FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	3D lokalizace bezdrátových modulů v reálném čase
Student:	Bc. Adam Kubišta
Vedoucí:	Ing. Alexandru Moucha, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačové systémy a sítě
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Navrhněte a otestujte lokalizaci mnoha nezávislých bezdrátových modulů ve 3D prostoru v reálném čase pomocí

1) akcelerometrů,
2) triangulace pomocí bezdrátových technologií, jako například ZigBee,
3) jejich kombinace.

Výsledky testů porovnejte a zhodnoťte. Je možné, že testování bude vyžadovat kalibraci. V tomto případě je dostatečná jedna instance tohoto problému (jedna místnost).

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 4. prosince 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

3D lokalizace bezdrátových modulů v reálném čase

Bc. Adam Kubišta

Vedoucí práce: Ing. Alexandru Moucha, Ph.D.

7. ledna 2019

Poděkování

Chci poděkovat doktorovi Mouchovi za konzultace a zapůjčení hardwarových čipů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. ledna 2019 .....

Odkaz na tuto práci

Kubišta, Adam. *3D lokalizace bezdrátových modulů v reálném čase.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce definuje problém 3D lokalizace v reálném čase, navrhuje algoritmy pro definovanou lokalizaci a navrhuje testy, kterými ověří jakost u navržených algoritmů.

V praktické části tyto testy provádí a v poslední řadě vyhodnocuje. Součástí je také systém, který je schopný lokalizovat v reálném čase nebo pořizovat záznamy k dalšímu zpracování.

Práce je zaměřená na laciné a dostupné senzory.

Klíčová slova lokalizace, reálný čas, akcelerometr, bezdrátové sítě, levný hardware

Abstract

This thesis defines 3D localisation problem in realtime, suggests algorithms for defined localisation and according tests to prove their quality.

In the practical part, the tests are done and analyzed. Byproduct of this thesis is also a localising software capable of realtime processing or recording for furhter analysis.

Thesis aims for low cost hardware.

Keywords localisation, realtime, accelerometer, wireless network, low cost hardware

Obsah

Seznam obrázků

Seznam tabulek

Seznam definic a vět

Úvod

Lokalizace je problém, který má smysl řešit. Například lokalizace GPS při použití navigace je nezbytnou součástí a tento problém určitým způsobem řeší. Dále se například v poslední době objevují aplikace a hardware pro virtuální realitu, která potřebuje znát pozici ovladačů v prostoru a taky problém lokalizace řeší. Jenže tato řešení jsou cenově ne uplně dostupná a mají specifické využití a předpoklady.

Levnější řešení by se dalo například použít pro virtualizaci sportovních her: pokud by bylo možné rozdat malé čipy jednotlivým hráčum, mohl by pak divák sledovat jejich virtualizaci v reálném čase. Například Laser game[1] se častokrát hraje v uzavřené aréně a diváci na baru vidí akorát skóre jednotlivých hráčů namísto kompletního přehledu pozic v reálném čase.

Pokud by byla lokalizace velice přesná, bylo by možné zkoumat detailně záznamy a případně vytvářet na základě toho strategie.

Využití levného řešení by se dalo najít i více.

Tato práce se na levnou lokalizaci zaměřuje.

Kapitola 1

Cíl práce

Práce si klade za cíl otestovat, zda-li jsou cenově dostupné senzory vhodné k použití pro řešení problému lokalizace.

Nejprve analyzuje zadání a definuje problém lokalizace. Dále analyzuje zvolený hardware z hlediska limitací a API, navrhuje způsob realizace jednotlivých instancí problému, navrhuje algoritmy, které využívají senzorických dat pro určení pozice jednotlivých modulů, navrhuje systém, který je schopný pozice modulů sledovat v reálném čase a pořizovat či přehrávat záznam jednotlivých instancí problému a v poslední řadě navrhuje testy, které prokáží použitelnost zvoleného hardwaru.

V praktické části práce jsou jednotlivé návrhy implementovány a prováděny testy. Závěrem je pak vyhodnocení těchto testů a odpověď na otázku vhodnosti laciného hardwaru.

Kapitola 2

Analýza

2.1 Zadání práce

Navrhněte a otestujte lokalizaci mnoha nezávislých bezdrátových modulů ve 3D prostoru v reálném čase...

Problém lokalizace bude definovván po rozboru zadání.

Mnoha moduly je chápáno dva a více. V práci jsou použity dva, ale systém bude navržen na libovolný počet - dle limitací hardwaru. Jako platforma je zvoleno Raspberry Pi 3 B V1.2[2] s operačním systémem Raspbian[3].

Bezdtrátovosti modulů je porozuměno jako využití bezrdrátových počítačových sítí a to dokonce i nezávislosti na síti elektrické. Pro jednoduchost budou moduly zapojené do elektrické sítě, ale systém bude fungovat stejně, jako by moduly byly napájeny bateriemi.

3D prostorem je míněna volnost pohybu ve třech dimenzích. Reálným časem systém, který lokalizuje moduly podle posledních dat do velmi krátké doby - řádově milisekundy.

...pomocí

1. **akcelerometrů,**
2. **triangulace pomocí bezdrátových technologií, jako například ZigBee,**
3. **jejich kombinace...**

Výsledky testů porovnejte a zhodnoťte. Je možné, že testování bude vyžadovat kalibraci. V tomto případě je dostatečná jedna instance tohoto problému (jedna místnost).

Jako akcelerometry jsou použity čipy MPU6050[4] v komponentě GY-521[5], jako bezdrátové čipy jsou použity XBee Series 2[6].

Nároky na přesnost lokalizace se v testech budou postupně zvyšovat a je možné, že při špatných výsledcích nebude dávat smysl dělat pokročilejčí testy - např. kombinace akcelerometrů a XBee.

2.2 Problém lokalizace

Problematika lokalizace se zabývá určením polohy hledaného modulu.

Každá instance problému se odehrává v místnosti, která má svoji soustavu souřadnic - počátek a ortonormální bázi o dimenzi 3.

U každého modulu lze sledovat minimálně 3 atributy - polohu v ose X, Y a Z. Poloha je v soustavě souřadnic místnosti. Navíc díky gyroskopům lze také sledovat orientaci modulu skrze 3 atributy - rotační úhly v ose X, Y a Z. Rotační úhly jsou v soustavě souřadnic určené senzorem[4].

Pro následující definice platí, že čas $t = t_0 = 0$ znamená počátek libovolného testu lokalizace. Čas je z oboru reálných čísel a nabývá hodnot z intervalu $[0; \infty)$. V implementaci této práce dokonce platí, že $t_i = i * dt$, kde i je index snímku (index je určen pořadím šestice hodnot z akcelerometru a gyroskopu) a dt je konstanta určená nastavením senzoru dle frekvence snímání hodnot.

Definice 2.2.1 (Opravdová poloha $Pr(t_i)$)**.** *Nazvěme $Pr(t_i)$ jako opravdovou polohu sledovaného modulu v čase t_i od začátku testu. $Pr(t_i)$ je funkce času, jejímž obrazem je skutečná pozice (trojice souřadnic v soustavě souřadnic testovací místnosti) sledovaného modulu v čase t_i. Jednotlivé členy trojice navýbvají hodnot z oboru reálných čísel v intervalu $(-\infty; \infty)$.*

Definice 2.2.2 (Opravdové rotační úhly $Or(t_i)$)**.** *Nazvěme $Or(t_i)$ jako opravdové rotační úhly sledovaného modulu v čase t_i od začátku testu. $Or(t_i)$ je funkce času, jejímž obrazem jsou skutečné pravotočivé rotační úhly (trojice pravotočivých úhlů v soustavě souřadnic senzoru) sledovaného modulu v čase t_i. Jednotlivé členy trojice navýbvají hodnot z oboru reálných čísel v intervalu $(-360; 360)$. Trojice je v jednotkách $^\circ$.*

Definice 2.2.3 (Vypočítaná poloha $P(t_i)$)**.** *Nazvěme $P(t_i)$ jako vypočítanou polohu sledovaného modulu v čase t_i od začátku testu. $P(t_i)$ je funkcí času jejímž obrazem je vypočítaná poloha (trojice souřadnic v soustavě souřadnic testovací místnosti) určená počáteční polohou modulu ($P(t_0) = Pr(t_0)$) a zpracovanou sekvencí dat ze senzorů dle volitelného algoritmu. Jednotlivé členy trojice navýbvají hodnot z oboru reálných čísel (v standardní 64bitové float reprezentaci) v intervalu $(-\infty; \infty)$.*

Definice 2.2.4 (Vypočtené rotační úhly $O(t_i)$)**.** *Nazvěme $O(t_i)$ jako vypočítané pravotočivé rotační úhly sledovaného modulu v čase t_i od začátku testu. $O(t_i)$ je funkcí času jejímž obrazem jsou vypočítané pravotočivé úhly (trojice pravotočivých úhlů v soustavě souřadnic senzoru) určené počáteční rotací ($O(t_0) = Or(t_0)$) a zpracovanou sekvencí dat ze senzorů dle algoritmu v čase*

t_i. Jednotlivé členy trojice navýbvají hodnot z oboru reálných čísel (v standardní 64bitové float reprezentaci) v intervalu $(-360; 360)$*. Trojice je v jednotkách* $^\circ$*.*

Algoritmy budou označeny za úspěšné, bude-li $P(t_i) \simeq Pr(t_i)$ nebo $O(t_i) \simeq Or(t_i)$ pro t_i na konci jednotlivých testů.

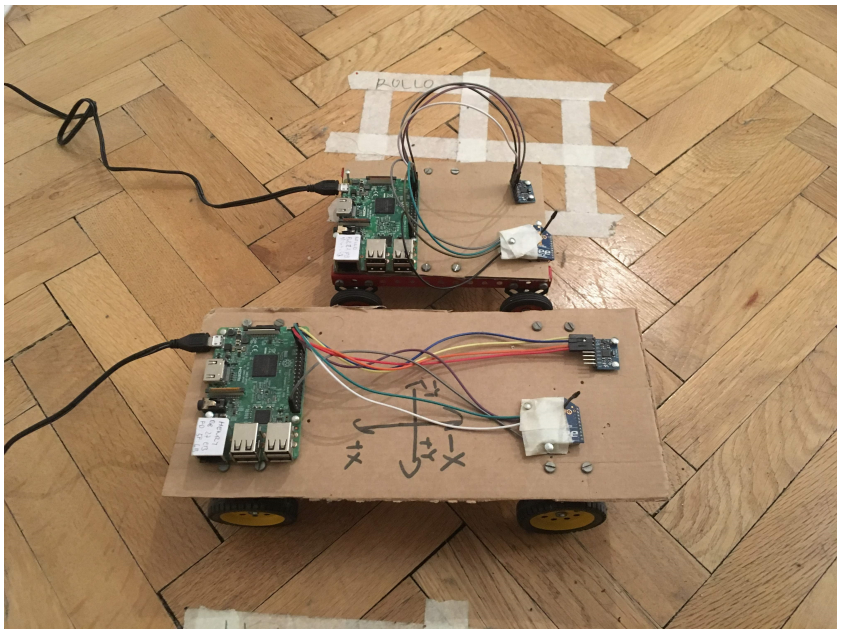
2.3 Zvolený hardware a software

2.3.1 Server

Jako servery jsou využity standardní domácí počítače nebo laptopy s operačním systémem Windows 10[7].

2.3.2 Lokalizovaný modul

Pro bezdrátový modul bylo zvoleno Raspberry Pi 3 B V1.2[2] které je cenově dostupné a díky standardnímu operačnímu systému (Raspbian[3]) je vhodné na tuto práci z hlediska jednoduchosti implementace softwaru. Dále také díky GPIO[1] pinům na desce Raspberry Pi bude jednodušší zprovoznení komunikačních protokolů I2C[2][8] a UART[3][9] pro senzory, které je používají.



Obrázek 2.1: Lokalizované moduly

[1]General Purpouse Input Output
[2]Inter Integrated Circuit
[3]Universal Asynchronous Input Output

2.3.3 Akcelerometr a gyroskop

Zvolený MEMS[4][4] čip produkuje sadu aktuálních hodnot s volitelnou frekvencí. Využívá interního osciloskopu schopného kmitat v různých volitelných frekvencích mezi 1 Hz až 8 kHz. Akcelerační a gyroskopické senzory jsou odděleny. Akcelerační senzory využívají gyroskopického oscilátoru a to od frekvence 1 kHz a méně. Senzorická data jsou nezávisle měřena ve 3 ortogonálních osách jako snímek každou periodu. Tedy každou periodu MEMS vyprodukuje 6 hodnot, které je možno číst. Dále lze nastavit citlivost senzoru pro sadu gyroskopů a akcelerometrů. Senzorická data jsou volitelně čtena v rozsahu 16bitového čísla s citlivostí $\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$ pro akcelerometry a $\pm 250°/s$, $\pm 500°/s$, $\pm 1000°/s$ a $\pm 200°/s$ pro gyroskopy.[10]

Dokumentace[10] se také zmiňuje o programovatelných filtrech šumu, ale bez jakýchkoliv detailů a tak nebudou v této práci využity.

Tento MEMS čip je umístěn na každém lokalizovaném modulu. Počítač s ním může komunikovat například I2C[8][10] protokolem. Pro použití v práci je tento protokol dostatečný a jeho princip není důležitý, mohl by být využit i jiný. Prakticky se jedná o zapojení 4 jump kabelů a použití knihovny WiringPi[11], která protokol implementuje a vystavuje API pro zápis a čtení 8bitových registrů čipu.

Zápisem do registrů čipu se MEMS ovládá. Čip obsahuje interní cyklickou frontu o velikosti 1024 bytů do které se vejde 85 kompletních šestic senzorických dat. Tyto data se čtou ze dvou speciálních registrů, které zároveň ovládají cyklickou frontu a garantují její korektnost.[12]

2.3.4 XBee

Použité XBee moduly[6] se dělí na 3 typy: koordinátor, router a koncové zařízení. Přitom z hlediska firmware se jedná o 2 typy: koordinátor a router, který může být přepnut na koncové zařízení a naopak.[13]

Každá topologie XBee sítě musí mít právě 1 koordinátor, který síť inicalizuje. Dále se mohou připojovat routery a nebo koncová zařízení. Každé nové zařízení připojené v síti má rodičovský uzel - router a nebo koordinátor. Koncové zařízení poté zprávy deleguje přes svůj rodičovský router (případně koordinátor).[13]

Použitý XBee protokol[14] umožňuje i konstrukci full-mesh topologií. Jednotlivé uzly mohou komunikovat skrze unicast, multicast a nebo broadcast. Přičemž jednotlivé zprávy jsou dle typu filtrovány hardwarem modulu - pokud není zpráva určena pro modul, ignoruje ji.[13]

Unicast se posílá maximálně 3x, dokud odesílatel neobdrží ACK[5] paket. Multicast a broadcast mají stejné podmínky - každou zprávu modul opakuje 3x. Broadcastů/multicastů z jednoho uzlu může být najednou maximálně 8.

[4] Mikro elektro-mechanický systém

[5] Acknowledgement

Obrázek 2.2: „Maják“ ve své pozici

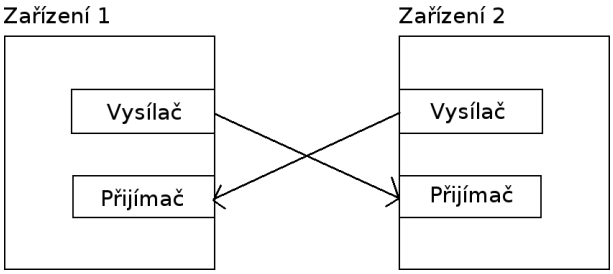
Ty se ukládají v seznamu, ze kterého po 8 sekundách zmizí - jeden modul tedy může průměrně vysílat až 1 broadcast za sekundu.[13]

Tyto čipy jsou umístěny jak stabilně v testovací místnosti, tak na každém lokalizovaném modulu. V místnosti jsou rozmístěny právě 4 čipy[15] připojené do jednoho ze serveru, v práci jsou označovány jako „majáky“.

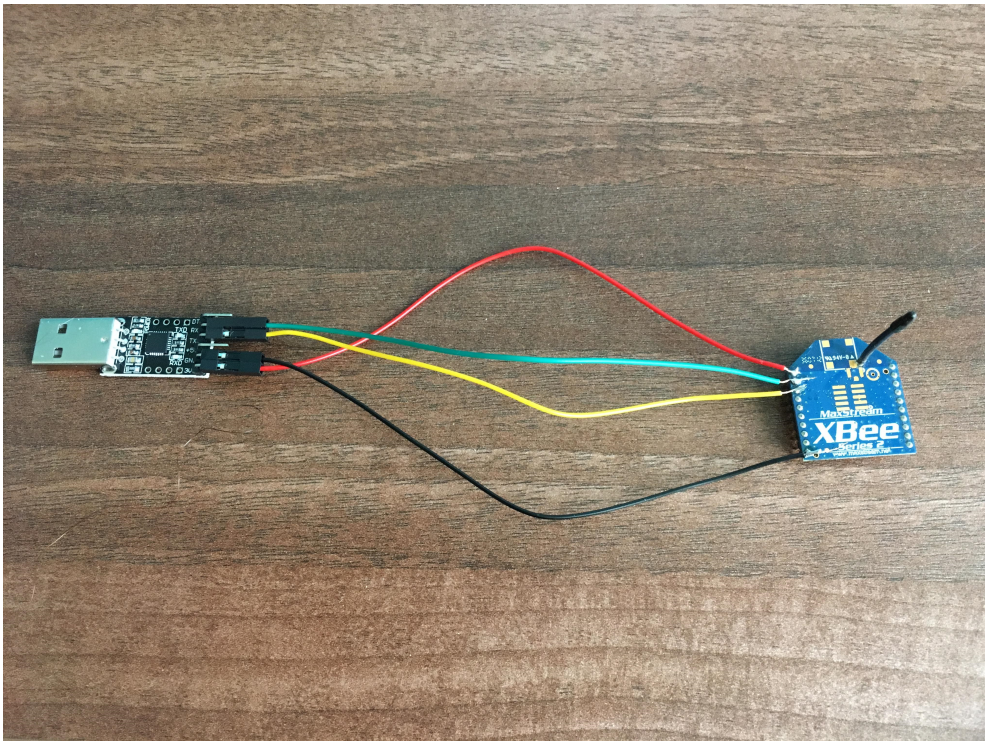
Počítač komunikuje s XBee modulem skrze UART[6][9] protokol[13]. Implementace protokolu je ponechána na ovladačích OS/třetích stran. Protokol umožňuje nezávisle data psát a číst na každé straně komunikace (viz schéma 2.3)[9]. Prakticky se jedná o zapojení 4 kabelů a zápis/čtení souboru skrze API operačního systému. V případě chybějících GPIO pinů na stranách serverů je využit USB[7] adaptér (obrázek 2.4).

Zjednodušeně řečeno použité čipy pracují ve dvou módech: příkazový mód a vysílací mód. Výchozí mód po zapojení modulu je vysílací mód. Pro vstup do příkazového módu je třeba definovanou dobu nevysílat, vyslat definovanou sekvenci dat a opět stejnou dobu nevysílat. Poté je čip přepnut do ovládacího módu. Z ovládacího módu se dá explicitně přejít do vysílacího módu a nebo po určité době vysílací nečinosti přejde do něho čip automaticky. V ovládacím módu se čip nastavuje nebo se dají číst konfigurační hodnoty. Jedná

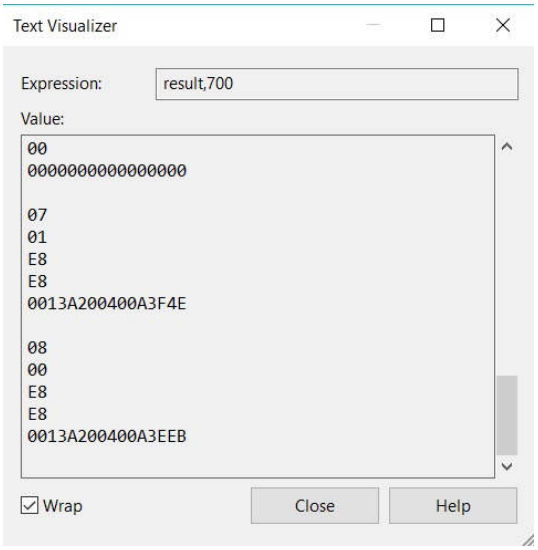
[6]Universal Asynchronous Receiver-Transmitter
[7]Universal Serial Bus



Obrázek 2.3: Schéma UART



Obrázek 2.4: Adaptér XBee modulu pro server



Obrázek 2.5: Útržek informací o posledních přijatých zprávách XBee čipu. V práci nevyužito pro nezdokumentovaný význam parametrů

se o předem definované ASCII zprávy vyslané po sériové lince. Výrobce čipů tvrdí, že každý čip má unikátní SID[8] číslo, které se dá použít pro směrování zpráv. V tomto módu se mj. nastavuje tato adresa příjemce, identifikátor PAN pro připojení do stejné sítě, použitý vysílací kanál, čekací doba vysílacího ticha pro vstup do příkazového módu, restartování modulu, přepojení do sítě...

Důležité nastavení pro použití v této práci jsou zmíněna v předchozí větě.[13]

API také umožňuje nastavit vysílací výkon, ale už neumožňuje číst velikost přiajtého výkonu. Sice se dají číst informace o přijatých paketech, ale bohužel jejich význam není zdokumentován[13].

Útržek na snímku 2.5 ukazuje frontu posledních přijatých zpráv. Tyto informace nejsou v práci využity.

Čili jediné relevantní měření pro lokalizaci lze provést mimo čip a to čtením a vysíláním zpráv. V této práci je použito měření času mezi zprávami.

[8]Serial Identification

Kapitola 3

Návrh

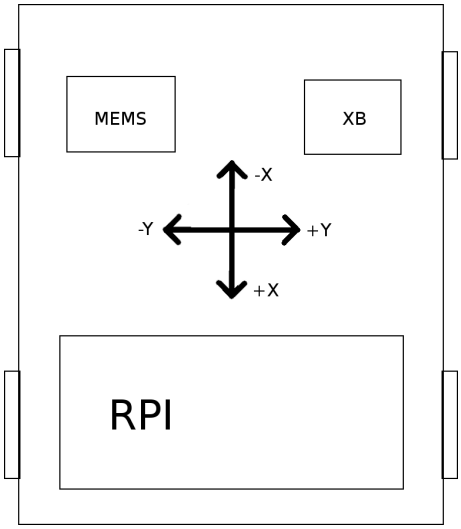
3.1 Testovací instance

Jednotilvé testovací instance se odehrávají v místnosti, která má zvolený počátek soustavy souřadnic v rohu, zvolenou orientaci ortogonálních os ve třech dimenzích a má předem naměřené a označené důležité body - počáteční nebo koncové pozice lokalizovaných modulů a pozice XBee majáků. Díky tomuto lze sestrojit počáteční model, ve kterém jsou všechny původní hodnoty známé.

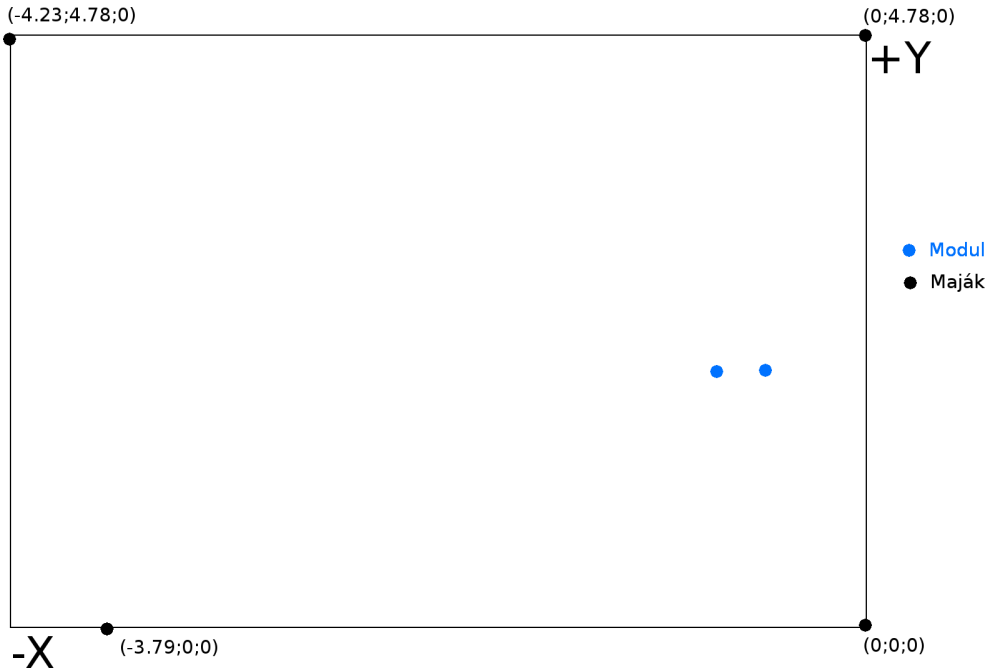
Trojice poloh v jednotlivých osách budou vždy v pořadí X, Y a Z. Osa Z míří v kladném směru dolů, jako směr gravitační síly.

V případě rotace je to taktéž - znamená to pravotočivá rotace podle os X, Y, Z v jejich kladném směru. Osy jsou definovány dle použitého senzoru[13], viz schéma 3.1. Osa Z míří v kladném směru dolů, jako směr gravitační síly.

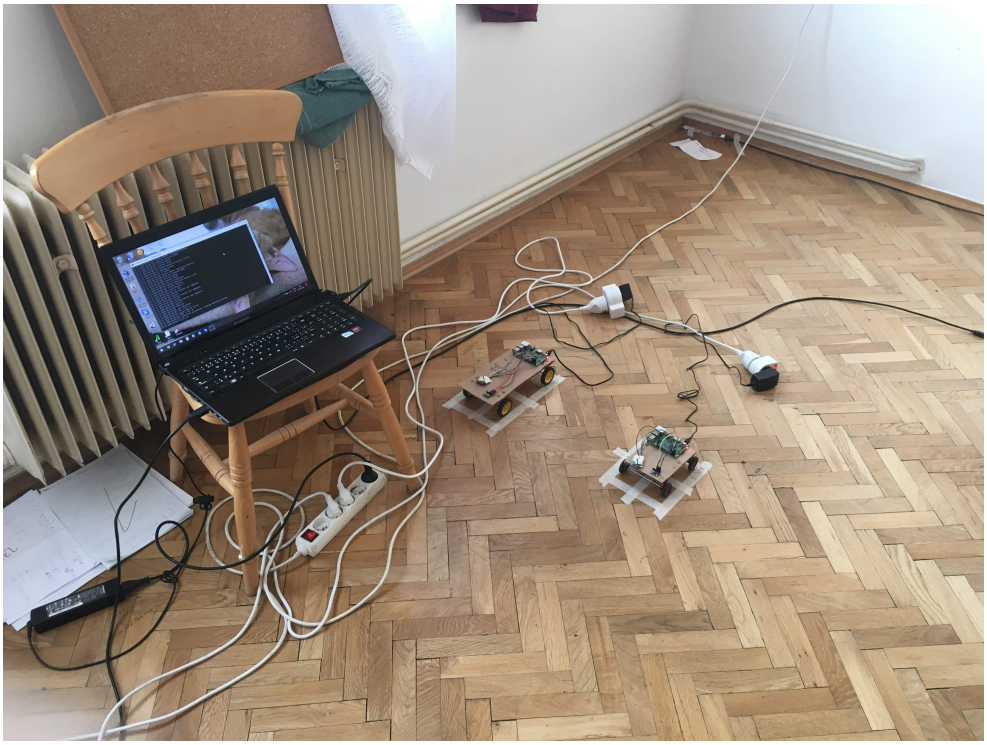
Testovací místnost je přibližně znázorněna ve schématu 3.2 a část skutečnosti pak vypadá jako na fotografii 3.3. Na fotografii je vidět 1 ze serverů, oba lokalizované moduly a 1 maják na pozici $(0; 0; 0)$.



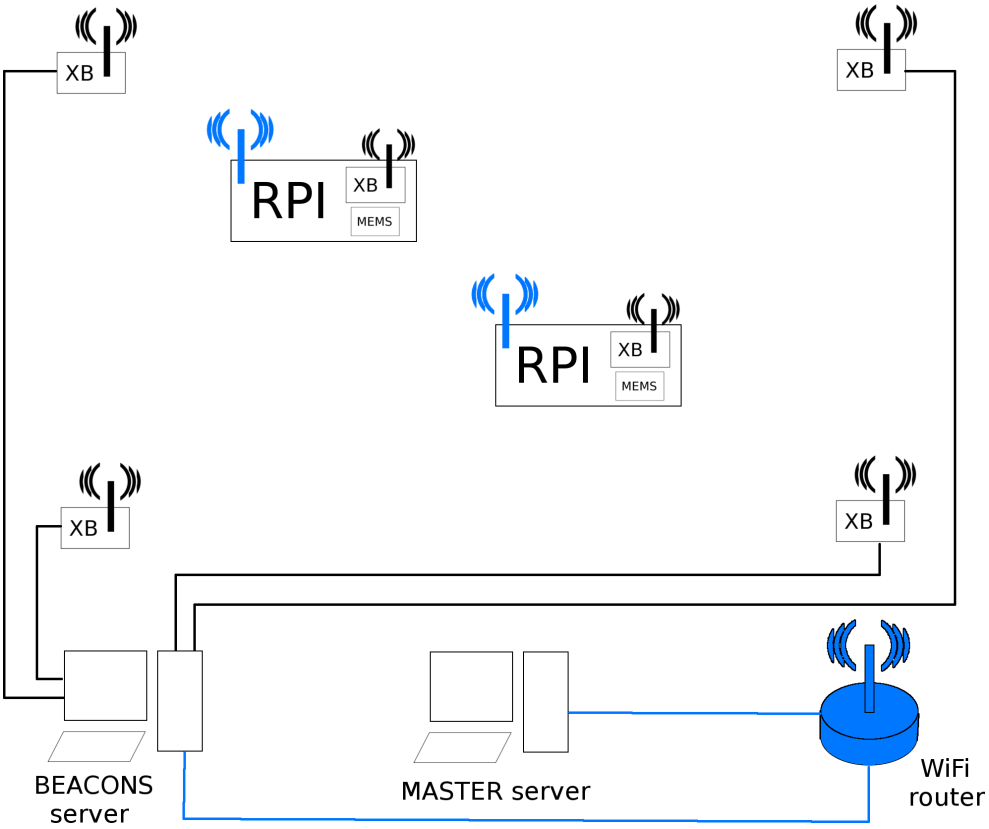
Obrázek 3.1: Schéma lokalizovaného modulu. Pohled shora. RPI = Raspberry Pi, MEMS = akcelerometr+gyroskop, XB = XBee bezdrátový modul



Obrázek 3.2: Schéma testovací místnosti. Pohled shora.



Obrázek 3.3: Testovací místnost ve skutečnosti



Obrázek 3.4: Síťová architektura. XB = bezdrátový XBee modul, RPI = Raspberry Pi, MEMS = akcelerometr+gyroskop

3.2 Vysokoúrovňová architektura systému

Celý systém obsahuje 2 sítě - WiFi LAN a XBee PAN, 2 typy serverů a 1 typ klienta.

Klient je bezdrátový modul, který systém lokalizuje. Je napojen do dvou sítí - přes WiFi na master server a skrze XBee na beacons server. Posílá sekvenci dat z MEMS senzorů přes WiFi a echo zprávu skrze XBee na beacons server.

Beacons server akumuluje tato XBee data a přeposílá je na master server.

Master server má kompletní datový snímek a lokalizuje klienty v prostoru.

3.3 MEMS Algoritmy

Tabulka 3.1: Tabulka algoritmů

Kód algoritmu	Popis	Varianty
MEMS.ORI	Sleduje orientaci modulu	citlivost senzoru x frekvence
MEMS.LOCO	Sleduje pohyb modulu (nezahrnuje rotaci)	citlivost senzoru x frekvence
MEMS.COMB	Sleduje orientaci a pohyb modulu zároveň	dle dílčích algoritmů
XB.PUMP	Modul vysílá periodicky signál skrze broadcast	frekvence broadcastu
XB.PING	Modul reaguje na požadavky serveru - ping	frekvence pingování
COMB.PUMP	Kombinace MEMS.COMB a XB.PUMP	dle dílčích algoritmů
COMB.PING	Kombinace MEMS.COMB a XB.PING	dle dílčích algoritmů

Obecně pro všechny algoritmy platí, že $O(t_0) = Or(t_0)$ a $P(t_0) = Pr(t_0)$, kde t_0 je počátek lokalizace a $Or(t_0)$, $Pr(t_0)$ jsou známé. Dále také platí, že MEMS senzory produkují posloupnost datových snímků jejíž členy jsou zaznamenány s konstatním časovým odstupem dt. Indexem i je označována šestice hodnot v pořadí se sekvence hodnot vyprodukovaných MEMS čipem.

Toto jsou důležité předpoklady, které musí být pro správnost algoritmů naplněny.

3.3.1 Algoritmus MEMS.ORI

Vstupem algoritmu je sekvence trojic úhlových rychlostí v jednotlivých ortogonálních osách modulu v jednotkách $°/s$.

Definice 3.3.1 (Úhlové rychlosti $Gs(t_i)$)**.** *Nazvěme $Gs(t_i)$ jako úhlové rychlosti v čase t_i od začátku testu. $Gs(t_i)$ je funkcí času jejímž obrazem jsou pravotočivé úhlové rychlosti (trojice) určené hodnotami dat ze senzorů sledovaného modulu v pořadí i ze sekvence hodnot. Tyto hodnoty jsou 16bitová čísla reprezentující hodnotu z rozsahu nadefinovaného[10] dle citlivosti senzoru.*

Věta 3.3.1 (Algoritmus MEMS.ORI)**.** *Označme $O(t_i) = (\sum_{j=1}^{j=i} Gs(t_j)) \cdot dt - i \cdot G$, kde G je gyroskopický šum určen jako střední hodnota náhodné veličiny určené kalibrační sekvencí.*

Kalibrační sekvence analyzuje šum jako náhodnou veličinu. U vstupních dat sleduje střední hodnotu. Analýza předpokládá platnost centrální limitní

věty[16] už pro zvolené velikosti kalibračních sekvencí v testech. Díky tomuto lze předpokládat, že v klidovém stavu sledovaného modulu by platilo $\lim_{i\to\infty} O(t_i) = G$. A tak lze použít bodový odhad pro střední hodnotu[17].

Důkaz algoritmu MEMS.ORI.
Protože se jedná o ortogonální soustavu souřadnic, lze vypočítávat jednotlivé souřadnice nezávisle.

Pro každou hodnotu $Gs(t_i)$ v sekvenci se provede lineární otočení a_i podle newtonovské mechaniky[18] jako:

$$a_i = (Gs(t_i) - G) \cdot dt$$

kde $Gs(t_i)$ je trojice úhlových rychlost naměřená v čase t_i, G je senzorický šum (střední hodnota kalibrační sekvence) a $dt = 1/f$, kde f je frekvence snímání nastavená na senzoru.

Jednotlivé postupé pohyby se pak dají sečíst následovně:

$$\begin{aligned}
O(t_i) &= a_1 + a_2 + ... + a_i \\
O(t_i) &= (Gs(t_1) - G) \cdot dt + (Gs(t_2) - G) \cdot dt + ... + (Gs(t_i) - G) \cdot dt && \text{(nahrazení } a\text{)} \\
O(t_i) &= ((Gs(t_1) - G) + (Gs(t_2) - G) + ... + (Gs(t_i) - G)) \cdot dt && \text{(vytknutí } dt\text{)} \\
O(t_i) &= (\sum_{j=1}^{j=i} (Gs(t_j) - G)) \cdot dt && \text{(převedení na sumu)} \\
O(t_i) &= (\sum_{j=1}^{j=i} Gs(t_j)) \cdot dt - i \cdot G && \text{(extrakce G)}
\end{aligned}$$

□

3.3.2 Algoritmus MEMS.LOCO

Algoritmus navíc pro svoji korektnost předpokládá, že modul se pohybuje buďto dopředu a nebo dozadu a to vždy v jedné ose - tedy při stejné orientaci.

Vstupem algoritmu je sekvence trojic zrychlení v jednotlivých ortogonálních osách modulu v jednotkách g.

Definice 3.3.2 (Akcelerace $A(t_i)$)**.** *Nazvěme $A(t_i)$ jako velikost akcelerace v čase t_i od začátku testu. $A(t_i)$ je funkcí času jejímž obrazem jsou hodnoty zrychlení (trojice) určené hodnotami dat ze senzorů sledovaného modulu v pořadí i ze sekvence hodnot. Tyto hodnoty jsou 16bitová čísla reprezentující hodnotu z rozsahu nadefinovaného[10] dle citlivosti senzoru.*

Definice 3.3.3 (Superpozice akcelerace $As(t_i)$)**.** *Nazvěme $As(t_i) = \sum_{j=1}^{j=i}(A(t_j) - A)$ jako superpozici jednotlivých snímků, kde A je akcelerační šum určen jako střední hodnota náhodné veličiny určené kalibrační sekvencí.*

Definice 3.3.4 (Rychlostní suma $Vs(t_i)$)**.** *Nazvěme* $Vs(t_i) = \sum_{j=1}^{j=i}(As(t_j))$ *jako rychlostní sumu.*

Věta 3.3.2 (Algoritmus MEMS.LOCO)**.** *Označme* $P(t_i) = P(t_0) + (As(t_i) + 2 \cdot Vs(t_{i-1})) \cdot 0.5 \cdot g \cdot dt^2$ *jako odhadovanou pozici modulu, kde* t_0 *je počátek testu a* g *je gravitační konstanta s hodnotou 9,8196.*

Kalibrační sekvence analyzuje akcelerační šum stejným způsobem jako gyroskopický šum.

Důkaz algoritmu MEMS.LOCO.
Protože se jedná o ortogonální soustavu souřadnic, lze vypočítávat jednotlivé souřadnice nezávisle.

Pro každou hodnotu $A(t_i)$ v sekvenci se provede lineární pohyb m_i podle newtonovské mechaniky[18] jako:

$$m_i = v_{i-1} \cdot dt + 0.5 \cdot g \cdot (A(t_i) - A) \cdot dt^2$$

kde v_{i-1} je rychlost spočtená dle newtonowské mechaniky[18] jako:

$$v_{i-1} = v_{i-2} + (A(t_{i-1}) - A) \cdot g \cdot dt \qquad (\text{pro } i > 1,\ v_0 = 0, v_{-1} = 0)$$

$A(t_i)$ je trojice úhlových rychlostí naměřená v čase t_i, A je senzorický šum (střední hodnota kalibrační sekvence) a $dt = 1/f$, kde f je frekvence snímání nastavená na senzoru. g je gravitační konstanta s hodnotou 9,8169. Je použita, protože $A(t_i)$ je v jednotkách g.

Jednotlivé rychlosti se dají spočítat následovně:

$$\begin{aligned}
v_0 &= 0 \\
v_1 &= 0 + (A(t_1) - A) \cdot g \cdot dt \\
v_2 &= 0 + (A(t_1) - A) \cdot g \cdot dt + (A(t_2) - A) \cdot g \cdot dt \\
&\vdots \\
v_i &= (\sum_{j=1}^{j=i} A(t_j) - A) \cdot g \cdot dt \\
&= As(t_i) \cdot g \cdot dt
\end{aligned}$$

Jednotlivé postupé pohyby se pak dají sečíst následovně:

$$
\begin{aligned}
&P(t_i) = P(t_0) + m_1 + m_2 + ... + m_i \\
&P(t_i) = P(t_0) + 0.5 \cdot g \cdot (A(t_1) - A) \cdot dt^2 + \\
&\quad + As(t_1) \cdot g \cdot dt \cdot dt + 0.5 \cdot g \cdot (A(t_2) - A) \cdot dt^2 + ... \\
&\quad ... + As(t_{i-1}) \cdot g \cdot dt \cdot dt + 0.5 \cdot g \cdot (A(t_i) - A) \cdot dt^2 && \text{(nahrazení } m\text{)} \\
&P(t_i) = P(t_0) + (0.5 \cdot (A(t_1) - A) + \\
&\quad + As(t_1) + 0.5 \cdot (A(t_2) - A) + ... \\
&\quad ... + As(t_{i-1}) + 0.5 \cdot (A(t_i) - A)) \cdot g \cdot dt^2 && \text{(vytknutí } g \cdot dt^2\text{)} \\
&P(t_i) = P(t_0) + ((A(t_1) - A) + \\
&\quad + 2 \cdot As(t_1) + (A(t_2) - A) + ... \\
&\quad ... + 2 \cdot As(t_{i-1}) + (A(t_i) - A)) \cdot 0.5 \cdot g \cdot dt^2 && \text{(vytknutí 0.5)} \\
&P(t_i) = P(t_0) + (\sum_{j=1}^{j=i} (A(t_j) - A) + 2 \cdot \sum_{j=1}^{j=i-1} As(t_j)) \cdot 0.5 \cdot g \cdot dt^2 && \text{(převedení na sumy)} \\
&P(t_i) = P(t_0) + (As(t_i) + 2 \cdot Vs(t_{i-1})) \cdot 0.5 \cdot g \cdot dt^2 && \text{(nahrazení z definice)}
\end{aligned}
$$

□

3.3.3 Algoritmus MEMS.COMB

Vstupem algoritmu je sekvence šestic (úhlové zrychlení a velikost akcelerace ve třech ortogonálních osách) dat ze senzorů

Oproti oběma předcházejícím algoritmům je tato verze iterativní. Díky malým dt (řádově mikro až milisekundy) považuje algoritmus jednotlivé vstupní šestice a předchozí $P()$, $O()$ jako zrdroj pro jediný lineární pohyb v daném rámci.

Pohybová rovnice pak využívá newtonovské mechaniky[18]. V každém snímku se přepočítává pozice podle následujících formulí:

Definice 3.3.5 (Aktuální akcelerace v místnosti $At(t_i)$)**.** *Nazvěme $At(t_i) = R \cdot A(t_i) - R \cdot A$ jako aktuální akceleraci modulu v soustavě souřadnic místnosti, očištěnou od šumu. R je rotační matice stvořená z hodnot $O(t_i)$ a A je akcelerační šum vypočtený z kalibrační sekvence. $At(t_i)$ je trojice násobků konstanty g. Jedná se tedy o lineární zobrazení vektoru $A(t_i)$ z báze senzoru do báze místnosti.*

Věta 3.3.3 (Aktuální rychlost v místnosti $Vt(t_i)$)**.** *Označme $Vt(t_i) = V(t_{i-1}) + At(t_i) \cdot dt$ jako aktuální rychlost pohybu modulu v soustavě souřadnic místnosti. [18]*

Věta 3.3.4 (Algoritmus MEMS.COMB)**.** *Označme* $P(t_i) = P(t_{i-1}) + Vt(t_{i-1}) \cdot dt + 0.5 \cdot g \cdot dt^2 \cdot At(t_i))$ *jako odhadovanou pozici modulu, kde g je gravitační konstanta s hodnotou 9,8196.*

Důkazy nejsou prováděny, protože se jedná standardní rovnice klasické mechaniky z [18].

3.4 XB algoritmy

Algoritmy naivně předpokládají, že doba zpracování dat mezi jednotlivými stanicemi je na jednotlivých stanicích konstantní a že zpráva, kterou modul vyslal byla přijata napoprvé a cestovala nejkratší možnou cestou.

Dále předpokládají, že zpráva putuje prostorem rychlostí světla.

Poté se dá vypočítat vzdálenost jednotlivých XB modulů newtonovskou mechanikou[18], jako $r = c \cdot t$, kde r je vzdálenost XBee modulů, t je doba přenosu zprávy a c je rychlost světla.

Protože algoritmy lokalizují ve 3D prostoru, je zapotřebí umístit do prostoru alespoň 4 majáky [15] (k nalezení v příloze).

Tyto algoritmy neurčují $O(t)$, určují pouze $P(t)$. Algoritmy se navzájem liší výpočtem vzdálenosti majáku a lokalizovaného modulu.

Po tomto výpočtu oba algoritmy vypočítají průnik AABB[9] sfér jednotlivých majáků (určenou vzdáleností lokalizovaného modulu r od pozice majáku).

Výsledkem je kvádr ve kterém se lokalizovaný modul nachází - $P(t)$ se pak vypočte jako střed diagonály výsledného kvádru.

Pokud se modul zároveň nachází v dosahu každého majáku, tak se určitě nachází v průniku všech těchto sfér určených pozicí majáku a vzdálenosti lokalizovaného modulu r od daného majáku.

Pro jednoduchost výpočtu je použit AABB namísto sféry, který pro instance v této práci je dostatečný. AABB je krychle obalující tuto sféru, stěny krychle jsou paralelní s soustavou souřadnic místnosti. Výsledný kvádr pak obsahuje lokalizovaný modul. Střed diagonály je pak střed hran ve všech rozměrech, což je zároveň nejpravděpodobnější pozice výskytu.

3.4.1 Algoritmus XB.PUMP

Vstupem algoritmu jsou dvojice identifikátoru majáku a časový interval. Tento algoritmus je stejně jako MEMS.COMB iterativní.

Lokalizovaný modul s učitou periodou T vysílá broadcastem zprávu. Jednotlivé XB majáky měří dobu mezi jednotlivými příjmy.

Algoritmus poté upraví vzdálenosti podle následující formule:

[9] Axis Aligned Bounding Box

Věta 3.4.1 (Vzdálenost v algoritmu XB.PUMP)**.** *Označme* $r(i) = r(i-1) - (t_i - T) \cdot c$ *jako vzdálenost lokalizovaného modulu od majáku.* t_i *je naměřený časový interval mezi přijatými broadcast zprávami na majáku v pořadí* $i-1$ *a* i*.*

Důkaz vzdálenost v algoritmu XB.PUMP.
Data putují jedním směrem a podle předpokladu na stanicích stráví konstatní čas T_1 a T_2.

Jednotlivé zprávy budou tedy přijaty v časech t_{r_i}, t_{t_i} je doba strávená bezdrátovým přenosem.

$$\begin{aligned} t_{r_1} &= T_1 + t_{t_1} + T_2 \\ t_{r_2} &= T + T_1 + t_{t_2} + T_2 \\ t_{r_3} &= 2T + T_1 + t_{t_3} + T_2 \\ &\vdots \\ t_{r_i} &= (i-1) \cdot T + T_1 + t_{t_i} + T_2 \end{aligned}$$

Rozdíly mezi přijatými časy jsou tedy:

$$\begin{aligned} t_i &= t_{r_i} - t_{r_{i-1}} \\ t_i &= (i-1) \cdot T + T_1 + t_{t_i} + T_2 - ((i-2) \cdot T + T_1 + t_{t_{i-1}} + T_2) \\ t_i &= T + t_{t_i} - t_{t_{i-1}} \end{aligned}$$

Pokud se lokalizovaný modul nehýbe, je naměřený čas $t_i = T$, protože $t_{t_i} = t_{t_{i-1}}$.

Pokud se hýbe, jediný čas, který se změnil je čas bezdrátového přenosu a ten je přímo úměrný změně vzdálenosti, protože rychlost přenosu je pro obě zprávy stejná.

$$\begin{aligned} r(i) &= r(i-1) - (t_i - T) \cdot c \\ r(i) &= r(i-1) - (T + t_{t_i} - t_{t_{i-1}} - T) \cdot c \\ r(i) &= r(i-1) - (t_{t_i} - t_{t_{i-1}}) \cdot c \end{aligned}$$

Zde už se opět pracuje s klasickou mechanikou[18]. Pokud se modul přiblíží, bude $t_{t_{i-1}} < t_{t_i}$ a tím pádem i $r(i) < r(i-1)$ a analogicky naopak. □

3.4.2 Algoritmus XB.PING

Vstupem algoritmu jsou stejné veličiny, jako u XB.PUMP algoritmu. Tento algoritmus není iterativní a ani nepotřebuje znát $P(t_0)$.

Algoritmus je inspirován pingem[19]. Maják začne měřit čas, vyšle unicast žádost a očekává odpověď od lokalizovaného modulu, poté zastaví časomíru. Předpokladem je, že během času zpracovávání zprávy na straně lokalizovaného modulu se modul nehýbe.

Věta 3.4.2 (Vzdálenost v algoritmu XB.PING)**.** *Označme* $r(i) = (t_{p_i} \cdot 0.5 - 2 \cdot (T_1 + T_2))) \cdot c$ *jako vzdálenost lokalizovaného modulu od majáku.* t_{p_i} *je naměřený časový interval,* $2 \cdot (T_1 + T_2)$ *je doba strávená na jednotlivých zařízeních a ne bezdrátovým přenosem. Doba strávená na jednotlivých zařízeních je vypočtena z kalibrační sekvence.*

Při kalibraci jsou vzdálenosti známy díky určeným předpokladům a tak u jednotlivých kalibračních t_{p_i} lze dopočítat dobu strávenou na zařízeních jako $2 \cdot (T_1 + T_2) = t_{p_{max}} - 2 \cdot r(0)$, kde $r(0)$ je známá počáteční vzdálenost a $t_{p_{max}}$ je nejdelší naměřená doba z kalibrační sekvence.

Opět se jedná o klasickou mechaniku[18].

3.4.3 Algoritmy COMB.PUMP a COMB.PING

Algoritmy by byly navrženy v případě dobrých výsledků testů pro jednotlivé dílčí algoritmy.

3.5 Testy

Jednotlivé testy jsou prováděny postupně a jsou na sobě závislé. Nedává smysl test provádět, pokud ten předchozí nedopadl dobře.

Tabulka 3.2: Tabulka testů

Kód testu	Popis	Účel
FREQUENCY	Analyzuje datový výstup ze senzorů	určit nastavení frekvence snímání
SENS	Analyzuje datový výstup ze senzorů	určit nstavení citlivosti senzorů
WARMUP	Analyzuje smysluplnost datového výstupu od určitého snímku	určit zahřívací snímek
STABLE	Sleduje lokalizovaný modul v klidu	analyzovat šum
BASIC	Sleduje triviální změny pohybu a nebo orientace	ověřit úspěšnost algoritmu, sledovat chybovost
ADVANCED	Sleduje složitější kombinace pohybu a orientace	ověřit úspěšnost algoritmu, sledovat chybovost
FINAL	Sleduje simulovaný běžný provoz	analyzovat akumulaci chyb/určit dobu překalibrování

Jednotlivé posutpy a výsledky jsou zmíněny v kapitole Testování.

Kapitola 4

Implementace

V této kapitole je vysokoúrovňově popsáno, jak systém funguje. Detailnější informace viz zdrojový kód na přiloženém médiu.

Systém je vyvynut v jazyce C++ za použití standardní knihovny, systémového API a knihovny WiringPi[11]. Bezpečnost nebyla pro jednoduchost brána v potaz, to stejné platí pro optimalizaci kódu pro rychlost/paměť. Dále systém předpokládá, že uživatel ho používá správným způsobem a připojuje klienty/servery do sítě ve správném pořadí.

Systém vyžaduje právě 4 XBee majáky zapojené do jednoho Beacons serveru a podporuje více lokalizovaných modulů. Testován v průběhu implementace je při připojení jednoho nebo dvou lokalizovaných modulů.

Celá myšlenka systému je testovací instanci nainicializovat, nakalibrovat, libovolně dlouho lokalizovat moduly a případně nahrávat záznam, který se dá při dalším spuštění přehrát.

V případě přehrání záznamu lze spustit jen Master server, který záznam přehrává dokola. Ukázka grafického rohraní je možná vidět ze snímku 4.1. Jednotlivé aplikace podporují konfigurační soubory ve kterých se dají nastavit parametry místnosti, lokalizovaných modulů a připojených senzorů.

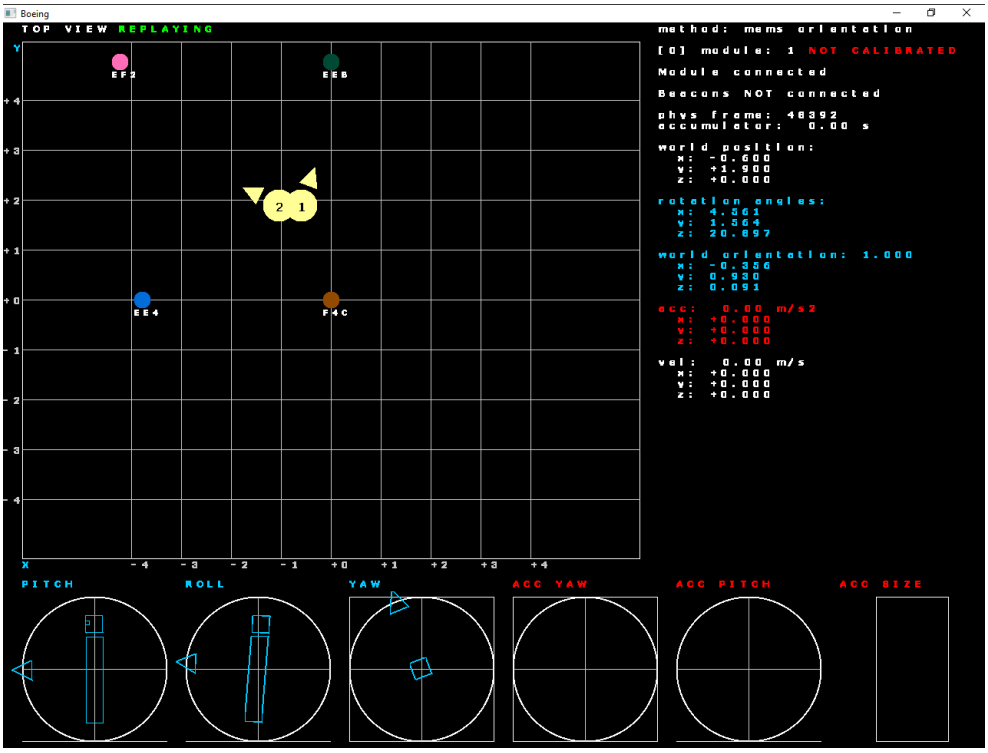
Jednotlivé síťové uzly mezi sebou komunikují aplikačními zprávami skrze TCP[20] nebo jednobytovými zprávami skrze XBee protokol[14].

Aplikace jsou vícevláknové a přistupují do společné paměti - jediné kritické sekce jsou přístupy do cyklických front, které jsou vyřešeny synchronizací metodou FAA[10] a interní přeinicializační funkce, které jsou synchronizovány aktivním čekáním a principu 1 písař, 1 čtenář.

Konfigurační soubory jsou textové soubory umožňující komentáře. Obsahují nastavení senzorů, informace pro TCP připojení a výchozí pozice nebo oritentace uzlů.

Záznamy obsahují nutné informace pro totožnou rekonstrukci živé lokalizace. Ukládána jsou data ze senzorů a nikoliv kalkulované hodnoty - aby

[10] Fetch and Add



Obrázek 4.1: Snímek grafického rozhraní při přehrávání záznamu. Mřížka reprezentuje testovací místnost a polohy/rotace modulů v reálném čase při pohledu shora, infografika dole ukazuje aktuální akcelerace/rotace z pohledu jednoho modulu, výpis informací vpravo obsahuje technické a vypočítané parametry sledovaného modulu.

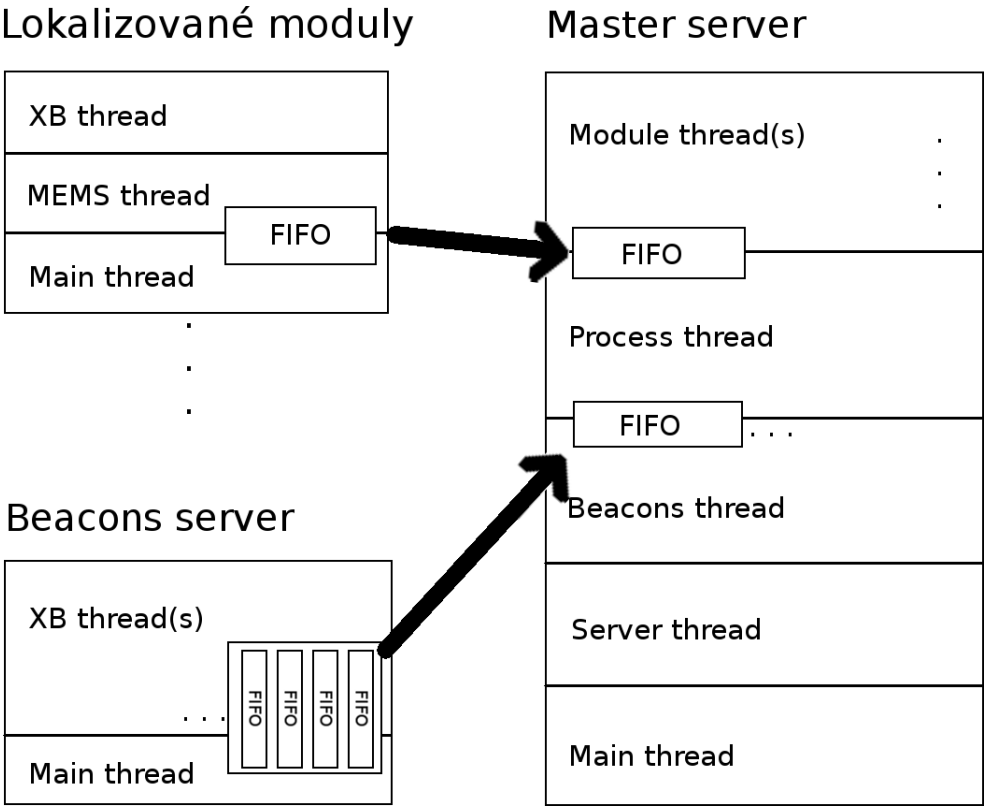
bylo možné senzorická data analyzovat a nebo jednodušše porovnávat různé algoritmy.

Pro měření času na Windows platformách je využito systémových volání QueryPerformaceCounter[21] a QueryPerformanceFrequency[22] a pro linuxové platformy je použito clock_gettime[23].

Pro vstup do řídícího módu u XB čipů byl zvolen minimální čekací interval, který bylo schopné konzistentně udržovat - a to v případě konrétního hardwaru 100ms. V případě XB.PUMP algoritmu je broadcast hops nastaven na 1, čili zpráva by podle dokumentace[13] měla být doručena všem uzlům přímo od zdroje - tím pádem také zůstává průměrný limit broadcast za sekundu bez ohledu na počet lokalizovaných modulů, protože zpráva se vysílá pouze ze zdrojového úzlu.

Náhled toku dat a jednotlivá vlákna je možné vidět ve schématu 4.2.

Jednotlivé rámy znázorňují aplikaci. Každý rám je rozdělen na jednotlivá



Obrázek 4.2: Schema vláken a toku dat v systému

vlákna a mezi vlákny jsou zvýraznené sdílené fronty (FIFO[11]).

Za každý lokalizovaný modul přítomný v systému je jedna aplikace „Lokalizované moduly“, čtveřice front v Beacons serveru, module vlákno s FIFO frontou a FIFO fronta mezi Becons thread a Process thread v Master serveru.

Protože data ze senzorů jsou sekvencí se známým konstatním časovým odstupem, je možné takto se zpožděním přeposílat skrze TCP[20] a celou sekvenci interpretovat se zanedbatelným zpožděním.

V případě XB.PING algoritmu dochází ke zbytečnému zpoždění, ale toto zpoždění nemá vliv na korektnost výpočtu, pouze na odezvu změny polohy, která je v práci pro jednoduchost zanedbána.

Pro reprezentaci jsou použity 64 bitové verze datových typů float nebo int.

Výhodou MEMS.ORI a MEMS.LOCO algoritmů je, že implementace ukládá senzorická data v podobě 64 bitového int a nezavádí akumulaci chyb použitím float datového typu[24]. Odchylka v nutných float operacích se nepřenáší z rámce na rámec. Tato vlastnost bohužel neplatí pro MEMS.COMB algoritmus.

[11] First in First out

U XB.PING algoritmu se chyba neakumuluje, protože výpočet pozice je absolutní a nevyžaduje znalost předchozího stavu. Pro XB.PUMP se odchylky akumulují.

Porovnání odchylek je k nahlédnutí v kapitole Testování.

Jednotlivé testy jsou prováděny nad záznamy v separovaných malých ad hoc aplikacích.

4.1 Master server

Master server musí být spuštěn jako první. Očekává TCP[20] spojení od Beacons serveru a následně potvrzení o úspešné inicializaci XBee PAN. Poté očekává TCP[20] spojení od jednotlivých lokalizovaných modulů a následnou zprávu o úspešné inicializaci MEMS a připojení do XBee PAN.

Nyní systém přijímá data od Beacons serveru a od lokalizovaných modulů skrze TCP[20] spojení a jednotlivé moduly lokalizuje. Každý lokalizovaný modul obsahuje dvě cyklické fronty - pro XBee data a pro MEMS data, které jsou postupně využívány algoritmy.

Server využívá principu fixed time delta[25] pro implementaci reálného času.

Module thread ukládá přijatá MEMS data po síťi do lokální FIFO fronty.

Beacons thread ukládá přijadá XBee data po síťi do lokální FIFO fronty.

Process thread vypočítává lokalizační algoritmy na základě dat ze sdílených FIFO front.

Server thread obsluhuje nové TCP připojení.

Main thread vykresluje aktuální stav.

4.2 Beacons server

Beacons server očekává spuštěný Master server. Při spuštění restartuje XBee moduly do továrního nastavení a nakonfiguruje XB síť. Poté se připojí na Master server, oznámí úspěch a dle typu XB algoritmu očekává příkazy.

V případě XB.PING algoritmu to jsou: začni PING pro modul/skonči PING pro modul.

V případě XB.PUMP neočekává od Master serveru žádné řídící zprávy.

Pro XB.PING algoritmus po přijetí příkazu začni PING cyklicky vysílá z majáku XBee unicast pro lokalizovaný modul a očekává odpověď. Tuto prodlevu měří.

V případě XB.PUMP algoritmu pouze čte přijatou zprávu a označí ji časovou známkou.

Oba typy měření jsou ukádány do FIFO.

Jakmile fronty obsahují nějaká data, tak jsou odeslána skrze TCP[20] spojení na Master server.

Beacons server má počet vláken dle zvoleného algoritmu.

Main vlákno přeposílá FIFO data na server.

XBee vlákna ukládají časové údaje do lokálních front. Počet front je u obou algoritmů stejný.

Pro XB.PING algoritmus je počet XBee vláken 1. Toto vlákno dokola postupně pinguje z jednotlivých majáků střídavě lokalizované moduly.

V případě XB.PUMP algoritmu jsou XBee vlákna 4 - každé za jeden maják. Vlákno dokola čeká na přijatou zprávu a zařazuje naměřený čas do určené fronty.

4.3 Lokalizovaný klient

Klient vyžaduje spuštěný Master server. Po úspěšném TCP[20] spojení přijímá od serveru XBee PAN specifikaci, resetuje MEMS a XBee do továrních nastavení a nakonfiguruje je dle potřeby, poté oznámí úspěch Master serveru.

Poté sbírá MEMS data do lokální fronty, kterou při určité minimální velikosti skrze TCP[20] přeposílá na Master server.

V případě XB.PUMP algoritmu periodicky vysílá XB broadcast.

V případě XB.PING algoritmu čeká na jednotlivé požadavky a na ně odpovídá.

Klient má 3 vlákna.

XBee vlákno obsluhuje dle algoritmu bezdrátovou XBee komunikaci.

MEMS vlákno čte data z MEMS senzoru a uklídá je do lokální fronty.

Main vlákno tyto FIFO data přeposílá na Master server.

Kapitola 5

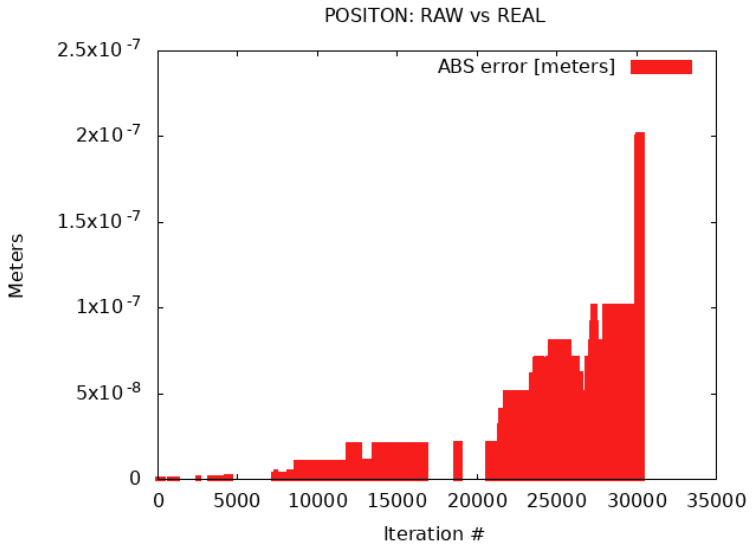
Testování

Pro testování byly lokalizované moduly označeny jako Henry a Rollo.

5.1 Test akumulace chyby typů float vs int

Test sleduje akumulaci chyby při výpočtu pozice v jednoosovém pohybu metodou MEMS.LOCO a MEMS.COMB. Pohybová senzorová data byla náhodně generována LCG[12][26]. Výpočet v každé iteraci probíhal dle definic algoritmů.

[12]Lineární Kongruenční Generátor



Obrázek 5.1: Graf absolutní chyby MEMS.LOCO vs MEMS.COMB

Z grafu 5.1 je vidět, že chyba se definitivně akumuluje a po 30 000 iteracích se pohybuje v řádech stovek nanometrů, což je zanedbatelná chyba v případě krátce trvajících lokalizacích.

Tedy z tohoto hlediska lze bezproblému v takovéto implementaci MEMS.COMB používat ale po určité době, kdy naakumulovaná chyba přesáhne chtěnou mez by se měla pozice modulu překalibrovat.

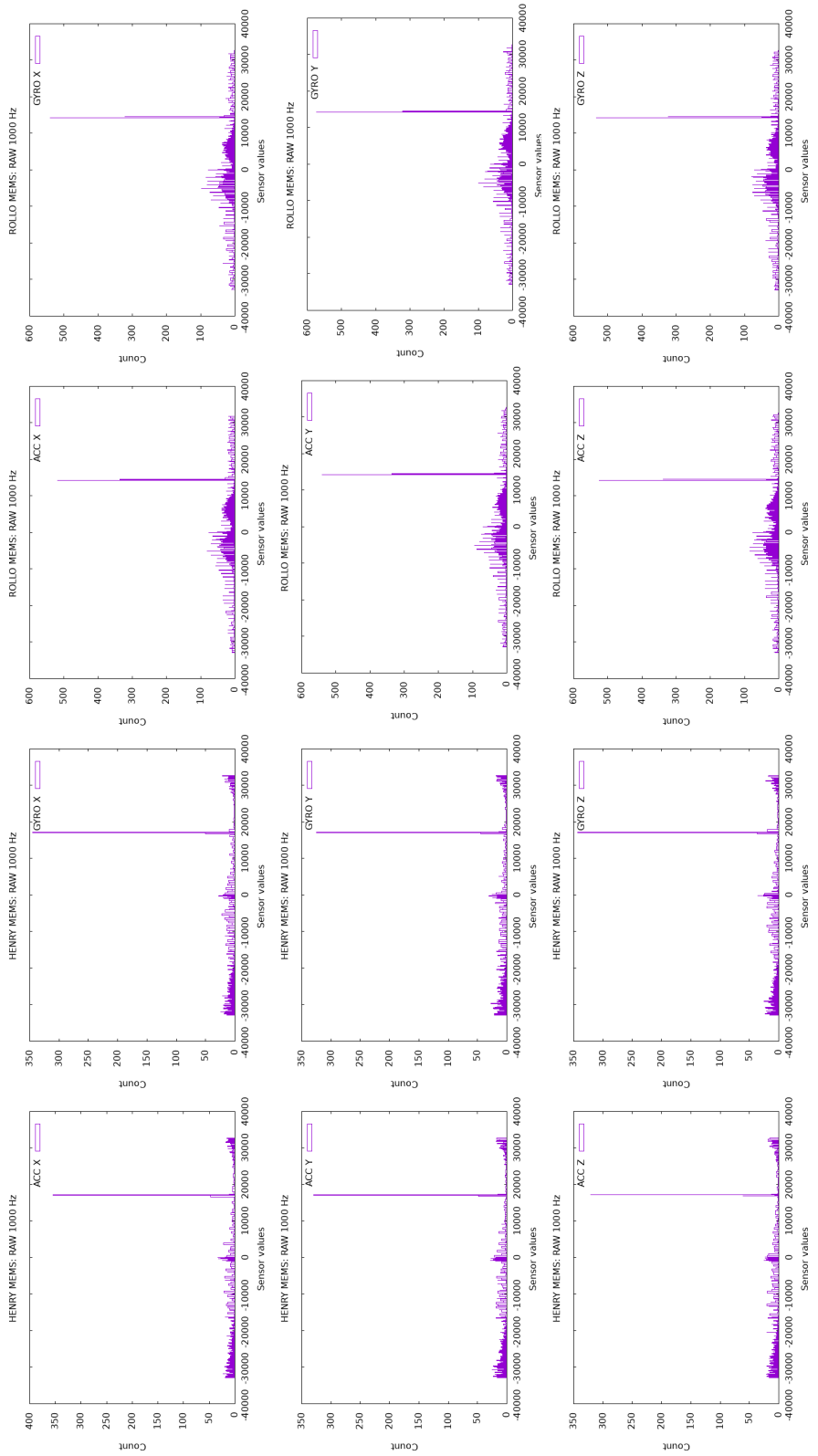
5.2 Test FREQUENCY

Tento test má za cíl nalézt frekvenci snímkování jednodlivých čipů. Větší frekvence je lepší. Pro XB.PING algoritmus nemá smysl frekvenci měřit, poněvadž moduly reagují na zprávy podle toho, jak dorazí. Pro XB.PUMP algoritmus je frekvence díky zvolenému hardwaru určena limitací maximálně 1 broadcast za sekundu, přesto se však ukázalo, že tato frekvence je na testovaném hardwaru příliš vysoká - nejmenší přijatelná frekvence byla 0.5 Hz. Pro všechny MEMS senzory je frekvence konfigurovatelná v rozmezí 1000 - 1 Hz.

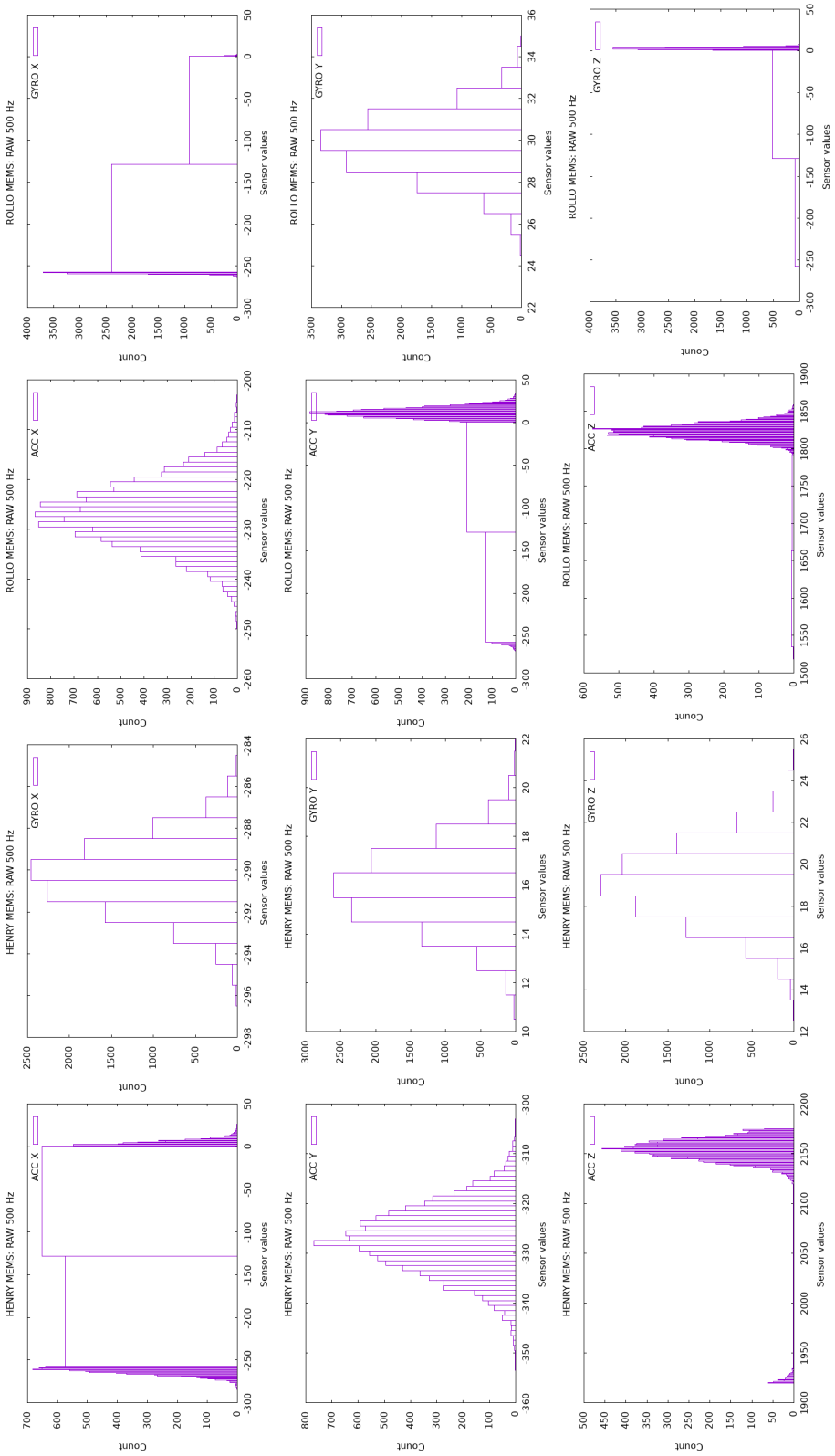
V testu stály moduly v klidu a byl pořizován záznam od prvních senzorických dat.

Z grafů je vidět, že pro frekvenci 1000 Hz jsou data výrazně jiná od nižších frekvencí, navíc v klidovém stavu jsou očekávány hodnoty v různých osách okolo určitých hodnot dle nastavení citlivosti - reprezentujících gravitační sílu. Tomu tak není. Problém taky je, že RPI[13] aplikace nestíhala odebírat data z MEMS fronty. Ostatní frekvence poté mají podobný charakter. Pro následující testy byla zvolena frekvence 500 Hz.

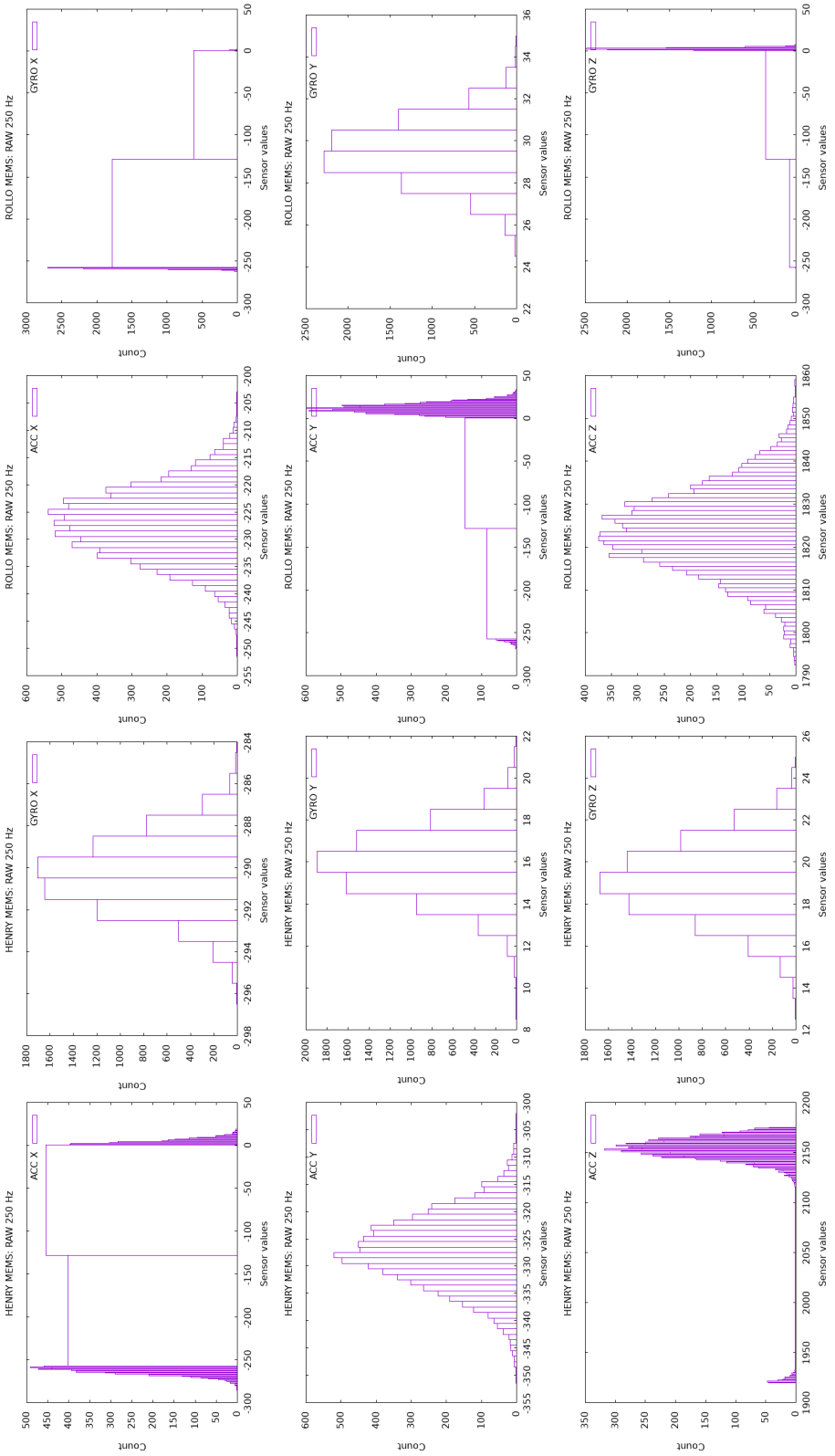
[13]Raspberry Pi



Obrázek 5.2: Histogramy hodnot jednotlivých senzorů testu FREQUENCY pro 1000 Hz



Obrázek 5.3: Histogramy hodnot jednotlivých senzorů testu FREQUENCY pro 500 Hz

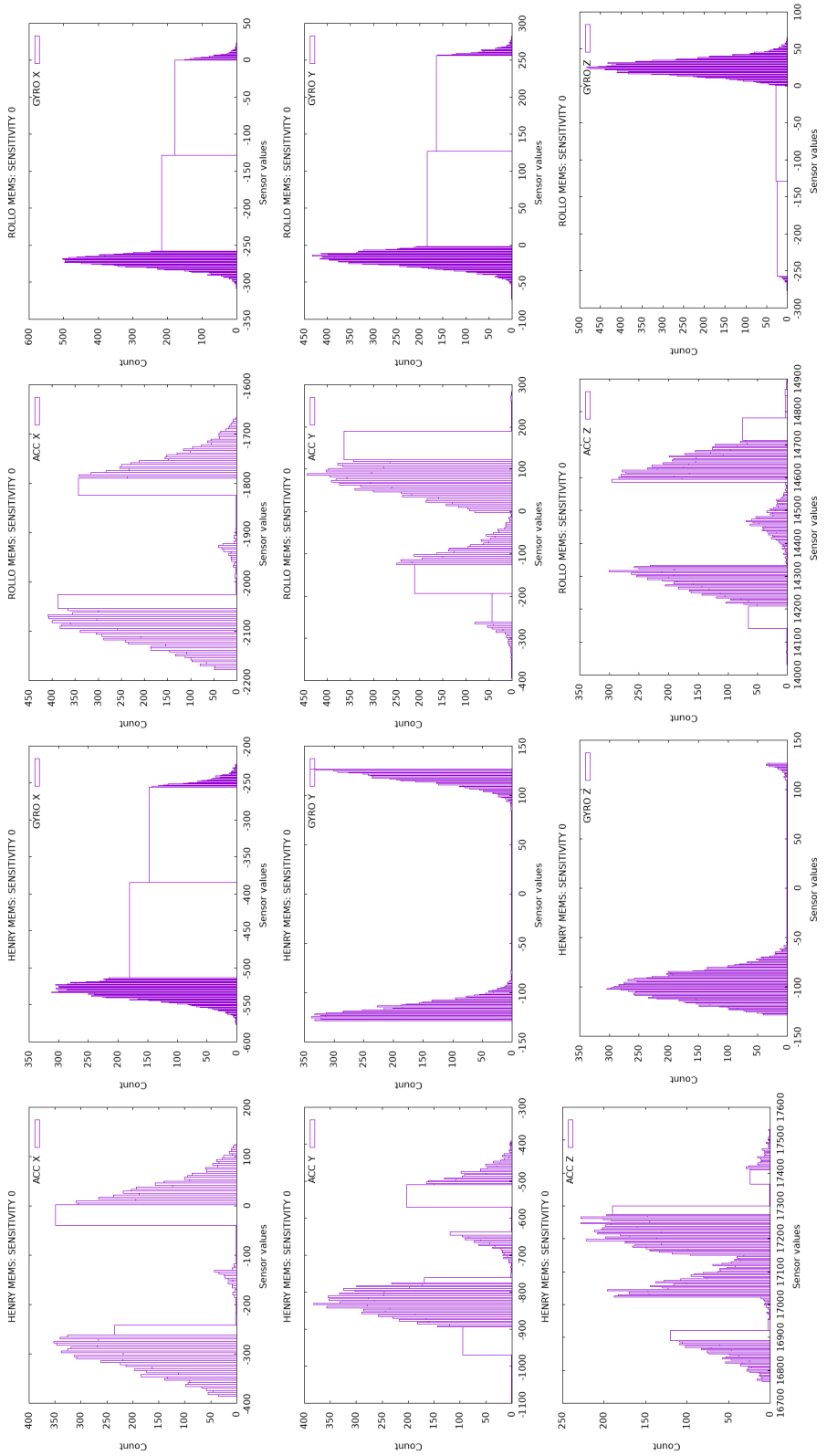


Obrázek 5.4: Histogramy hodnot jednotlivých senzorů testu FREQUENCY pro 250 Hz

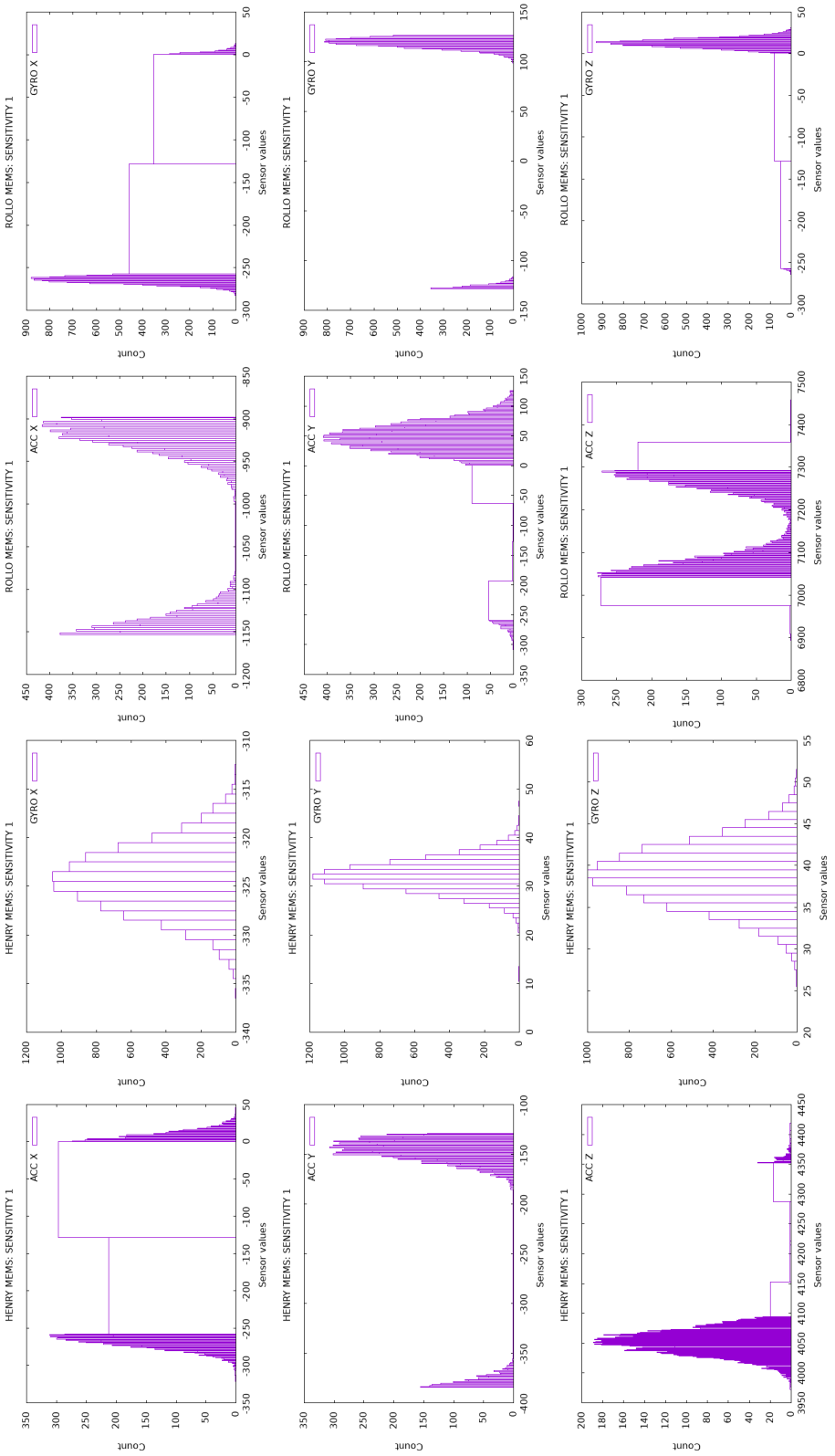
5.3 Test SENS

Test hledá vhodné nastavení citlivosti senzorů. Větší rozsah je méně detailnější a opomíjí některé menší pohyby. Citlivosti jsou pouze pro MEMS algoritmy. Celkem lze nezávisle nastavit 4 různé stupně citlivosti jak pro akcelerometry, tak pro gyroskopy.

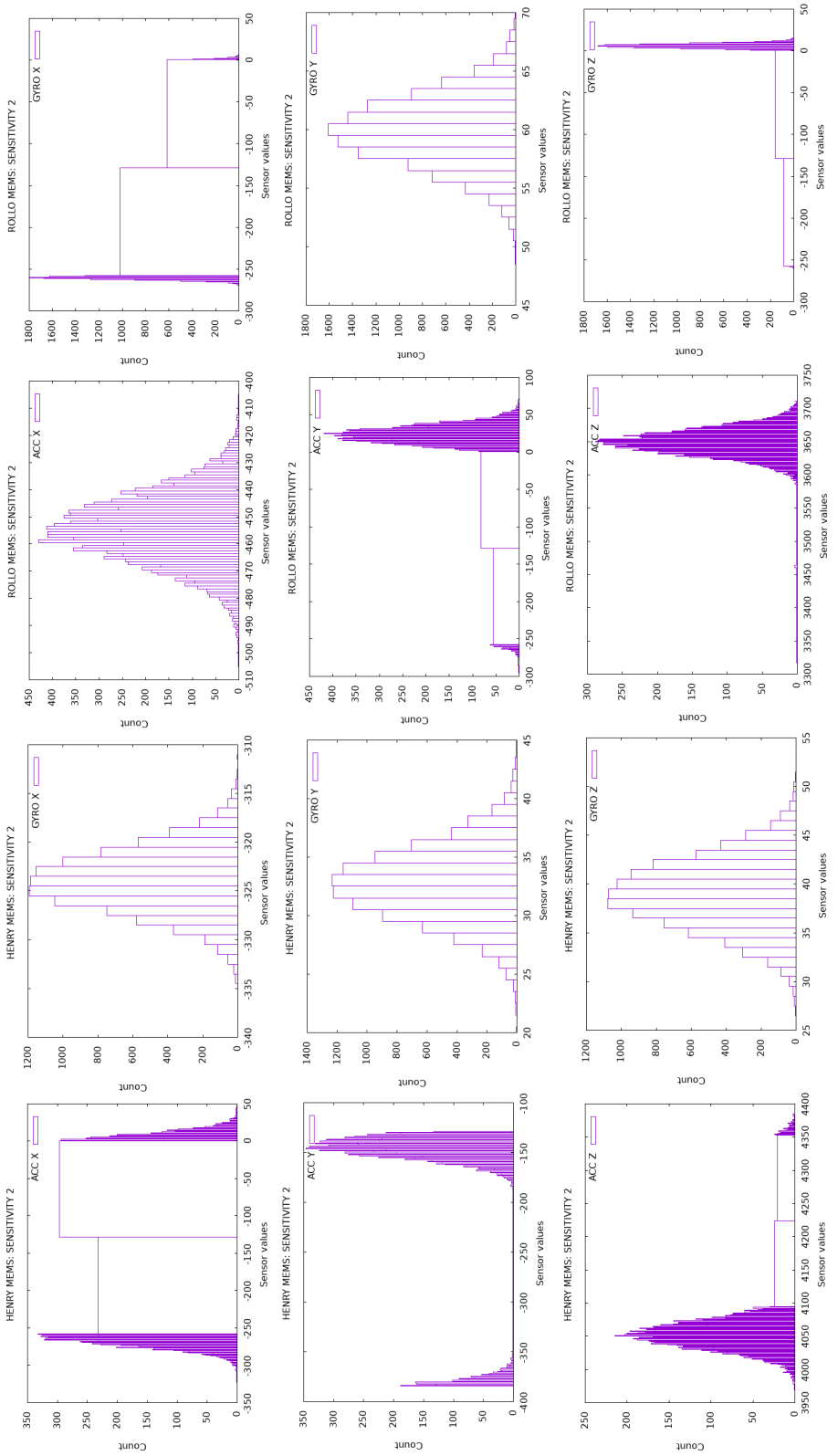
Z grafů je vidět, že senzory nejsou schopny produkovat v klidu konzistentní hodnoty. Zdá se, že prohozením znamének u některých hodnot by histogram více reprezentoval normální distribuci. Některé hodnoty jsou dokonce přesně posunuté tak, že „roztrhnou“ histogram. Pravděpodobně se jedná o chybu senzoru. Nejméně chaoticky vypadá citlivost 3 pro akcelerometry a 1 pro gyroskopy - což jsou rozsahy $\pm 16g$ a $\pm 500°/s$. Tyto citlivosty budou použity v následujících testech.



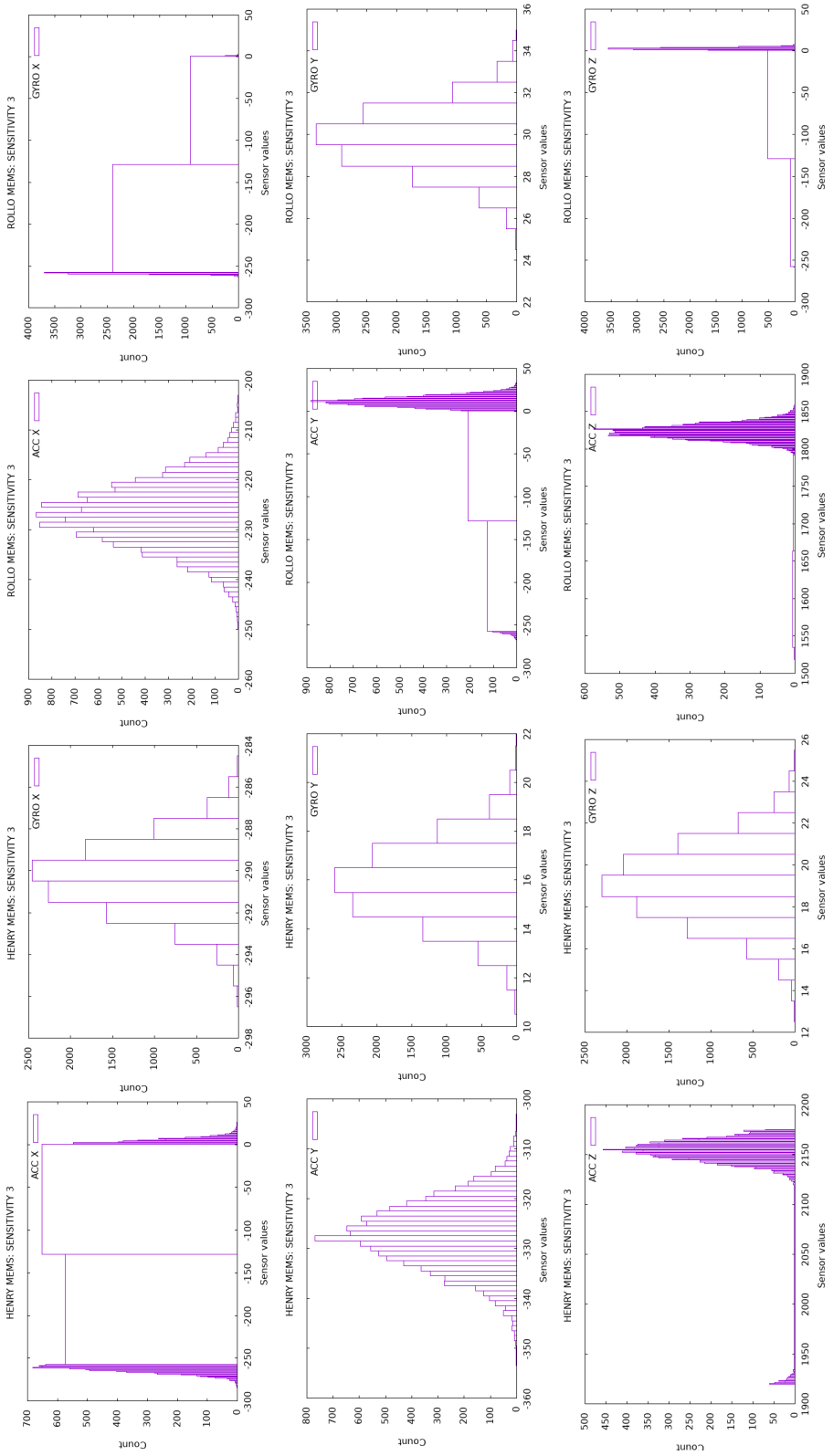
Obrázek 5.5: Histogramy hodnot jednotlivých senzorů testu SENS pro citlivost 0



Obrázek 5.6: Histogramy hodnot jednotlivých senzorů testu SENS pro citlivost 1



Obrázek 5.7: Histogramy hodnot jednotlivých senzorů testu SENS pro citlivost 2



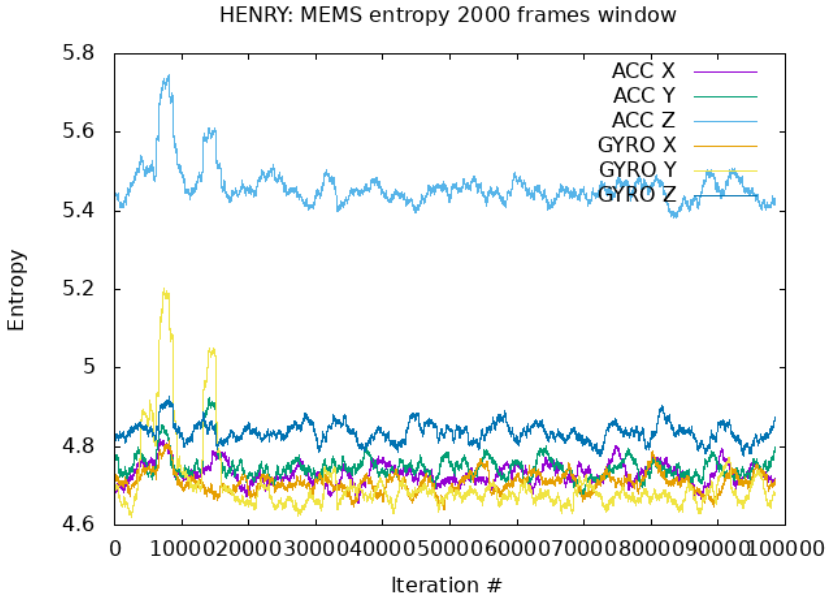
Obrázek 5.8: Histogramy hodnot jednotlivých senzorů testu SENS pro citlivost 3

5.4 Test WARMUP

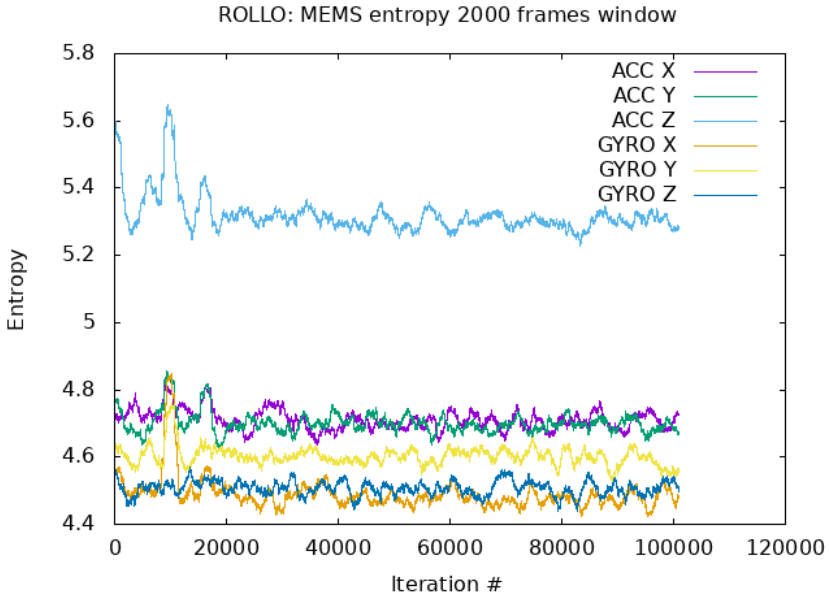
Test má za cíl nalézt zahřívací snímek - tedy rámec, od kterého jsou data smysluplná. Měřen je v klidovém stavu lokalizovaných modulů. Předpokládá, že senzorové hodnoty budou v určité distribuci a že před zahřívacím snímkem budou v jiné - nicneříkající distribuci. Jako sledovací měřítko je zvolena etnropie[27] v okénku posledních 2000 snímků.

Tento test má smysl pouze pro MEMS algoritmy.

Z grafů je vidět, že entropie se ustálí po 20 000 snímcích. Tento 20 000 snímek je použit jako zahřívací v následujících testech.



Obrázek 5.9: Entropie dat jednotlivých senzorů pro dvoutisícového okénko modulu 1



Obrázek 5.10: Entropie dat jednotlivých senzorů pro dvoutisícového okénko modulu 2

5.5 Test STABLE

Toto je poslední konfigurační test.

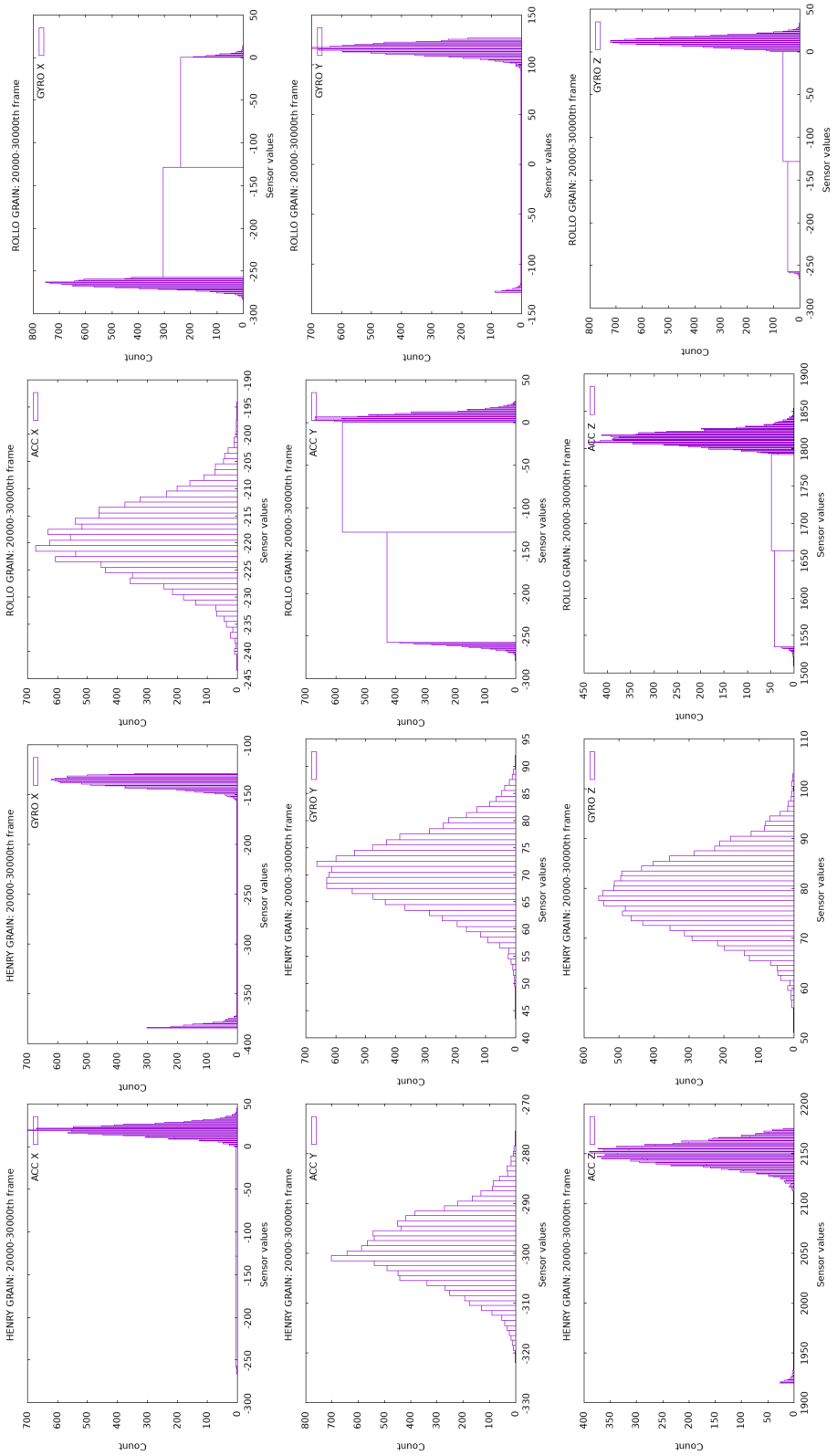
Za úkol má určit kalibrační sekvenci. Tato sekvence je rozmezí snímků počínající zahřívacím snímkem. Za použití CLT[14][16] se bodovým odhadem z těchto kalibračních dat určuje střední hodnota, která slouží jako hodnota šumu.

Tento test je určen pro MEMS algoritmy.

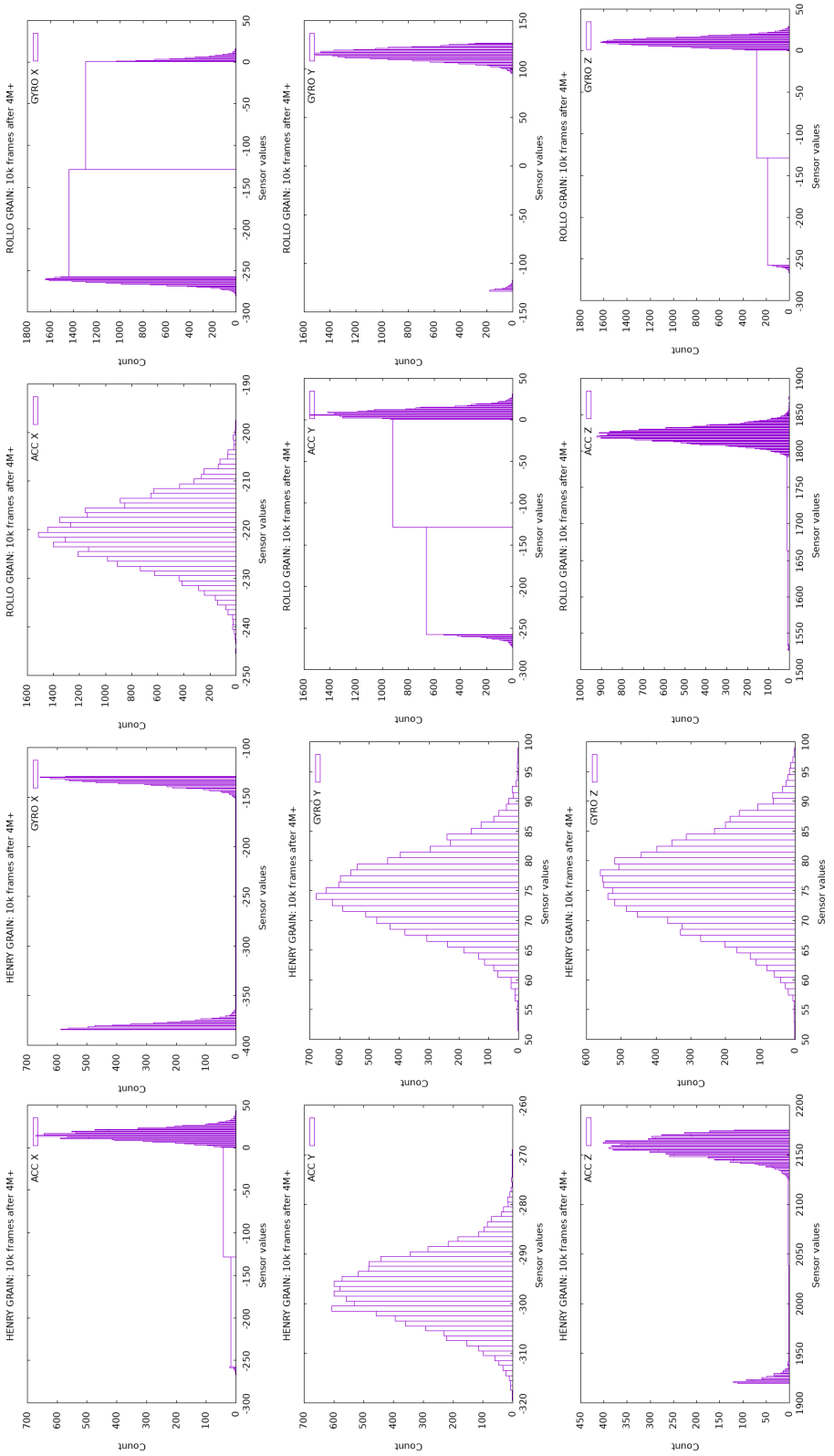
U XB algoritmů by dával smysl akorát u XB.PING. Toto měření by pak pomohlo určit dobu strávenou na zařízeních. Protože použitá Beacons server stanice má nejmenší měřitelnou jednotku stovky nanosekund tak i v nejlepším případě je stanice schopna změřit vzdálenost v násobcích stovek metrů. Čili v tomto bodě nemá smysl nadále pracovat s XBee moduly. Tím pádem odpadají varianty algoritmů COMB.PUMP a COMB.PING. Více je popsáno v závěru práce.

Z grafů je vidět, že 10 000 snímků relativně dobře reprezentuje (až na předem zmíněná špatná znaménka a místy posunutou distribuci) data pro odhad střední hodnoty. Stejné problémy se opakují i pro 10 000 snímků po zahřívácím rámci větším než 4 miliony. Čili zahřívací rámec 20000 rozhodně stačí.

[14] Centrální limitní věta



Obrázek 5.11: Histogramy hodnot jednotlivých senzorů kalibrační sekvence pro 10 000 snímků po zahřívacím snímku. 10 000 iterací = 20 sekund.



Obrázek 5.12: Histogramy hodnot jednotlivých senzorů kalibrační sekvence pro 10 000 snímků po 4 milionech snímků. 10 000 iterací = 20 sekund.

5.6 Test BASIC

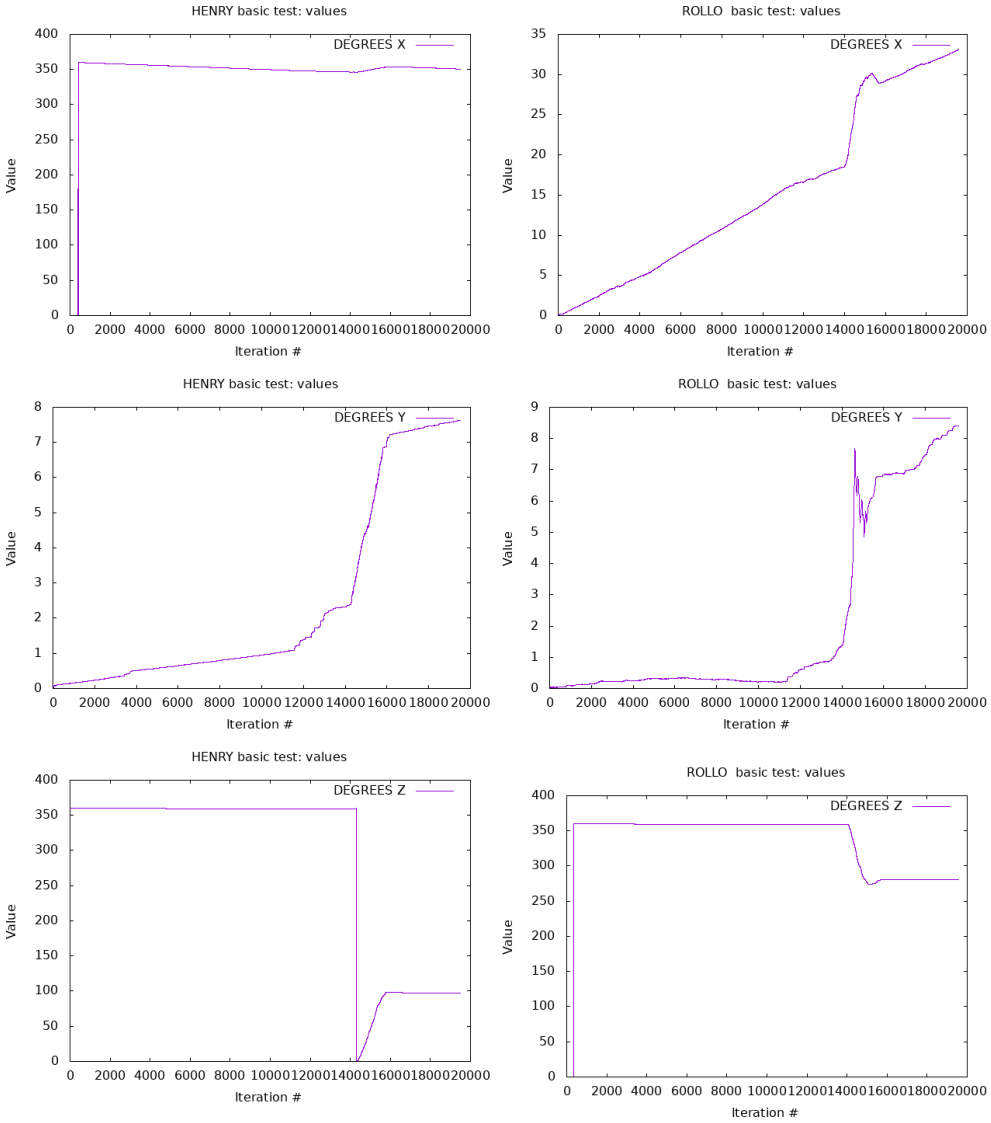
Tento test sleduje odchylku $Or(t_f)$ od $O(t_f)$ a $Pr(t_f)$ od $P(t_f)$ kde tf je čas konce testu. Tento test je prováděn pouze pro MEMS.ORI a MEMS.LOCO algoritmy.

Protože byla zvolena frekvence 500 Hz, tak v následujících grafech znamená, že 10 000 iterací je 20 sekund v realitě.

5.6.1 Test MEMS.ORI

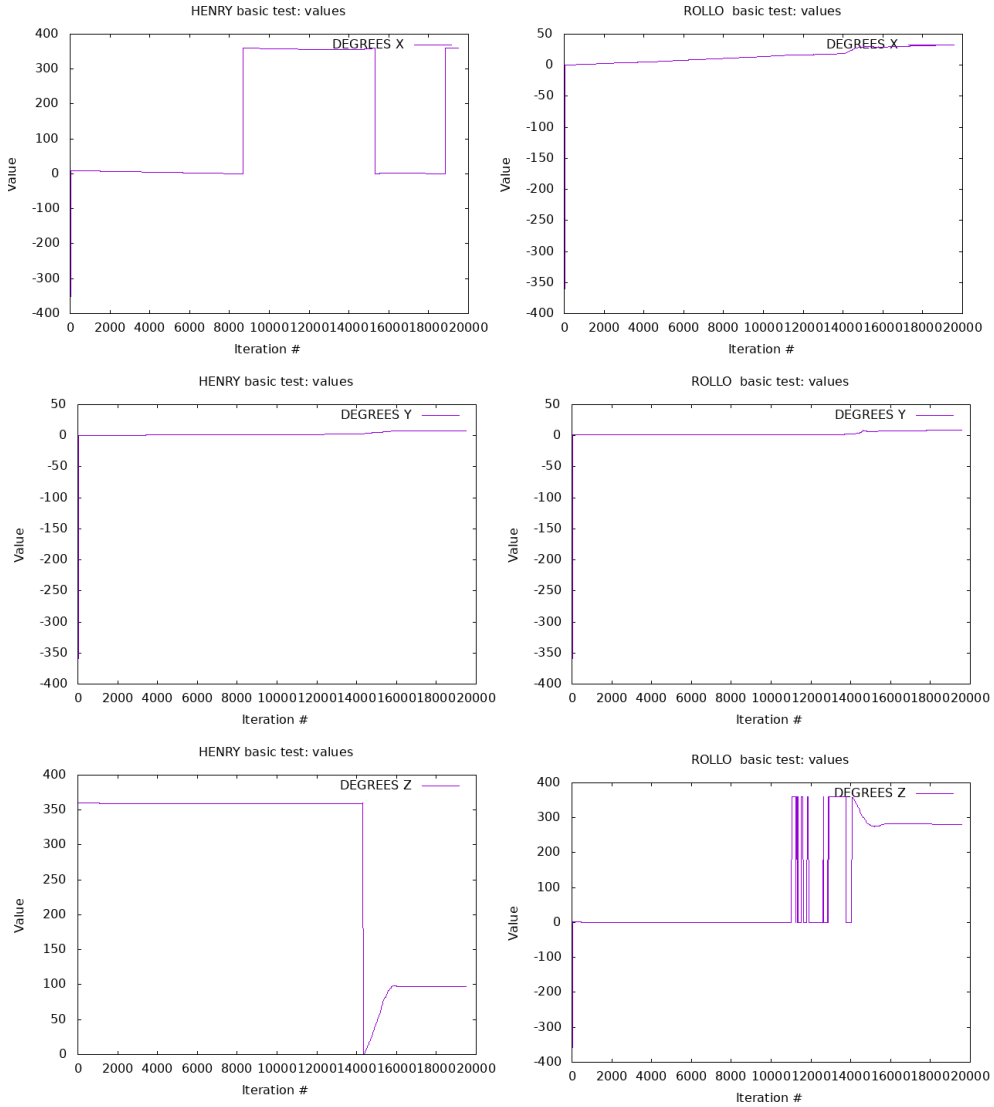
Moduly je otočeno přibližně o 90° doleva nebo doprava. Pro každý lokalizovaný modul pohyb začíná v různých snímcích, probíhá různě rovnoměrně a trvá různou dobu.

Na grafech 5.13 by mělo být vidět otočení kolem osy Z 90° pro modul Henry a pro modul Rollo -90°. Další osy by měly zůstat konstantní. Hodnoty se pohybují v modulu 360.



Obrázek 5.13: Průběh vypočítaných pravotočivých úhlů jednotlivých os podle modulů z 1 snímku. 10 000 iterací = 20 sekund.

Z grafů je viditelný i pohyb o nezanedbatelné stupně v jiných osách. Zavedeme-li jako zdrojová data průměr posledních 25 hodnot (a relevantně zvětšíme *dt* na 25ti násobek), je odhadovaná rotace přesnější. To je vidět v grafech 5.14.



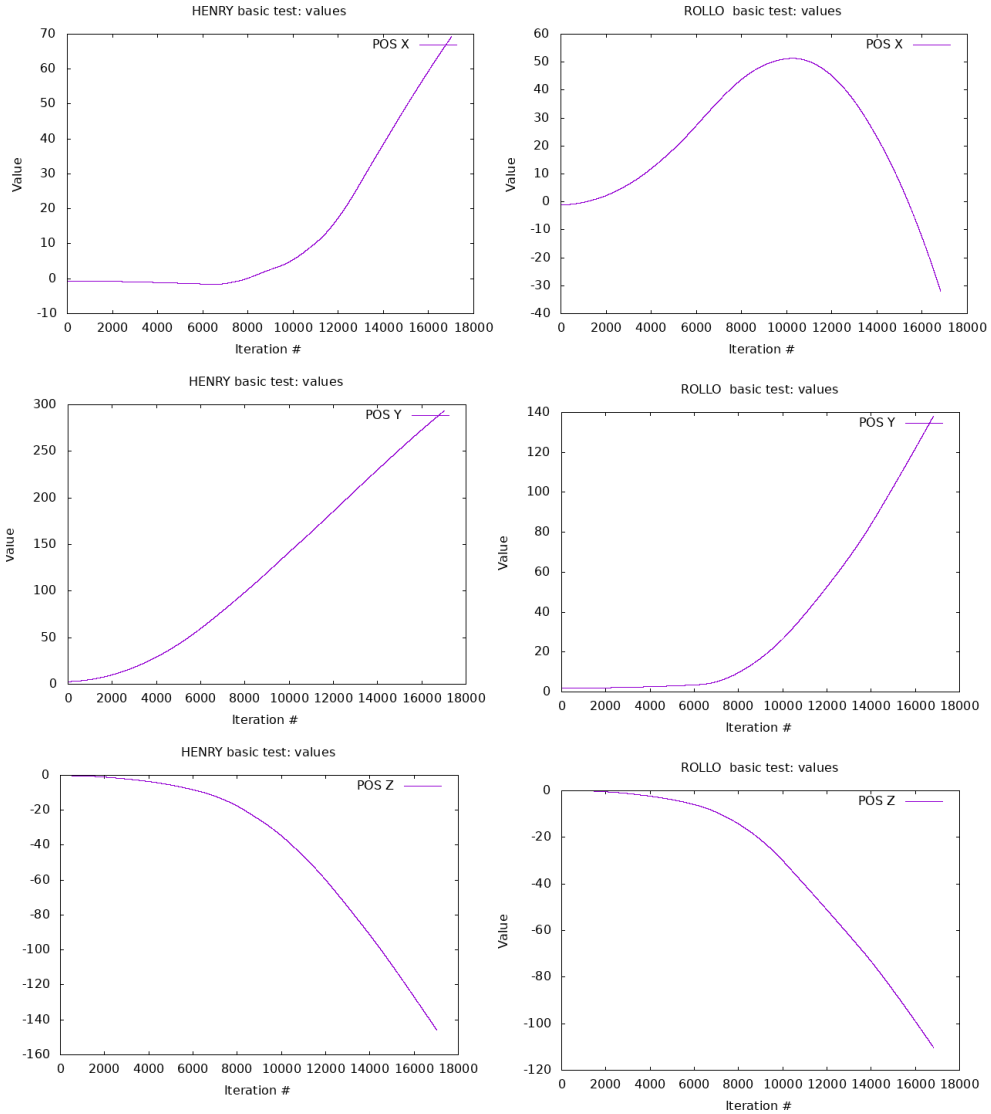
Obrázek 5.14: Průběh vypočítaných pravotočivých úhlů jednotlivých os podle modulů z průměru 25 snímků. 10 000 iterací = 20 sekund.

Pro MEMS.ORI algoritmus má smysl pokračovat v pokročilejších testech s použitím průmeru hodnot z 25 snímků.

5.6.2 Test MEMS.LOCO

Moduly je posunuto zhruba o 1 metr dopředu nebo dozadu. Pro každý lokalizovaný modul pohyb začíná v různých snímcích, probíhá různě rovnoměrně a trvá různou dobu.

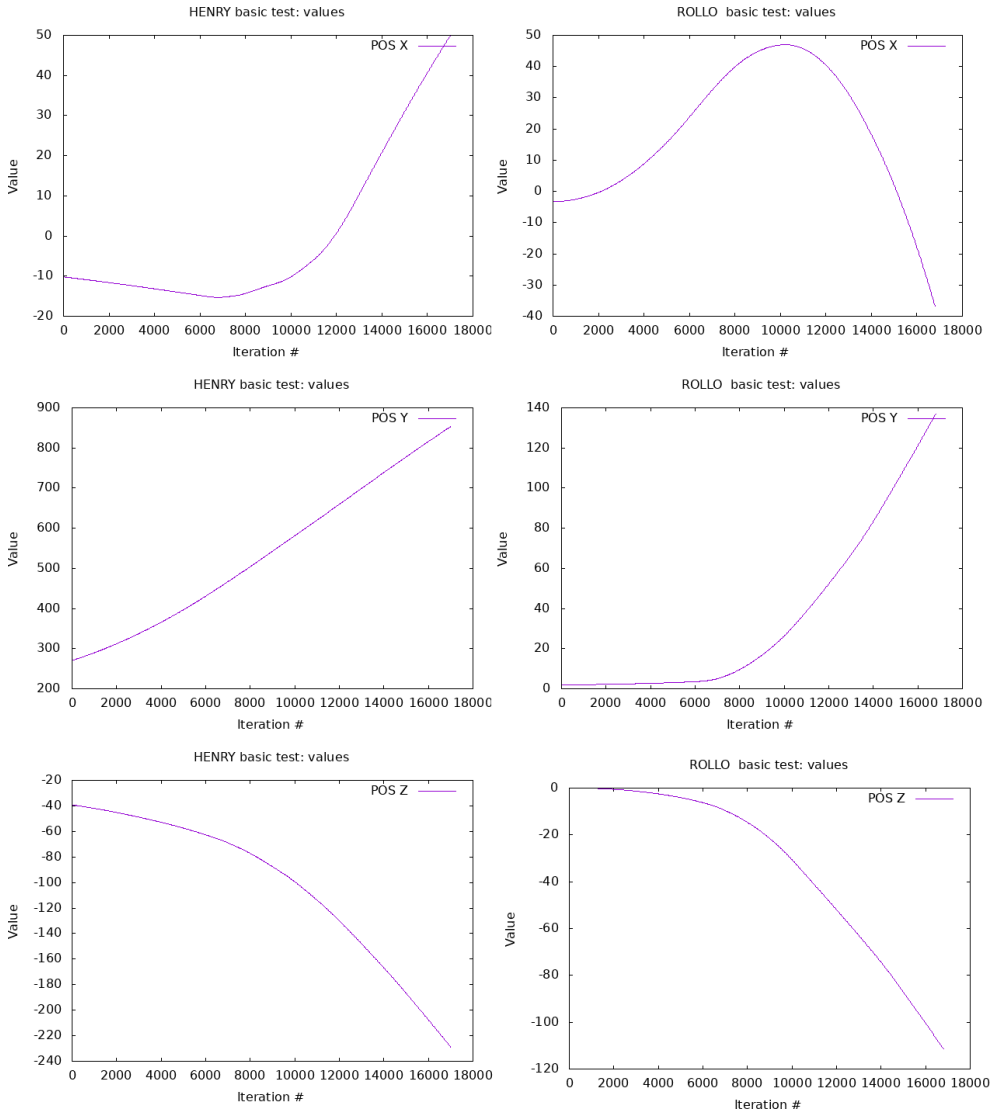
Na grafech 5.15 by měl být vidět pohyb v ose Y o 1 metr blíže k 0 u modulu Henry a o 1 metr dále od 0 u modulu Rollo.



Obrázek 5.15: Průběh vypočítaných pozic jednotlivých os podle modulů v 1 snímku. 10 000 iterací = 20 sekund.

Z grafů je viditelný pohyb i v jiných osách a to o desítky metrů. V ose Y ani není vidět správný směr pohybu. Zavedení průměru jako u MEMS.ORI

nemá pozitivní efekt (viz grafy 5.16).



Obrázek 5.16: Průběh vypočítaných pozic jednotlivých os podle modulů z průměru 25 snímků. 10 000 iterací = 20 sekund.

Za vinu špatného výpočtu dávám nepřesným senzorickým datům - viz rozpojené distribuce z předchozích testů.

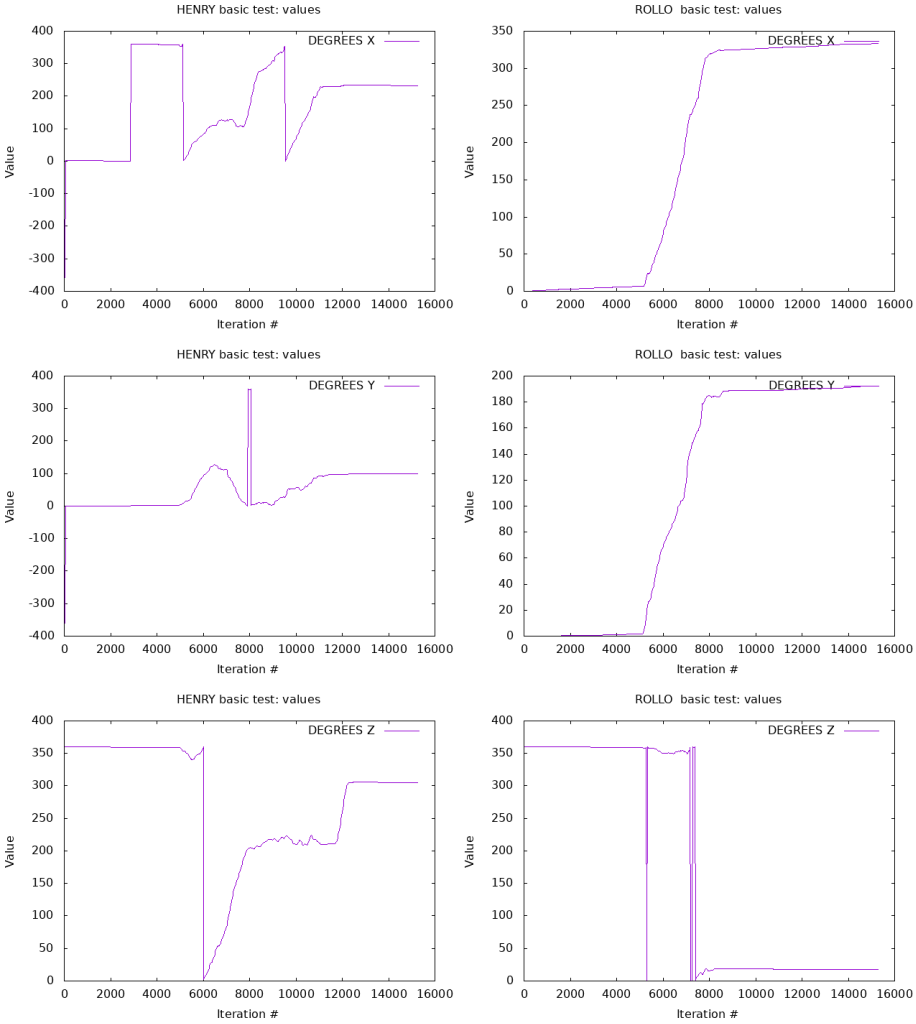
Nemá tedy smysl nadále testovat MEMS.LOCO a MEMS.COMB algoritmy.

5.7 Test ADVANCED

Tento test je prováděn pouze pro algoritmus MEMS.ORI. Hodnoty se pohybují v modulu 360.

5.7.1 Test 1

Moduly jsou nezávisle otáčeny v různých osách v různém pořadí. Pro každý lokalizovaný modul pohyb začíná v různých snímcích, probíhá různě rovnoměrně a trvá různou dobu. Modul Henry by měl končit otočen v ose Z o 90°, v ostatních osách vůbec a Rollo by měl být natočen stejně, jako na počátku testu.

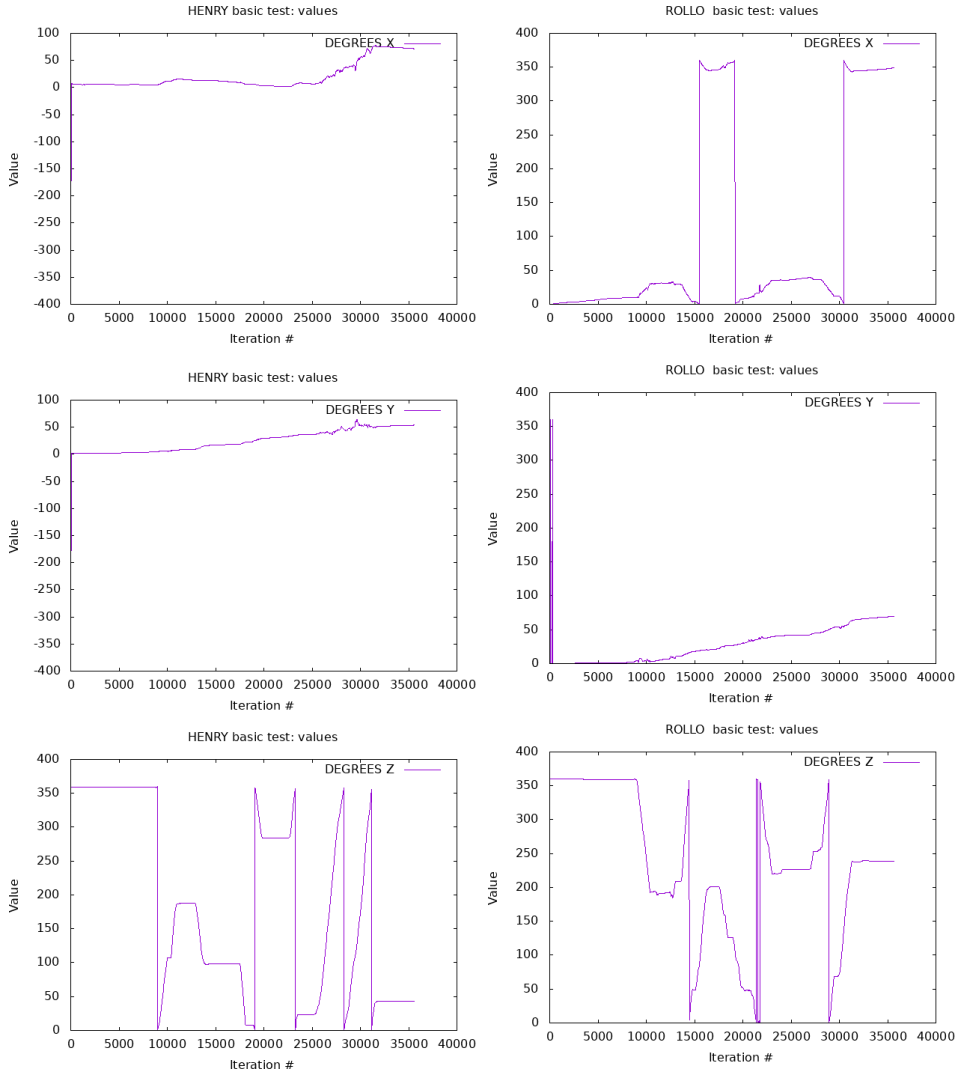


Obrázek 5.17: Test 1: Průběh vypočítaných pravotočivých úhlů jednotlivých os podle modulů z průměru 25 snímků. 10 000 iterací = 20 sekund.

Z grafů je vidět nepřesnost - některé úhly jsou o stovky stupňů mimo očekávanou hodnotu. V osách probíhal nějaký pohyb během testu, ale konečná hodnota by měla být vždy okolo 0. V případě osy Z by měla být hodnota pro modul Henry 90°.

5.7.2 Test 2

Moduly jsou nezávisle otáčeny v různých osách v různém pořadí. Hlavní osou ve které se provádělo otáčení byla osa Z, pro zbylé osy by měla být rotace minimální. Pro každý lokalizovaný modul pohyb začíná v různých snímcích, probíhá různě rovnoměrně a trvá různou dobu. Oba moduly by měly končit s rotačními úhly přibližně 0°.



Obrázek 5.18: Test 2: Průběh vypočítaných pravotočivých úhlů jednotlivých os podle modulů z průměru 25 snímků. 10 000 iterací = 20 sekund.

Z grafů je vidět nepřesnost - některé úhly jsou o stovky stupňů mimo očekávanou hodnotu. Hodnoty se zdaleka nepřibližují realitě.

Kapitola 6

Závěr

V práci byl pedevším definován problém lokalizace, navrženy algoritmy pracující s gyroskopy, akcelerometry a XBee bezdrátovými moduly. Byl implementován systém pro sledování výsledků lokalizace v reálném čase a provedeny testy.

Testy prokázaly, že zvolený levný a dostupný hardware není vhodný pro lokalizaci ve 3D prostoru navrženými algoritmy. Jediný test, který se jevil slibný je přímočará rotace podle 1 osy.

Tabulka 6.1: Tabulka výsledu testů - o = úspěch, x = neúspěch, (prázdno) = test nebyl prováďen

Kód testu	MEMS.ORI	MEMS.LOCO	MEMS.COMB	XB.PUMP	XB.PING	COMB.PUMP	COMB.PING
FREQUENCY	o	o	o	o			
SENS	o	o	o				
WARMUP	o	o	o	x	x		
STABLE	o	o	o				
BASIC	o	x	x				
ADVANCED	x						
FINAL							

Použité MEMS senzory jsou maximálně vhodné pro určení přibližného směru gravitace vzhledem k definované ose čipu a pomalé rotace. Na obrázku 5.11 je vidět, že senzory v klidu neprodukují příliš smysluplná data - distribuce je často rozdělená a posunutá nebo mají hodnoty opačná znaménka. Vzorky příliš nekopírují CLT podle předpokladu. Z tohoto pozorování je patrné, že kvalita výstupu jednotlivých algoritmů nemůže kopírovat realitu.

XBee senzory neumožňují sledovat použitelné informace o přijatých zprávách. V případě, že by senzory umožňovaly číst sílu přijatého výkonu, dal by se použít jiný algoritmus[15]. Dále by bylo vhodné pro zvýšení frekvence lokalizování uvolnění limitu broadcastů za sekundu. Lepší přesnost XB algoritmů v této práci by se uplatnila v případě, že by XBee čip měl vlastní oscilátor a přijaté zprávy značkoval časem v momentě jejich příjmu. Mj. by také čip mohl sledovat o kolikáté přeposlání se jednalo. Algoritmus by si tak mohl vybírat zprávy přijaté napoprvé.

V případě navázání na tuto práci by bylo možno porovnávat jednotlivý levný hardware pomocí navržených testů a hledat ten nejvhodnější. Dále by se daly testy vylepšit jiným, nezávislým systémem, který by uměl plynule pohybovat nebo otáčet lokalizovanými moduly. Dal by se pak porovnávat průběh $Or(t)$ s $O(t)$ a $Pr(t)$ s $P(t)$ kde $Or(t)$ a $Pr(t)$ by byly přesně známy z jiného systému. Vznikla by pak kompletní testovací platforma pro podobný hardware.

Příloha A

Seznam použitých zkratek

GPS Global Positioning System

API Application Program Interface

3D Trojrozměrný

GPIO General Purpouse Input Output

I2C Inter Integrated Circuit

UART Universal Asynchronous Receiver-Transmiter

MEMS Mikro elektro-mechanický systém

ACK Acknowledgement

USB Universal Serial Bus

ASCII American Standard Code for Information Interchange

SID Serial Identification

PAN Personal Area Network

RPI Raspberry Pi

XB XBee

LAN Local Area Network

AABB Axis Aligned Bounding Box

FAA Fetch and Add

FIFO First in First out

LCG Lineární kongruenční generátor

CLT Centrální limitní věta

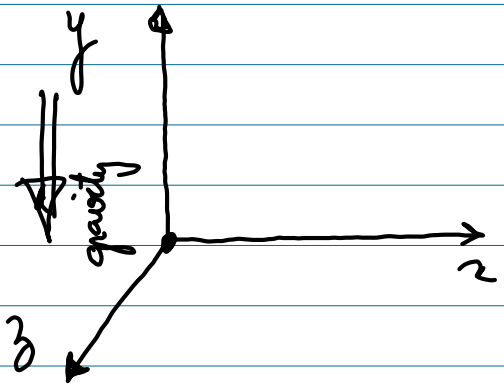
Příloha B

Lokalizace od Ing. Alexandru Mouchy, PhD.

6

$$\begin{cases} y_{i+1} = y_i + v_{y_i} \cdot \Delta t + \dfrac{(a_{y_i} - g)}{2} \Delta t^2 + \dfrac{\Omega_y}{6} \Delta t^3 \end{cases}$$

Part 3: 3D space



On axis Oz equations like on Part 2 Oz.

On axis Oy equations like on Part 2 Oy.

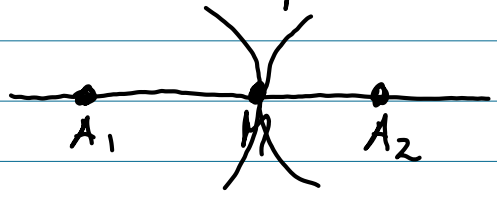
Spin on the device only makes equations worse. At beginning no spin.

Triangulating the position of the module

1. Initial thoughts:

i/ nr of anchors (devices with known position)

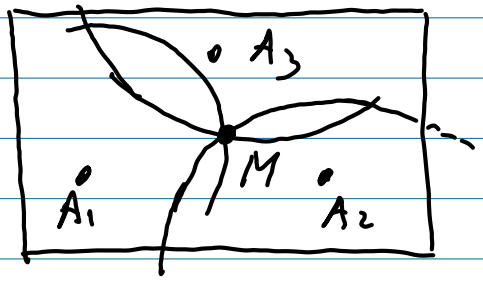
1D (axis) => 2
2D (plane) => 3
3D (space) => 4



based on signal we compute the distance but module M can be

to the left or to the right of A_1 on the axis $\Rightarrow$ we need A_2 to pinpoint the position

2D



ii/ We suppose all the antennas having the same gain $[G]_{dBi}$ in all directions to simplify computation.

iii/ For this computation we suppose that the anchors transmit and the modules receives and that the transmission power is the same for all modules. If these conditions are not followed the computation is more complex but doable.

iii/ We work in far field (distances $\gg \lambda$)

v/ Laws :

in decibels : $[P_R]_{dBm} = [P_T]_{dBm} + [G_T]_{dBi} + [G_R]_{dBi} + [FSPL]_{dB}$

where :

$$FSPL = 20 \log_{10} \frac{c}{4\pi d f} < 0$$

$c = 300\,000\,000$ m/s

$[d]_m$

$[f]_{Hz}$ = frequency

8

in linear units: $[P_R]_{mW} = [P_T]_{mW} \cdot [G_T^*]_- \cdot [G_R^*]_- \cdot [FSPL]_-$

where: $G_T^* = 10^{G_T/10}$ $\quad [G_i]_{dBi}$

$$FSPL = \left(\frac{\lambda}{4\pi d f}\right)^2 < 1$$

Computation 1 in 3D: } - position of anchors are known
- gains are equal and known
- P_T is known
- P_R is measured at module M from each anchor A_i $(i = 1, 2, 3, 4)$

Thus for each anchor A_i $(i = 1, 2, 3, 4)$ compute

$$P_{Ri} = P_T \cdot \underbrace{G_T^*}_{G^*} \cdot \underbrace{G_R^*}_{G^*} \cdot \left(\frac{\lambda}{4\pi d f}\right)^2$$

$$\sqrt{\frac{P_{Ri}}{P_T}} = G^* \cdot \frac{\lambda}{4\pi d_i f} \Rightarrow \frac{4\pi d_i f}{\lambda} = G^* \sqrt{\frac{P_T}{P_{Ri}}}$$

$$\Rightarrow \boxed{d_i = \frac{\lambda G^*}{4\pi f} \sqrt{\frac{P_T}{P_{Ri}}}}$$

9

Thus we know $A_1 = (x_1, y_1, z_1)$ $A_3 = (x_3, y_3, z_3)$
$A_2 = (x_2, y_2, z_2)$ $A_4 = (x_4, y_4, z_4)$

d_1, d_2, d_3, d_4 distances from $A_1 \dots A_4$

$P_T, G^*, P_{R1}, P_{R2}, P_{R3}, P_{R4}, f, c, \bar{u}$:)

We need (x_m, y_m, z_m)

But it is difficult to determine the gains and P_T. What we can do is to start the network from a known position of the module.

Computation 2 in 3D:

Anchors are fixed $A_i(x_i, y_i, z_i)$

Module is known $M(x_m, y_m, z_m)$

Based on this we compute $\boxed{d_i = \sqrt{(x_m - x_i)^2 + (y_m - y_i)^2 + (z_m - z_i)^2}}$

and $\boxed{d_i = \frac{c G^*}{4 \bar{u} f} \sqrt{\frac{P_T}{P_{R_i}}}}$

Thus: $P_{Ri} = P_T \cdot G^{*2} \left(\frac{c}{4 \bar{u} d_i f} \right)^2 \Rightarrow \frac{c G^* \sqrt{P_T}}{4 \bar{u} f} = d_i \sqrt{P_{R_i}}$

10

Now we move the module to $(x_M' ; y_M' ; z_M')$ and thus to

$$d_i' = \underbrace{\frac{\lambda G^* \sqrt{P_T}}{4\pi}}_{d_i \sqrt{P_{R_i}}} \cdot \sqrt{P_{R_i}'} \Rightarrow \boxed{d_i' = d_i \sqrt{\frac{P_{R_i}'}{P_{R_i}}}}$$

where P is measured in mW

and this makes the computation much easier.

Příloha C

Obsah přiloženého média

`přiložené médium` ..... hlavní adresář obsahující mj. buildovací skripty
- `baselib` adresář se zdrojovými soubory obecné funkcionality (práce se soubory, sítí atp.)
- `build` ..... adresář se zkompilovanými soubory
  - `data` .... adresář s konfiguračními soubory a bitmapovým písmem
  - `records` ..... adresář pro výstup nahrávek z realtime lokalizace
- `issues` adresář s testy - zdrojové soubory, binární aplikace, upravené nahrávky a grafy
- `sources` ..... adresář se zdrojovými soubory pro specifické aplikace lokalizačního systému
- `DP.tex` ..... zdrojový kód práce
- `DP.pdf` ..... elektronická verze práce ve formátu PDF

Literatura

[1] MAXLASERGAME s.r.o. [online]. [cit. 1. ledna 2019]. Dostupné z: `http://www.maxlasergame.cz`

[2] RASPBERRY PI FOUNDATION. *Raspberry Pi 3 B V1.2* [hardware].

[3] RASPBERRY PI FOUNDATION. *Raspbian* [software]. [cit. 1. ledna 2019]. Dostupné z: `https://www.raspberrypi.org/downloads/raspbian/`

[4] INVENSENSE. *MPU6050* [hardware].

[5] ARDUINO S.r.l. *GY-521* [hardware].

[6] MAX STREAM. *XBee^{TM} Series 2 OEM RF Module* [hardware].

[7] MICROSOFT. *Windows 10* [software]. [cit. 1. ledna 2019]. Dostupné z: `https://www.microsoft.com/cs-cz/software-download/windows10`

[8] NXP SEMICONDUCTORS N.V. *I2C-bus specification and user manual* [online]. UM10204. Rev. 6. 4. dubna 2014. Dostupné z: `https://www.nxp.com/docs/en/user-guide/UM10204.pdf`

[9] TEXAS INSTRUMENTS. *KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide* [online]. Listopad 2010. Dostupné z: `http://www.ti.com/lit/ug/sprugp1/sprugp1.pdf`

[10] INVENSENSE. *MPU-6000 and MPU-6050 Product Specification* [online]. Revision 3.4. Srpen 2013. Dostupné z: `https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V34.pdf`

[11] HENDERSON, G. *WiringPi* [software]. [cit. 1. ledna 2019]. Dostupné z: `http://wiringpi.com/`

[12] INVENSENSE. *MPU-6000 and MPU-6050 Register Map and Descriptions* [online]. Revision 4.2. Srpen 2013. Dostupné z: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

[13] MAX STREAM. *XBee^{TM} Series 2 OEM RF Modules* [online]. Červenec 2007. Dostupné z: http://www.farnell.com/datasheets/27606.pdf

[14] IEEE COMPUTER SOCIETY. *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)* [online]. IEEE Standard for Local and metropolitan area networks. Září 2011. Dostupné z: http://ecee.colorado.edu/~liue/teaching/comm_standards/2015S_zigbee/802.15.4-2011.pdf

[15] MOUCHA, A. *Emailová konzultace* [online]. Duben 2018.

[16] BLAŽEK, R., HRABÁKOVÁ J., HRABÁK P. *Limitní vety* [přednáška]. 2017. In: courses.fit.cvut.cz [cit. 2019-01-06]. Dostupé z: https://courses.fit.cvut.cz/MI-SPI/media/lectures/mi-spi-lec-08-slides.pdf

[17] BLAŽEK, R., HRABÁKOVÁ J., HRABÁK P. *Statistika* [přednáška]. 2017. In: courses.fit.cvut.cz [cit. 2019-01-06]. Dostupé z: https://courses.fit.cvut.cz/MI-SPI/media/lectures/mi-spi-lec-09-slides.pdf

[18] BEDNAŘÍK, M., ŠIROKÁ, M., BUJOK, P. *Fyzika pro gymnázia: Mechanika.* 3. vydání. Praha. Prometheus, 2002.

[19] POSTEL, J. *Internet control message protocol* [online]. RFC 792, Network Working Group. Září 1981. Dostupné z: https://tools.ietf.org/html/rfc792

[20] INFORMATION SCIENCES INSTITUTE. *Transmission control protocol* [online]. RFC 793, University of Southern California. Září 1981. Dostupné z: https://tools.ietf.org/html/rfc793

[21] MICROSOFT. *QueryPerformanceCounter function* [software]. [cit. 1. ledna 2019]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms644904(v=vs.85).aspx

[22] MICROSOFT. *QueryPerformanceFrequency function* [software]. [cit. 1. ledna 2019]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms644905(v=vs.85).aspx

[23] FREE SOFTWARE FOUNDATION, Inc. *clock_gettime(3) - Linux man page* [software]. [cit. 1. ledna 2019]. Dostupné z: https://linux.die.net/man/3/clock_gettime

[24] KLOUDA, K., KALVODA T., STAROSTA Š. *Strojová čísla* [přednáška]. 2018. In: courses.fit.cvut.cz [cit. 2019-01-06]. Dostupé z: https://courses.fit.cvut.cz/MI-MPI/latex/lectures/czech/mi-mpi-prednaska-20-strojova-cisla.pdf

[25] FIEDLER, G. *Fix Your Timestep!* [online]. Červen 2004. Dostupné z: https://gafferongames.com/post/fix_your_timestep/

[26] ISO/IEC 9899:201x. *Programming languages - C*. Strana 347. Prosinec 2010. Dostupné z: https://courses.fit.cvut.cz/MI-KOD/lectures/mi-kod-01-intro.pdf

[27] HOLUB, J. *Data Compression. Introduction* [přednáška]. 2018. In: courses.fit.cvut.cz [cit. 2019-01-06]. Dostupé z: https://courses.fit.cvut.cz/MI-KOD/lectures/mi-kod-01-intro.pdf