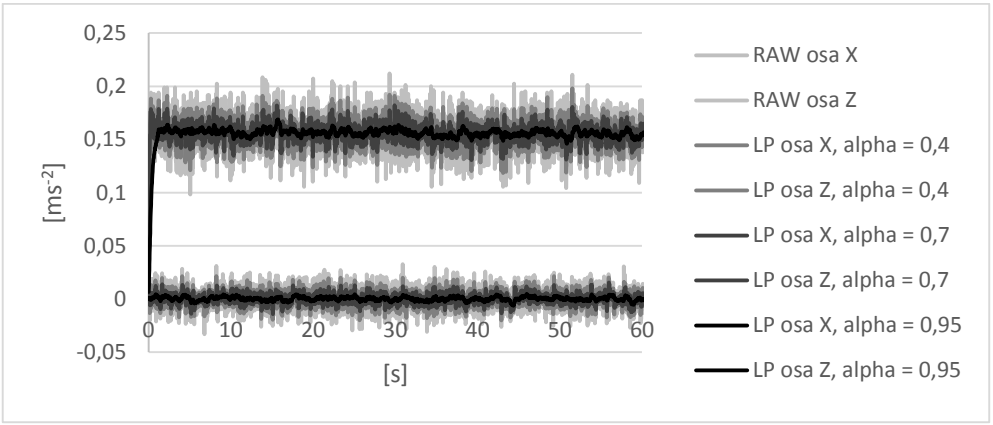
Obrázek 37: Srovnání průběhu zrychlení dvou zařízení

Data byla pro hladší průběh a větší přehlednost přefiltrována jednoduchým *Low Pass* filtrem. Porovnáním výstupu z těchto dvou zařízení je vidět rozdílný průběh hodnot. Tam kde se data zařízení Nexus 7 teprve pozvolna vracejí k nulové hodnotě, data zařízení Transformer žádný takový jev neobsahují. Výzkumným zařízením nicméně byl tablet od společnosti Google a proto další kapitoly a metody zpracování počítají s výstupem z tabletu Nexus 7.

3.4.4.2 Simple Low Pass filter

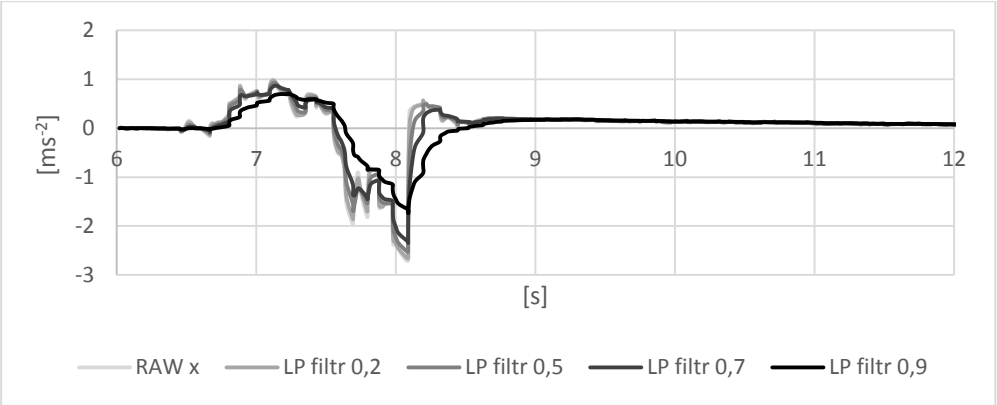
Aplikací tohoto typu filtru získáme hladší průběh signálu. Míra vyhlazení určuje koeficient *alpha*. Pokud je zařízení v klidu, požadovanému chování signálu se přiblížíme použitím vyšší hodnoty koeficientu *alpha*.

Při aplikaci filtru je třeba zohlednit vypočtených nejistot *bias offsetu* v předchozí sekci. Tedy pro dosažení podobné míry vyhlazení na všech osách, je třeba na citlivější osu Z použít vyšší koeficient *alfa* než na osách *X* a *Y*.



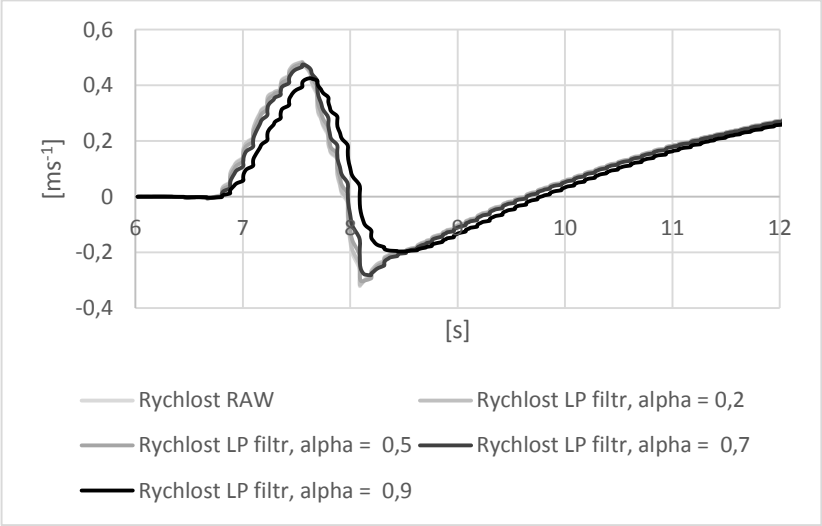
Obrázek 38: Low Pass filter – zrychlení v klidu

Nevýhoda spočívá však v tom, že při těchto vyšších hodnotách koeficientu *alpha* dochází ke zpoždění signálu. V klidovém stavu, kdy se měřené hodnoty mění jen minimálně, to nevadí, ovšem při záznamu pohybu větší zpoždění již způsobuje nepřesnost měření.



Obrázek 39: Low Pass filter - zrychlení při pohybu

Z grafu je vidět, že vysoká hodnota koeficientu *alpha* vyhladí celkový průběh a tam, kde nefiltrovaná data ze senzoru při zastavení pohybu naměří kladný vrchol, od kterého se vrací hodnoty k nule, tam *Low Pass filter* provede vyhlazení. *Offse*t ovšem zůstane zachován a je třeba se ho pokusit řešit jiným způsobem. Na grafu níže je zobrazení vypočtené rychlosti v závislosti na nastavení filtru.

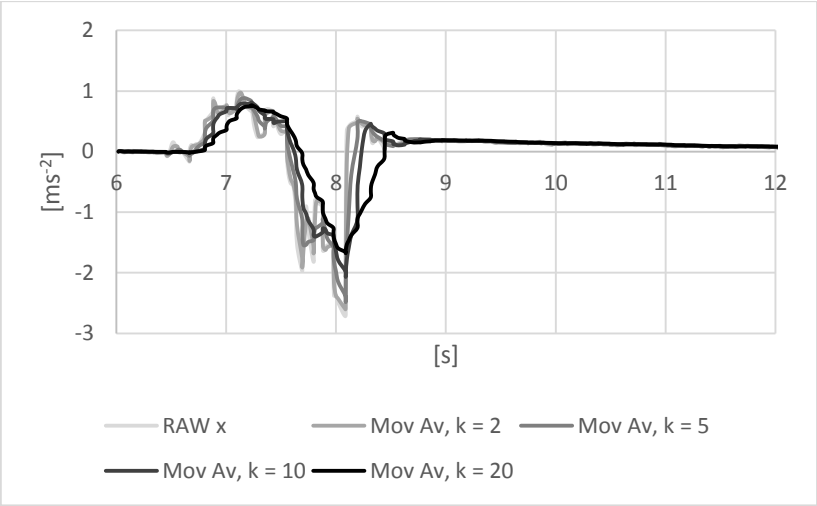


Obrázek 40: Low Pass filter - průběh rychlosti

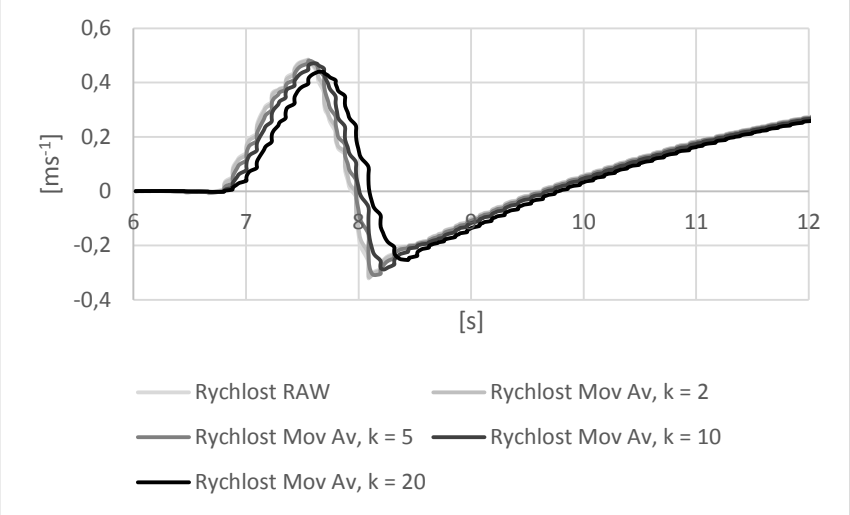
3.4.4.3 Moving Average filter

Aplikace tohoto typu filtru vykazuje podobné výsledky jako předchozí *Low Pass filter*. Filtrovaný signál je opět hladší než signál původní. Vzniklé zpoždění při *k=20* je 300 ms.

Při aplikaci filtru je třeba opět zohlednit vypočtené nejistoty *bias offsetu*. Tedy pro dosažení podobné míry vyhlazení na všech osách, je třeba na citlivější osu *Z* použít vyšší číslo *k* pro zvýšení počtů vzorků k průměrování než na osách *X* a *Y*.



Obrázek 41: Moving Average - průběh zrychlení při pohybu



Obrázek 42: Moving Average - průběh rychlosti

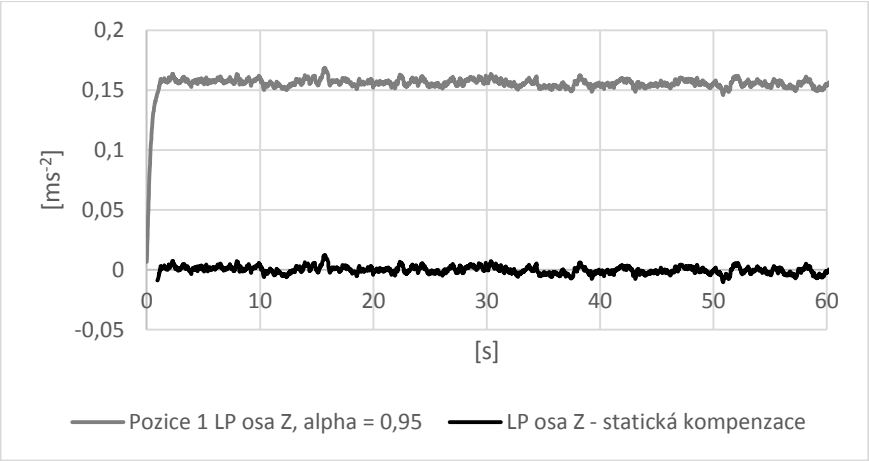
3.4.4.4 Kompenzace offsetu

Na rozdíl od gyroskopu není v případě hodnot z lineárního akcelerometru možnost *bias offset* zanedbat, nýbrž je nutnost ho kompenzovat. V rámci kompenzace jsem prozkoumal dva způsoby kompenzace. Kompenzaci pomocí odečítání statického *offsetu* a kompenzaci pomocí odečítání dynamicky určovaného *offsetu*.

3.4.4.4.1 Kompenzace pomocí odečtení statického offsetu

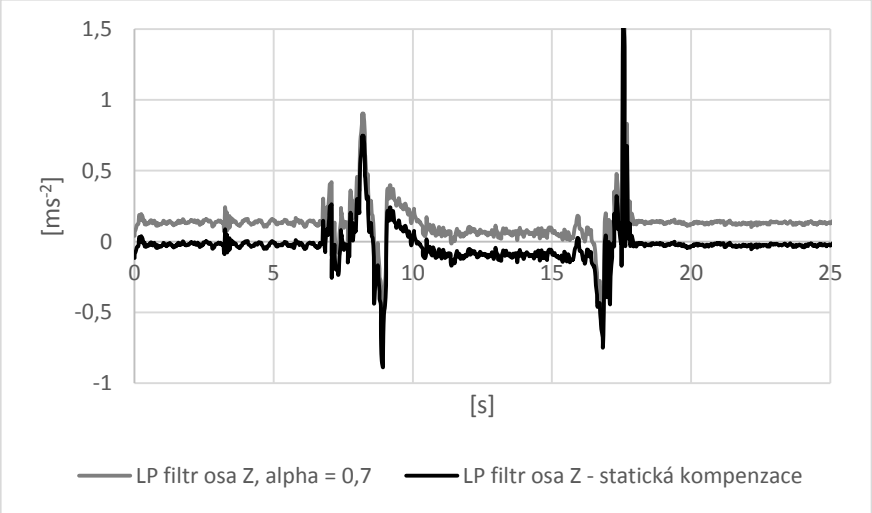
Tato metoda funguje na principu odečtení změřeného *bias offsetu* od měřených hodnot. Ideální funkčnosti dosáhneme, pokud známe orientaci zařízení a během měření zařízení tuto orientaci nemění.

Příkladem úspěšné kompenzace je, když víme, že zařízení leží na vodorovné ploše s displejem směrem vzhůru. Pro takový případ je vypočtený *offset* viz tabulka v 3.4.3.



Obrázek 43: Ukázka statické kompenzace v klidu

V případě že zařízení nemění svoji orientaci, kompenzace funguje bezchybně. Ovšem pokud během měření zařízení podrobíme rotaci, kdy se změní orientace tabletu, změní se i *offset*.

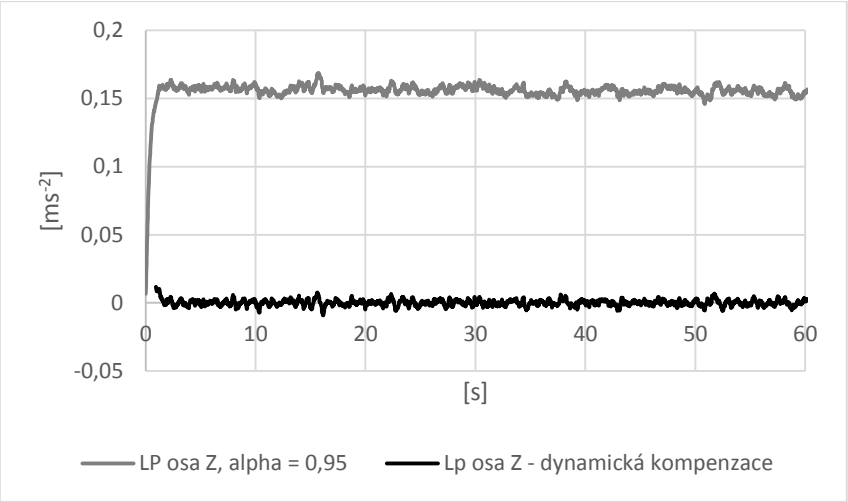


Obrázek 44: Ukázka statické kompenzace při změné orientace

Zařízení během tohoto měření změnilo polohu pomocí rotace okolo osy *Y* o *+90°*. V ní zařízení zůstalo pět vteřin, poté byla opět provedena rotace okolo osy *Y* o *-90°* zpět do původní polohy. V takovém případě statická kompenzace selhává.

3.4.4.4.2 Dynamická kompenzace pomocí plovoucího průměru

Pro dynamickou kompenzaci je použit výpočet *offsetu* pomocí plovoucího průměru. *Bias offset* není pevně určen, nýbrž je vypočítáván pomocí metody *Moving Average*. Počet vzorků *k=25*, neboli výpočet průměru ze vzorků za posledních *500ms*. Tímto způsoben zjištěný *bias offset* se poté odečte od změřené hodnoty.



Obrázek 45: Ukázka dynamické kompenzace v klidu

Kompenzace během chvíle, kdy zařízení nemění svoji orientaci, funguje bezchybně. V takovémto případě dynamická kompenzace dává velmi podobné výsledky jako statická. Odlišná situace nastává, pokud zařízení je v pohybu.

Obrázek 46: Ukázka dynamické kompenzace při změně orientace

V tomto případě dynamická kompenzace funguje výrazně lépe než statická. I při změně orientace je *offset* během půl vteřiny vykompenzován. Nevýhodou je však situace, kdy zařízení během změny rotace podléhá zrychlení. V této situaci kompenzační filtr odečítá nejen *offset*, ale i podstatnou část údaje o zrychlení.

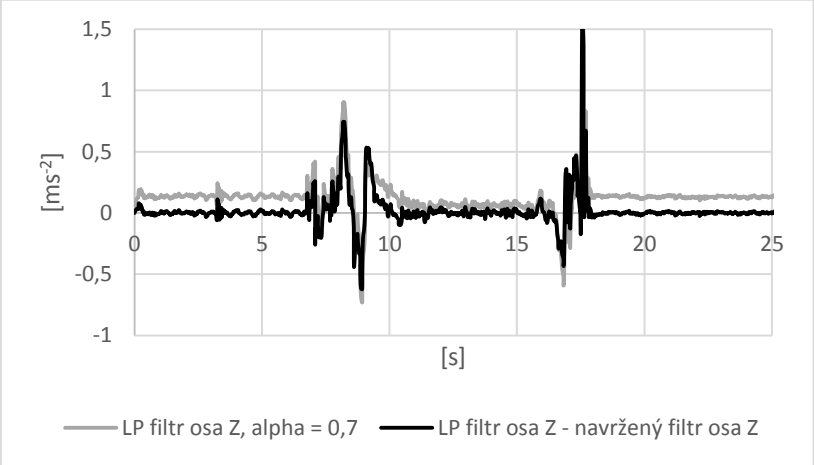
3.4.4.4.3 Navržený kompenzační filtr

Pro současné kompenzování *bias offsetu* a zároveň minimální ovlivnění měřeného zrychlení byl navržen filtr kombinující výhody změřeného statického *bias offsetu* a dynamické kompenzace *offsetu*.

Změřený maximální *bias offset* působící na každou z os se použije pro nastavení *treshold* filtru. Pomocí prahovacího filtru se koriguje maximální velikost dynamické kompenzace. Hranice filtrace pro jednu osu se určí jako:

$$hranice_i = maxBiasOffset_i + rozšířenáNejistotaBiasu_i \qquad (21)$$

kde, $maxBiasOffset_i$ je hodnota *offsetu* naměřená pro použitou osu. Pokud výpočet plovoucího průměru pro dynamickou kompenzaci překročí tuto hranici, nastaví se kompenzace na hodnotu $maxBiasOffset_i$.



Obrázek 47: Ukázka navržené dynamické kompenzace offsetu při změně orientace

Díky tomuto je zaručeno, že maximální hodnota kompenzace nepřekročí práh *bias offsetu* a nedojde ke ztrátě podstatné části informace o zrychlení.

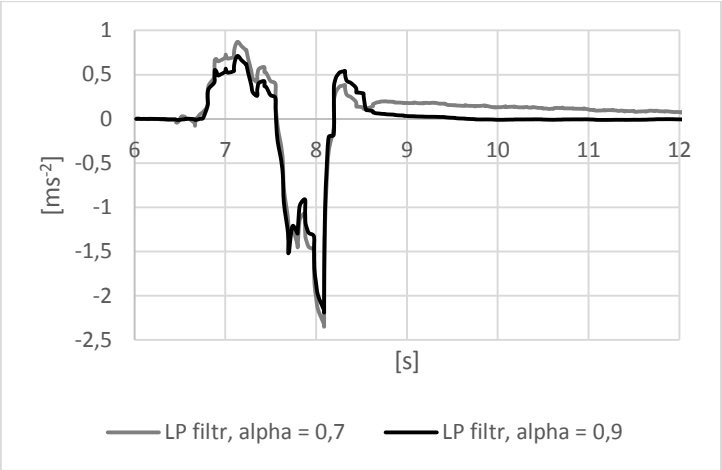
3.4.4.5 Treshold filtr

Použití jednoduchého *hard treshold* filtru není v tomto případě tak jednoduché jako u gyroskopu, neboť hodnoty ze senzoru lineární akcelerace jsou silně zatíženy *bias offsetem*. Pro správné filtrování dat je tedy nutné tento *offset* nejprve vykompenzovat.

Treshold filtr se již používá pro kompenzaci *offsetu*, tím ovšem jeho využití nekončí. Další aplikací je opět porovnání, zda změřená hodnota patří do intervalu nastaveným hraniční hodnotou. Pokud leží výsledná dat v tomto intervalu, který je experimentálně určen, změní se *alpha* vyhlazovacího filtru na *alpha = 0,9*. Tím dojde k většímu vyhlazení hodnot, u kterých se předpokládá, že nedochází ke změně pohybu, sníží se rušení vlivem vibrací a zároveň nedojde ke zvýšení zpoždění signálu.

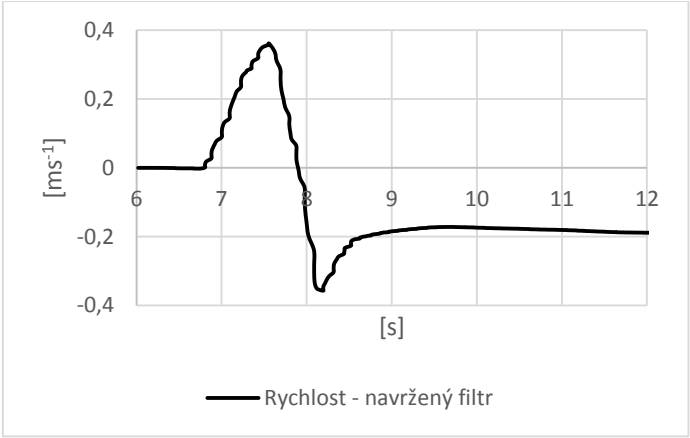
3.4.4.6 Navržený filtr

Implementace filtrování vstupních dat ze senzoru lineární akcelerace se skládá z navrženého filtru kompenzace *offsetu* a prahem ovládaným *Low Pass* filtrem. Hodnoty vyfiltrované tímto filtrem jsou ve velké míře oproštěny od zmíněných jevů, ovšem stále nejsou ideální.



Obrázek 48: Průběh zrychlení při posunu o 1 m po vodorovné podložce s navrženým filtrem

Příkladem budiž vypočtená rychlost, která svým průběhem připomíná spíše průběh zrychlení.



Obrázek 49: Graf vypočtené hodnoty s aplikací navrženého filtru

Obrázek 50: Graf vypočtené pozice po aplikaci navrženého filtru

3.5 Fúze dat z výstupu gyroskopu a senzoru lineární akcelerace

Vyfiltrovaná data lineární akcelerace se skrze senzorovou fúzi transformují pomocí dat z gyroskopu. K transformaci se využívá výpočet pomocí *quaternionů* zmíněných v kapitole XY.

Metoda pro transformaci je implementována ve třídě *SensorFusion*. Výpočet popsaný rovnicí XY lze pomocí algoritmizace implementovat tímto způsobem:

```
double [] akcelPoleQuat;
private double [] akcel = quatNasobeni(quatNasobeni(qgyro, akcelPoleQuat), quatKomplex(qgyro));

private double [] quatNasobeni(double [] b, double [] a){
    double [] nasobeni = new double [4];
    nasobeni[0] = a[0]*b[0]-a[1]*b[1]-a[2]*b[2]-a[3]*b[3];
    nasobeni[1] = a[0]*b[1]+a[1]*b[0]+a[2]*b[3]-a[3]*b[2];
    nasobeni[2] = a[0]*b[2]-a[1]*b[3]+a[2]*b[0]+a[3]*b[1];
    nasobeni[3] = a[0]*b[3]+a[1]*b[2]-a[2]*b[1]+a[3]*b[0];
    Log.d("SensorFusion", "quatNasobeni");
    return nasobeni;
}
/**Metoda pro vytvoreni komplexne zdruzeneho quaternionu  */
private double [] quatKomplex(double [] a){
    double [] complex = new double [a.length];
    complex[0] = a[0];
    complex[1] = -a[1];
    complex[2] = -a[2];
    complex[3] = -a[3];
    Log.d("SensorFusion", "quatComplex");
    return complex;
}
```

Obrázek 51: Ukázka algoritmu pro transformaci zrychlení

Metodě je předána změřená rotace v podobě *quaternionu* a s její pomocí se vynásobí vyfiltrovaná lineární akcelerace, která je zapsána v *quaternionové* formě jako:

$$a_q = \{0;\ a_x;\ a_y;\ a_z\} \tag{22}$$

Vrácený vynásobený výsledek je dále vynásoben komplexně sdruženým *quaternionem* rotace. Výsledkem je transformovaný vektor lineárního zrychlení z lokální soustavy souřadnic do globální soustavy.

3.6 Určení rychlosti a pozice

Výpočet provádějící určování rychlosti a pozice se provádí z vyfiltrovaného a transformovaného lineárního zrychlení.

Výpočet rychlosti je integrálem zrychlení ve spojitém čase, neboli:

$$v = \int a\ dt \tag{23}$$

Jelikož sbíráme vzorky ze senzorů, tudíž operujeme v diskrétním čase, výpočet rychlosti se změní z integrálu na součet přírůstků rychlosti, neboli:

$$v_n = \sum_{i=0}^{n}(a_i * \Delta t) \tag{24}$$

Výpočet vzdálenosti je integrálem rychlosti ve spojitém čase, potažmo dvojitým integrálem zrychlení:

$$s = \int v\ dt = \iint a\ dt \tag{25}$$

Použitá podoba vzorce pro diskrétní čas:

$$d_n = \sum_{i=0}^{n}(v_i * \Delta t + \frac{1}{2} a_i \Delta t^2) \quad (26)$$

Jak již bylo několikrát zmíněno, je třeba se snažit snížit chyby měření na co nejnižší úroveň, neboť díky dvojitému integrálu i malá chyba roste kvadraticky s časem a způsobuje drift pozice v čase.

3.7 Algoritmus přepočtení změřené pozice na body GPX

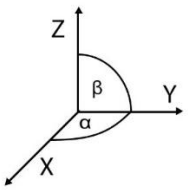
Získání nových zeměpisných souřadnic probíhá pomocí algoritmu, který načítá data z inerciální jednotky do záznamového pole mezi jednotlivými přijatými GPS pozicemi.

Jakmile *DataService* přijme první update GPS pozice, vyšle signál třídě *SensorFusion* k vynulování předchozí změřené pozice a rychlosti. Od tohoto okamžiku probíhá záznam nových vypočtených pozic z inerciální jednotky až do další aktualizace GPS.

Při přijmutí nové GPS pozice, je třídě *SensorFusion* předáno záznamové pole s dosud naměřenými pozicemi z IMU společně s údaji o současné a předchozí pozici z GPS.

Algoritmus zpracování má dvě fáze. V první fázi se z pole změřených pozic jednotkou IMU určí celková uražená vzdálenost. Poté se jednotlivé pozice tou to uraženou vzdáleností vydělí a následně vynásobí údajem o vzdálenosti z jednotky GPS. Tímto způsobem dojde ke srovnání výsledků z obou senzorů Dalším krokem je určení směru trasy neboli změřeného azimutu. Tento krok je důležitý pro správné sloučení GPS souřadnic. Při výpočtu se předpokládá, že zařízení bylo na začátku nastaveno do výchozí pozice, tedy osa *X* a *Y* jsou rovnoběžné s povrchem Země.

Úhel trasy α je vypočten z posledního záznamu v poli pozic a je vypočítán jako úhel, který svírá pozice s osou *Y*. Tento úhel je klíčovým pro správné určení azimutu všech dalších pozic. Pro výpočet nadmořské výšky je potřeba vypočíst elevaci, *heading* trasy. Úhel β je definován v rovině YZ jako úhel, který svírá cesta s osou *Y*.



Obrázek 52: Úhly použité pro výpočet nové souřadnice

V druhé fázi algoritmu jsou modifikované záznamy pozice jednotlivě předávány k výpočtu nového bodu zeměpisných souřadnic. Výpočet je založen na algoritmu dostupném ze stránek *Movable Type Scripts* [35].

Pro novou souřadnici je třeba znát souřadnice výchozího bodu, vzdálenost, azimut a elevaci. Vzdáleností je přepočtená pozice. Výsledný azimut *a* použitý pro výpočet je rozdílem azimutu získaného z GPS a rozdílem úhlů mezi současným vzorkem pozice a posledním vzorkem pozice.

$$a = a_{GPS} - \propto_{last} - \propto_i \quad (27)$$

Nadmořská výška h_i se získá podobným způsobem. Algoritmu se předá hodnota nadmořské výšky z nového bodu GPS. Je vypočten rozdíl v úhlech β mezi posledním a vybraným vzorkem pozic. Výsledný úhel se vynásobí uraženou vzdáleností a výsledek se odečte od nadmořské výšky změřené pomocí GPS.

$$\beta = \beta_{last} - \beta_i \tag{28}$$

$$h_i = h_{GPS2} - \tan\beta * d_i \tag{29}$$

Jakmile výpočet proběhne úspěšně, pole nových souřadnic je uloženo v třídě *DataService* a dojde k vynulování změřené pozice a zrychlení a k nastavení výstupní rychlosti jednotky IMU na hodnotu získanou z GPS. Díky tomuto nulování a korekci rychlosti je možné kompenzovat integrační *drift* ovlivňující měření jednotky IMU. Záleží však na četnosti aktualizace GPS. Čím větší interval mezi jednotlivými pozicemi, tím méně přesná bude změřená poloha pomocí IMU.

Díky implementaci tohoto algoritmu je možné získat mezi dvěma GPS pozicemi padesát nových zeměpisných souřadnic každou uplynulou vteřinu. Pokud tedy předpokládáme interval mezi dvěma GPS pozicemi jednu vteřinu, díky tomuto algoritmu získáme informace o padesáti nových pozicích, tedy získáme novou souřadnici každých 20 ms.

Přesnost dat se odvíjí od samotné implementace inerciální jednotky. Pro relevantní pozici je tedy třeba důkladně odladit navrženou jednotku IMU. Ukázky použitého algoritmu lze nalézt v příloze.

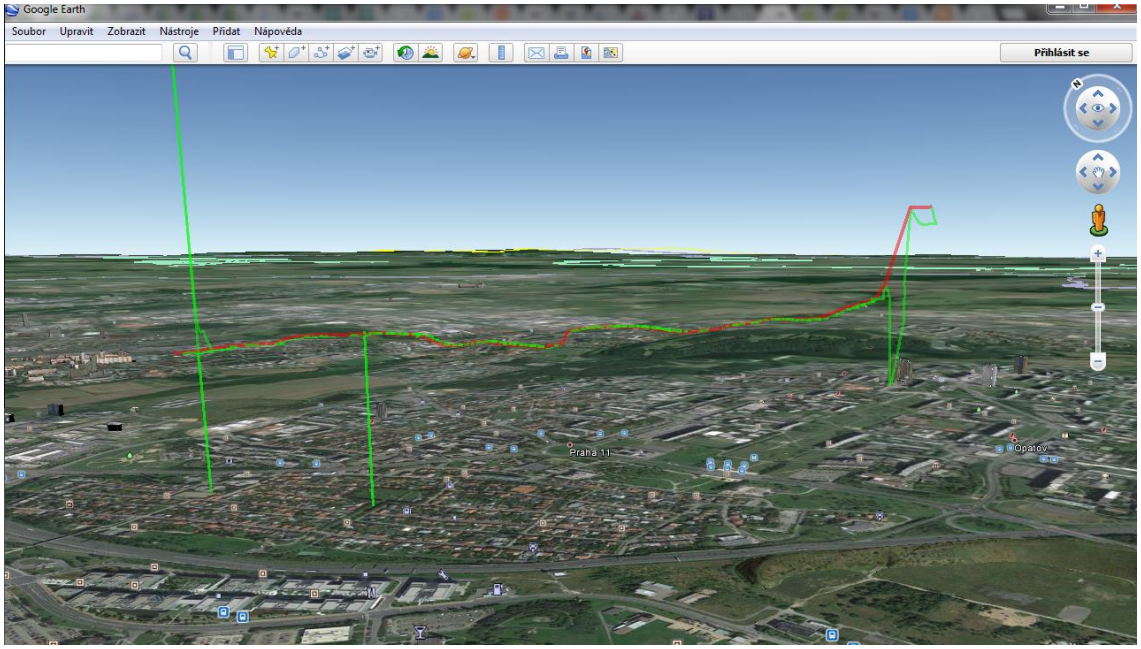
3.7.1 Ukázka vypočtených nových pozic kombinací IMU a GPS

Standartní nastavení při exportu souřadnic z jednotky IMU a GPS je zobrazení v absolutní nadmořské výšce. Data jsou vykreslena při zobrazení do prostoru. Pro analýzu a lepší práci s daty doporučuji v softwaru *Google Earth* zapnout ve vlastnostech trasy volbu *Připevněno k zemi*.

Na obrázcích níže je zobrazeno měření pěší trasy pomocí vypočtené pozice z IMU a změřené pozice z GPS.

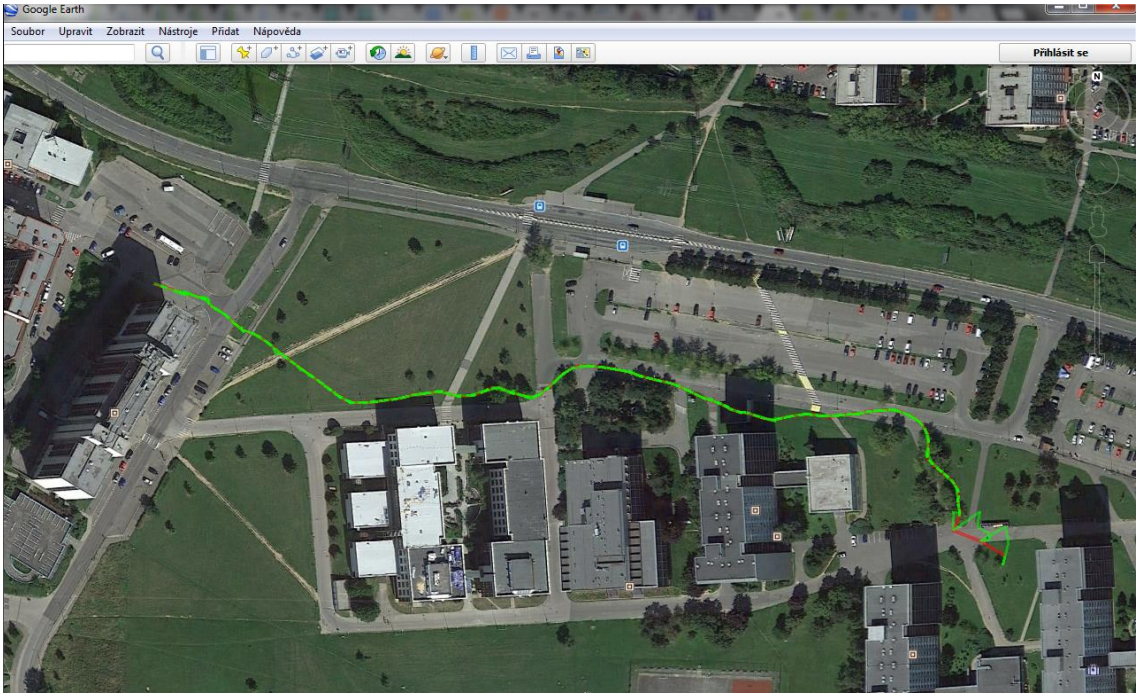
Trasa ze senzoru GPS je na snímcích obrazovky vykreslena červenou barvou, zatímco výstup z jednotky IMU je vykreslen barvou zelenou.

Kvůli chybám způsobených nevhodnou implementací jednotky IMU vypočtená pozice není ideální.



Obrázek 53: Změřená trasa zobrazena i s vypočtenou nadmořskou výškou

Největší chybou je zatížen výpočet nadmořské výšky. Dáno je to neideální kompenzací offsetu, který působí na osu měřící vertikální zrychlení. Největší chyba nastává, pokud se zařízením stojíme na místě a provádíme s ním rotaci. Na obrázku výše taková situace vyvstala při umístění a vyjmutí tabletu z tašky a umístěním tašky na záda.



Obrázek 54: : Změřená trasa pomocí aplikace Google Earth připevněná na zemský povrch

Poloha změřená pomocí implementované jednotky IMU je méně přesná než poloha z GPS. Přesnost závisí na velikosti intervalu mezi přijatými pozicemi z GPS. Čím delší interval, tím větší odchylka nastane.

Pro dosažení přesného záznamu trasy s přesností na desítky centimetrů je třeba dalšího výzkumu a vylepšení implementace polohové jednotky IMU.

4 Závěr

Cílem mé bakalářské práce bylo zpracování problematiky inerciálního určování polohy, senzorové fúze, osvojení základů programování pro systém Android a vytvoření demonstrační aplikace.

V první části bakalářské práce jsem se zaměřil na zpracování teoretických předpokladů a principu fungování systémů inerciální navigace. Praktická část se zabývá analýzou možnosti využití implementovaného senzoru lineární akcelerace pro návrh vlastní inerciální polohové jednotky (IMU) a vypracováním příslušných filtrů.

První problémem je samotný senzor lineární akcelerace, neboť v mobilních zařízeních zpravidla není hardwarového typu, ale jde o senzor tzv. virtuální, tedy udává softwarově vyfiltrované lineární zrychlení z výstupních hodnot senzoru akcelerace. Pro způsob implementace nevydává společnost Google stojící za systémem Android žádná doporučení, použití příslušných algoritmů je tedy na výrobci. Z toho důvodu se naměřená data mohou zařízení od zařízení lišit. Dále naměřená data jsou obtížně interpretovatelná kvůli zatížení *bias offsetem* a naměřeným průběhem hodnot po ukončení pohybu, kdy dochází k ustalování nulové hodnoty průměrně po dobu 4 s.

Výstupní data z gyroskopu ovlivňuje chyba zvaná *bias drift*, kterou však navržený filtr umožňuje kompenzovat.

Algoritmus pro sloučení výstupní pozice z jednotky IMU a ze senzoru GPS funguje správně a vypočítává nové souřadnice každých 20 ms, na rozdíl od četnosti aktualizací polohy pomocí GPS, kdy nejkratší dosažený interval za ideálních podmínek trvá 1 s.

Prototyp vytvořené aplikace může sloužit pro další výzkum, měření a testování implementace polohové inerciální jednotky.

Bakalářská práce slouží jako úvod do problematiky inerciální navigace a prozkoumává možnost využití senzoru lineární akcelerace pro návrh jednotky IMU. Na základě této práce je možný další výzkum v oblasti inerciálních polohových jednotek a vypracování pokročilejších metod filtrování dat.

V budoucí navazující práci bych se chtěl uvedenou problematiku dále zabývat a prozkoumat možnosti využití dalších senzorů pro vytvoření funkčního inerciálního systému.

5 Seznam použité literatury

[1] Kovach S.: How Android Grew To Be More Popular Than The iPhone [online], dostupné z: http://www.businessinsider.com/history-of-android-2013-8 [cit 2014-05-22]

[2] BloombergBusinessweek: Google Buys Android for Its Mobile Arsenal [online], dostupné z: http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal [cit 2014-05-22]

[3] Arthur Ch.: Andy Rubin moved from Android to take on 'moonshots' at Google [online], dostupné z: http://www.theguardian.com/technology/2013/mar/13/andy-rubin-google-move [cit 2014-05-22]

[4] Mysliveček D.: Krátké ohlédnutí za historií Androidu [online], dostupné z: http://www.svetandroida.cz/kratke-ohlednuti-za-historii-androidu-201305 [cit 2014-05-22]

[5] Open Handset Alliance [online], dostupné z: http://www.openhandsetalliance.com/oha_members.html [cit 2014-05-22]

[6] Smrček J.: Google Android – velký výlet do historie [online], dostupné z: http://www.cnews.cz/google-android-velky-vylet-do-historie [cit 2014-05-22

[7] Open Handset Alliance: Industry Leaders Announce Open Platform for Mobile Devices [online], dostupné z: http://www.openhandsetalliance.com/press_110507.html [cit 2014-05-22]

[8] Kilián K.: Android 4.0 ICS s 15,9 % předběhl Froyo a je druhý [online], dostupné z: http://www.svetandroida.cz/android-4-0-ics-s-159-predbehl-froyo-a-je-druhy-201208 [cit 2014-05-22]

[9] Google Developers: Platform versions [online], dostupné z: http://developer.android.com/about/dashboards/index.html [cit 2014-05-22]

[10] Yarow J.: Android Now Controls A Breathtaking 80% Of The Smartphone Market [online], dostupné z: http://www.businessinsider.com/android-now-controls-a-breath-taking-80-of-the-smartphone-market-2013-8 [cit 2014-05-22]

[11] Schupppler P.: Mobilní aplikace na platfromě Anroid pro MZK [online], dostupné z: https://is.muni.cz/th/255645/fi_b/bc.pdf [cit 2014-05-22]

[12] Vogel L.: Android Development - Tutorial [online], dostupné z: http://www.vogella.com/tutorials/Android/article.html [cit 2014-05-22]

[13] Google Developers: Application Fundamentals [online], dostupné z: https://developer.android.com/guide/components/fundamentals.html [cit 2014-05-22]

[14] Google Developers: Android NDK [online], dostupné z: http://developer.android.com/tools/sdk/ndk/index.html [cit 2014-05-22]

[15] Mohinder S. Grewal, Mohinder S.Lawrence R. Global Positioning Systems, Inertial Navigation, and Integration [online]. 1st ed. New York, NY [u.a.]: Wiley, 2000 [cit. 2014-05-23]. ISBN 04-712-0071-9

[16] Official U.S. Government information about the Global Positioning System (GPS) and related topics [online], dostupné z: http://www.gps.gov/systems/gps/control/ [cit 2014-05-22]

[17] Veškrna M.: Positioning system for small devices using principles of inertial navigation system [online], dostupné z: http://is.muni.cz/th/256569/fi_m/Masters_thesis.pdf [cit 2014-05-22]

[18] VECTORNAV Embedded Navigation Solution [online], dostupné z: http://www.vectornav.com/support/library?id=76 [cit 2014-05-22]

[19] S. Drapper C., Wrigley W., Hoang G., H. Batiin R., E. miller J., A. Koso D., L. Hopkis A., E Vander Velde W.: Apollo Guidance and Navigation [online], dostupné z: http://web.mit.edu/digitalapollo/Documents/Chapter5/r500.pdf [cit 2014-05-22]

[20] J. Woodman O.: An introduction to inertial navigation [online], dostupné z: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf [2014-05-22]

[21] D. King A.: Marconi Electronic Systems Ltd.: International Navigation - Forty Years of Evolution [online], dostupné z: http://www.imar-navigation.de/downloads/papers/inertial_navigation_introduction.pdf [cit 2014-05-22]

[22] GoogleTechTalks, David Sachs: Sensor Fusion on Android Devices: A Revolution in Motion Processing [online], dostupné z: http://www.youtube.com/watch?v=C7JQ7Rpwn2k [cit 2014-05-22]

[23] YouTube.com kanál Sciencedidgood: Hand Calculation of Quaternion Rotation [online], dostupné z: http://www.youtube.com/watch?v=8gST0He4sdE [cit 2014-05-22]

[24] Shala U., Rodrigues A.: Indoor Positioning using Sensor-fusing in Android Devices [online], dostupné z: http://hkr.diva-portal.org/smash/get/diva2:475619/FULLTEXT02.pdf [2014-05-22]

[25] Google Developers: Sensors Overview [online], dostupné z: http://developer.android.com/guide/topics/sensors/sensors_overview.html [2014-05-22]

[26] Google Developers: Motion Sensors [online], dostupné z: http://developer.android.com/guide/topics/sensors/sensors_motion.html [2014-05-22]

[27] Google Developers: Location APIs [online], dostupné z: http://developer.android.com/google/play-services/location.html [2014-05-22]

[28] Liggins, Martin E, David L HALL a James LLINAS. Handbook of multisensor data fusion: theory and practice [online]. 2nd ed. Boca Raton, FL: CRC Press, c2009, xx, 849 p. [cit. 2014-05-23]. ISBN 14-200-5308-6.

[29] Shala U., Rodrigues A.: Inddor Positioning using Sensor-fusing in Android Devices [online], dostupné z: http://hkr.diva-portal.org/smash/get/diva2:475619/FULLTEXT02.pdf [2014-05-22]

[30] UC3M: A system that improves the precision of GPS in cities by 90 percent [online], dostupné z: http://portal.uc3m.es/portal/page/portal/actualidad_cientifica/noticias/improves_precision_gps [2014-05-22]

[31] Soubra O.: Hands On: Project Tango, Google's 3D-Scanning Phone for Makers [online], dostupné z: http://makezine.com/2014/03/20/hands-on-project-tango-googles-3d-scanning-phone-for-makers [2014-05-22]

[32] Google: Project Tango[online], dostupné z: https://www.google.com/atap/projecttango/ [2014-05-22]

[33] John Lee: Project Tango demo at GPU Technology Conference 2014 [online], dostupný z: http://www.youtube.com/watch?v=iiM4X7wW5C0 [2014-05-22]

[34] Gavin Kistner: Frame Rate-Independent Low-Pass Filter [online], dostupné z: http://phrogz.net/js/framerate-independent-low-pass-filter.html [2014-05-22]

[35] Movable Type Scripts: Calculate distance, bearing and more between Latitude/Longitude points [online], dostupné z: http://www.movable-type.co.uk/scripts/latlong.html#destPoint [2014-05-22]