
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**
**DEPARTMENT OF INTELLIGENT SYSTEMS**


# SLEDOVÁNÍ POLOHY PRSTU VE 3D PROSTORU

**FINGER POSITION TRACKING IN 3D SPACE**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **MILOŠ HEGR**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. PETR MALANÍK**
**SUPERVISOR**



**BRNO 2023**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání bakalářské práce

144050

| | |
|---|---|
| Ústav: | Ústav inteligentních systémů (UITS) |
| Student: | **Hegr Miloš** |
| Program: | Informační technologie |
| Specializace: | Informační technologie |
| Název: | **Sledování polohy prstu ve 3D prostoru** |
| Kategorie: | Zpracování obrazu |
| Akademický rok: | 2022/23 |

Zadání:

1. Seznamte se se zařízením sloužícím pro sběr prostorových dat.
2. Nastudujte možnosti použití integrovaných senzorů za účelem rekonstrukce pohybu ve 3D prostoru.
3. Připravte firmware zařízení umožňující přenos dat ze senzorů přes Bluetooth. Program v zařízení by zároveň měl umožňovat předzpracování dat ze senzorů.
4. Navrhněte knihovnu umožňující příjem dat na PC připojenému k zařízení. Knihovna také musí být schopna určit základní data odvozená ze senzorických dat a vše být schopna exportovat.
5. Funkčnost knihovny demonstrujte integrací ovládání do demonstrační aplikace nebo libovolného CAD nástroje.
6. Zhodnoťte dosažené výsledky. Zaměřte se na to jak získaná data odpovídají realitě a určete zpoždění přenosu.

Literatura:


- Steven W. Smith ,The Scientist & Engineer's Guide to Digital Signal Processing, California Technical Pub, 1997, ISBN 0966017633.
- EDWARDS, Lewin A. R. W. Embedded system design on a shoestring. Boston: Newnes, 2003. ISBN 978-0-7506-7609-0.
- YAHYA, Muhammad, et al. Motion capture sensing techniques used in human upper limb motion: a review. Sensor Review, 2019.


Při obhajobě semestrální části projektu je požadováno:
Splnění bodů 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

| | |
|---|---|
| Vedoucí práce: | **Malaník Petr, Ing.** |
| Vedoucí ústavu: | Hanáček Petr, doc. Dr. Ing. |
| Datum zadání: | 1.11.2022 |
| Termín pro odevzdání: | 10.5.2023 |
| Datum schválení: | 3.11.2022 |

# Abstrakt

Cílem této práce je vytvořit univerzální systém pro měření dat popisující polohu ve 3D prostoru. Tato práce přináší teoretický přehled o popisu pozice ve 3D prostoru a ukazuje možnosti rekonstrukce pozice z dat senzorů. Dále podává bližší pohled na využité hardwarové zařízení RingOne a navrhuje i implementuje systém pro rekonstrukci polohy prstu s využitím tohoto zařízení. Tento systém se skládá z firmwaru pro měřící zařízení RingOne, knihovny jazyka C umožňující jednoduchou komunikaci s měřícím zařízením a protokolu definující způsob přeposílání dat a způsob konfigurace měřícího zařízení. Nakonec tuto práci doplňuje také upravený ovladač Spacenavd pro jednoduchou prezentaci naměřených dat.



# Abstract

The aim of this work is to create a universal system for measuring data describing position in 3D space. This paper provides a theoretical overview of position description in 3D space and shows the possibilities of position reconstruction from sensor data. It gives a closer look at the RingOne hardware device, and finally proposes and implements a system for finger position reconstruction using this device. This system consists of the firmware for the RingOne device, a C language library allowing easy communication with the measuring device and a protocol defining the way data is transmitted and how the measuring device is configured. This work is also complemented by a modified Spacenavd driver for easy presentation of the measured data.



# Klíčová slova

3D prostor, 3D orientace, kartézský souřadnicový systém, cylindrický souřadnicový systém, sférický souřadnicový systém, Eulerovy úhly, komplementární filter, akcelerometr, gyroskop, Bluetooth Low Energy

# Keywords

3D space, 3D orientation, cartesian coordinate system, cylindrical coordinate system, spherical coordinate system, Euler angles, complementary filter, accelerometer, gyroscope, Bluetooth Low Energy

# Citace

HEGR, Miloš. *Sledování polohy prstu ve 3D prostoru.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Malaník

# Sledování polohy prstu ve 3D prostoru

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Malaníka.

.....

Miloš Hegr

10. května 2023

# Obsah

# Kapitola 1

# Úvod

Celý život vnímáme svět ve 3D, ovšem většina běžně používaných uživatelských vstupů počítače umí popsat pohyb pouze ve 2D, jako například klasická počítačová myš. Tato práce za zabývá možností popisu 3D dat, návrhem programové realizace zařízení umožňující snímat pohyb právě v třídimenzionálním prostoru a následnou prezentací naměřených dat ve známých 3D programech.

Objekt ve 3D prostoru může být sledován pomocí externích senzorů, jako jsou kamery nebo lasery. Ne vždy je ovšem využití externích senzorů vhodné či dokonce možné. Z tohoto důvodu existují zařízení zvané „Inerciální měřící jednotky“. Tato zařízení mohou být připevněna k měřenému objektu a umožňují měřit hodnoty, jako jsou například úhlová rychlost otáčení objektu, zrychlení se kterým se objekt pohybuje a jiné další hodnoty v závislosti na daném zařízení. Z těchto hodnot je následně možné rekonstruovat celkový pohyb objektu. Ovšem většina těchto komerčních měřících zařízení jsou drahá (desítky tisíc korun) a nemohou fungovat jako samostatné zařízení, ale vyžadují jiná zařízení pro zpracování naměřených dat.

Výsledné měřící zařízení je vhodné pro širokou škálu aplikací. Může být využito pro měření polohy prstu při jeho skenování za účelem získání otisků tohoto prstu nebo pro zcela jiné účely, jako je například měření orientace kamery využívané pro detekci laserového paprsku a určení pozice jeho zdroje, nebo také záznam letových dat kluzáku pro vyhodnocení efektivity pilotova řízení.

Cílem této práce je vytvořit univerzální měřící systém, jehož součástí bude nejen univerzální měřící zařízení, které bude schopné poskytovat prostorová data dle uživatelovi potřeby, ale také programová knihovna umožňující uživateli jednoduchý přístup ke komunikaci se zmíněným zařízením. Měřící zařízení bude komunikovat s uživatelem speciálním protokolem navrženým na míru této aplikaci. Tento protokol využívá bezdrátovou technologii *Bluetooth Low Energy* a umožňuje efektivní přenos naměřených dat a také rozsáhlou konfiguraci měřícího zařízení.

# Kapitola 2

# Teoretický přehled

## 2.1 Určení pozice ve 3D prostoru

Určit pozici v 3D prostoru je hlavní cíl této práce, ovšem před samotným určováním je důležité definovat základní pojmy a principy. V následujících sekcích tedy bude popsán způsob popisu určité pozice a orientace a nakonec vysvětlena samotná rekonstrukce pohybu.

### 2.1.1 Souřadnicové systémy

Prvním krokem v určování pozice v prostoru je definování popisu dané pozice. K tomuto účelu existuje nemalé množství souřadnicových systémů. Takové systémy nám umožní dle daných pravidel přesně popsat cílenou pozici. Tato sekce vychází z [23].

#### Kartézský souřadnicový systém

Jedním z nejpoužívanějších souřadnicových systémů je kartézský souřadnicový systém. Základem tohoto sytému jsou tři vzájemně na sebe kolmé osy, běžně popisované jako $x, y, z$.

Pozice v tomto systému je popsána jako vektor tří čísel, kde každé číslo odpovídá jedné ose v daném pořadí. Význam těchto čísel lze vysvětlit více způsoby, ovšem u všech způsobů popisují stejná čísla stejnou pozici. Všechna tato čísla také popisují určitý vztah k bodu počátku, což je bod průniku všech tří hlavních os.

- **Promítání vzdálenosti**
  První způsob interpretuje čísla jako vzdálenost od určité roviny, například v případě čísla pojící se s osou $x$, dané číslo reprezentuje vzdálenost od roviny popsanou zbylými osami, tedy osami $y, z$. Výsledný bod se nachází v průsečíku všech tří rovin posunutých od počátku o danou vzdálenost ve směru dané osy.

- **Rovnoběžné vektory**
  Druhým způsobem je interpretace čísla jako vektoru rovnoběžného s odpovídající osou a jeho počátkem na rovině tvořenou taktéž zbylými osami. Výsledný bod v rovině je určen konečným vektorem se začátkem v počátečním bodě. Tento konečný vektor představuje vektorový součet všech tří vektorů.

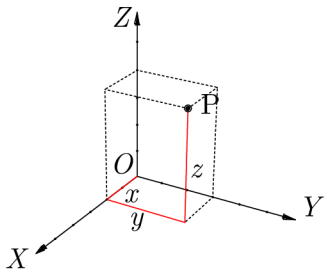
- **Násobky jednotkových vektorů**
  Další způsob interpretace zadaných čísel může být pohlížení na čísla jako na násobky

odpovídajících vektorů, kde výsledný bod je stejně jako v předchozím způsobu reprezentován konečným vektorem. Ovšem konečný vektor představuje vztah:

$$\vec{p} = a\vec{x} + b\vec{y} + c\vec{z} \tag{2.1}$$

kde $\vec{p}$ je konečný vektor, $(a, c, b)$ jsou skalární hodnoty popisující pozici a $\vec{x}, \vec{y}, \vec{z}$ jsou jednotkové vektory na odpovídající ose.

Na obrázku 2.1 je znázorněn popis polohy bodu $P$ v kartézském souřadnicovém systému.



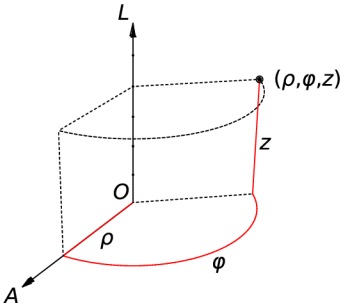

Obrázek 2.1: Kartézský souřadnicový systém, převzato a upraveno z [36].

### Cylindrický souřadnicový systém

Podobně jako u kartézského souřadnicového systému tvoří jeho základ tři navzájem kolmé osy a pozice se popisuje třemi hodnotami. Rozdíl je ovšem ve významu těchto číselných hodnot. Nejdříve si musíme zvolit hlavní rovinu tohoto souřadnicového systému (běžně rovina $xy$) a referenční směr na této rovině (běžně směr $[x = 0, z = 0]$). Poté můžeme jeden bod popsat jako:

$$P = (\rho, \varphi, z) \tag{2.2}$$

kde $\rho$ je vzdálenost projekce popisovaného bodu na hlavní rovině od počátku, $\varphi$ je azimut[1], neboli úhel mezi referenčním směrem a spojnicí projekce popisovaného bodu na hlavní rovině a počátku, a $z$ je výška, neboli vzdálenost hlavní roviny a popisovaného bodu. Tento popis souřadnicového systému je znázorněn na obrázku 2.2, kde bod $O$ značí počátek, $L$ značí hlavní svislou osu a $A$ znázorňuje dodatečnou osu tvořící referenční rovinu a také referenční směr.




Obrázek 2.2: Cylindrický souřadnicový systém, převzato a upraveno z [37].

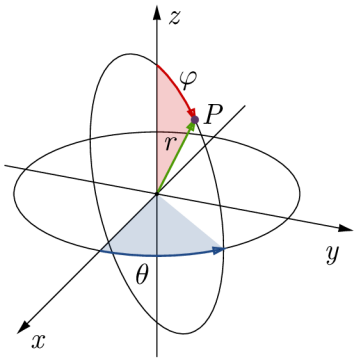
[1] Viz https://en.wikipedia.org/wiki/Azimuth.

#### Sférický souřadnicový systém

Také u tohoto souřadnicového systému jsou základem tři navzájem kolmé osy a tři číselné hodnoty. Před vysvětlením významu jednotlivých hodnot je také zde potřeba zvolit hlavní rovinu a na ní ležící referenční směr. Následně můžeme požadovaný bod definovat jako:

$$P = (r, \theta, \varphi) \tag{2.3}$$

kde $r$ je vzdálenost požadovaného bodu od počátku, $\theta$ je azimut, neboli úhel mezi referenčním směrem a spojnicí projekce popisovaného bodu na hlavní rovině a počátku, a $\varphi$ je inklinace[2], neboli úhel mezi kolmicí na hlavní rovinu a spojnicí požadovaného bodu a počátku.




Obrázek 2.3: Sférický souřadnicový systém, převzato a upraveno z [21].

V dalších částech bude používán kartézský souřadnicový systém ve formátu $p = (x, y, z)$ s osou $X$ popisující pohyb dopředu a dozadu, osou $Y$ popisující pohyb doprava a doleva a s osou $Z$ popisující pozici směrem nahoru a dolů.

### 2.1.2 Orientace ve 3D prostoru

Další důležitou vlastností objektu pohybujícího se ve 3D prostoru je jeho orientace. Orientace popisuje rotaci os souřadnicového systému tělesa vůči hlavním osám globálního souřadnicového systému 3D prostoru. Existuje několik hlavních způsobů jak orientaci daného tělesa popisovat. Tato sekce vychází z [23].

#### Osa-úhel

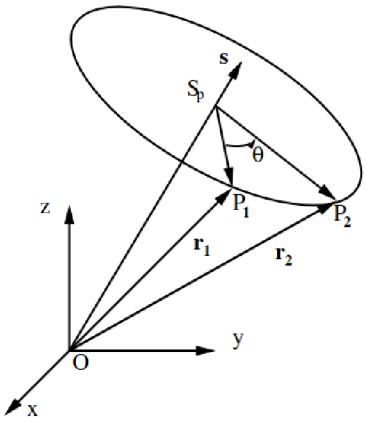
Eulerova věta o rotaci říká, že orientaci objektu lze vyjádřit jako otočení o úhel $\theta$ kolem osy $s$ relativně ke globálnímu souřadnicovému systému. Takový popis orientace můžeme vyjádřit jako čtveřici čísel ve tvaru:

$$R = [w, x, y, z] \tag{2.4}$$

kde $[x, y, z]$ je vektor ve směru osy otáčení a $w$ udává úhel otočení.

Z obrázku 2.4 je tento princip patrný, vektory $x, y, z$ znázorňují globální souřadnicový systém, vektory $r_1, r_2$ znázorňují postupně původní a rotovaný souřadnicový systém objektu, vektor $s$ je osa otáčení a $\theta$ je samotný úhel otáčení.

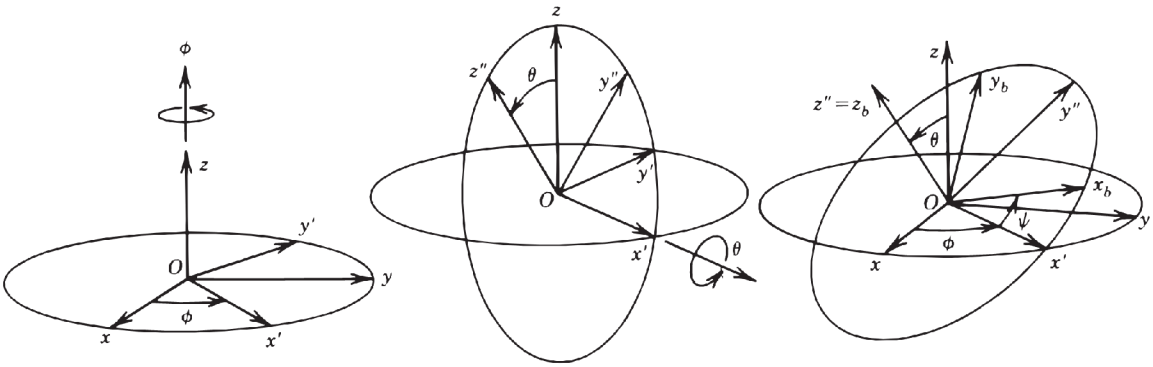
[2] Viz https://en.wikipedia.org/wiki/Orbital_inclination.




Obrázek 2.4: Popis orientace pomocí principu osa-úhel, převzato a upraveno z [46].

### Eulerovy úhly

Eulerovy úhly jsou intuitivní způsob popisu orientace tělesa. Orientace je popsána vektorem tří čísel, kde každé číslo udává otočení kolem určité osy souřadnicového systému tělesa. Otáčení kolem os je nekomutativní operace, a tedy pořadí os, kolem kterých se rotuje, je důležité. Při jeho nedodržení může být výsledkem zcela jiná orientace. Rotace nemusí být provedena okolo všech tří os, ale může být rotována pouze kolem dvou, je ovšem důležité provádět rotaci třikrát a každé dvě sousední rotace musí mít jinou osu otáčení. Orientace tedy může být pomocí Eulerových úhlů popsána například jako rotace kolem os $z, x, z$ s vektorem úhlů otáčení $r = [\phi, \theta, \psi]$. Tuto rotaci zobrazuje obrázek 2.5.




Obrázek 2.5: Postupná orientace pomocí Eulerových úhlů, převzato a upraveno z [45].

V letectví se orientace letadla udává jako trojice úhlů zvaná "yaw, pitch, roll". Jedná se o speciální typ Eulerových úhlů, kde podle definice souřadnicového systému z konce sekce 2.1.1 je pořadí os otáčení $z, y, x$ a hodnoty úhlů jednotlivých rotací jsou označovány jako $\psi, \theta, \phi$ [1]. V dalších částech bude používán právě tento popis orientace.

#### Kvaterniony

Kvaterniony jsou speciální způsob popisu rotací využívající komplexních čísel rozšířených do čtyř-dimenzionálního prostoru. S takto rozšířenými komplexními čísly je možné přesně popsat bod na jednotkové hyperkouli a díky tomu přesně určit orientaci ve 3D prostoru. Kvaternion může být zapsán jako čtveřice čísel, které popisují pozici daného bodu ve 4D kartézském souřadnicovém systému. Jelikož popisovaný bod musí ležet na jednotkové hyperkouli, pro vektor čísel popisující bod $P$ jako vektor $[a, b, c, d]$ by měla platit rovnost vyjádřená rovnicí 2.5.

$$a^2 + b^2 + c^2 + d^2 = 1 \tag{2.5}$$

Kvaternion je také možné jednoduše odvodit z popisu osa-úhel, viz sekci 2.1.2, pomocí vztahu:

$$p = \cos(w/2) + x\sin(w/2)i + y\sin(w/2)j + z\sin(w/2)k \tag{2.6}$$

kde $[x, y, z]$ je jednotkový vektor ve směru osy otáčení, $w$ je úhel otáčení a $i, j, k$ označují imaginární složky [42].

### 2.1.3 Rekonstrukce pohybu z dat senzorů

Senzory poskytují nezpracovaná nefiltrovaná data. Tato samotná data nemají silnou informační hodnotu a je potřeba je upravit před jejich prezentací. Tato sekce vychází z [9].

#### Akcelerometr

Akcelerometr poskytuje data popisující zrychlení ve třech hlavních osách, ovšem nejedná se pouze o lineární zrychlení ve směru dané osy, ale měřené zrychlení se skládá z více různých složek. První složka je samotné lineární zrychlení, dále může hodnota obsahovat složku tvořenou rotačním zrychlením v případě, že daný senzor neleží přesně ve středu otáčení. Další důležitou složkou je gravitační zrychlení, které nám pomůže aproximovat orientaci v prostoru. Poslední složkou je šum. Jelikož nelze vytvořit dokonalý senzor, šum bude ve větší či menší míře obsažen v každém měření.

Hlavní údaj o orientaci objektu, který nám akcelerometr poskytuje, se nazývá inklinace. Inklinace popisuje úhel naklonění souřadnicového systému tělesa vůči globálnímu souřadnicovému systému podle osy $Z$. Samotná hodnota inklinace není ovšem pro popis celkové orientace objektu dostačující, protože popisuje pouze úhel náklonu a ne jeho směr. Z tohoto důvodu můžeme odvodit pouze otočení vůči osám $X$ a $Y$. Za předpokladu, že je akcelerometr v klidu a velikost šumu je zanedbatelná, může být zrychlení ve směru jednotlivých os určeno vztahy:

$$a_x = g\sin(\theta) \tag{2.7}$$

$$a_y = -g\cos(\theta)\sin(\phi) \tag{2.8}$$

$$a_z = -g\cos(\theta)\cos(\phi) \tag{2.9}$$

kde $a_x$,$a_y$ a $a_z$ jsou zrychlení podle jednotlivých os souřadnicového systému objektu v jednotkách $ms^{-2}$, $g$ je gravitační zrychlení, $\theta$ je úhel otočení kolem osy $Y$ a $\phi$ je úhel otočení kolem osy $X$.

Pro výpočet úhlů otočení kolem os $X$, $Y$ z údajů z akcelerometru stačí pouze tuto soustavu rovnic transformovat do jiného tvaru:

$$\phi = \arctan\left(\frac{a_y}{a_z}\right) \tag{2.10}$$

$$\theta = \arcsin\left(\frac{a_x}{g}\right) \tag{2.11}$$

### Gyroskop

Data gyroskopu poskytují informace o velikosti rychlosti otáčení, tedy úhlové rychlosti, kolem určité osy. Stejně jako v případě akcelerometru nejsou data gyroskopu ideální, protože jejich součástí je datový šum a také s časem se měnící zkreslení [11]. Výpočet orientace může být uskutečněn integrováním naměřených dat od 0 do jedné periody měření $T$. Takto vypočteme změnu úhlů za jednu periodu. Pokud tuto změnu přičteme k předešlé orientaci, dostaneme orientaci aktuální.

$$\Delta\phi = \int_0^T \omega_x dt \tag{2.12}$$

$$\Delta\theta = \int_0^T \omega_y dt \tag{2.13}$$

$$\Delta\psi = \int_0^T \omega_z dt \tag{2.14}$$

Tento způsob výpočtu s sebou nese ovšem pár problémů. První problém je chybějící počáteční orientace, kterou nejsme schopni získat. Řešením může být například manuální nastavení počáteční orientace, nebo lépe použití orientace z akcelerometru.

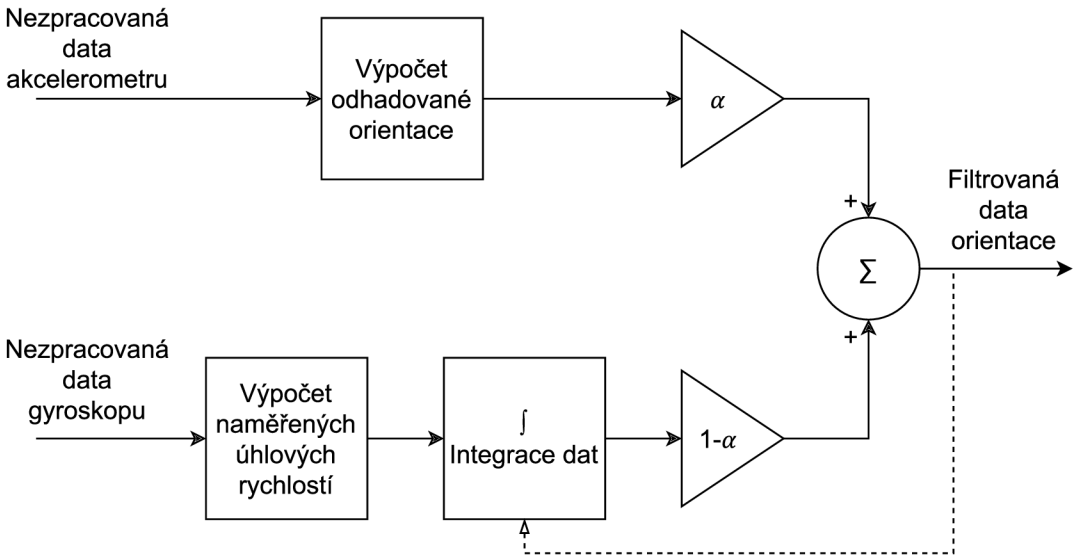
Dalším problémem je, že součástí integrované úhlové rychlosti je s časem se měnící zkreslení, které postupně více a více znepřesňuje naměřenou orientaci. Jedním řešením by mohlo být odečtení hodnoty zkreslení naměřené při kalibraci senzoru od každé hodnoty naměřené gyroskopem. Lepším případem je spojení tohoto řešení s použitím některého z fúzních filtrů.

### Komplementární filtr

Oba senzory, akcelerometr i gyroskop, mají určité nevýhody. V případě gyroskopu je to například zmíněná neschopnost získat počáteční polohu, nebo zvětšující se chyba měření z důvodu integrace dat. V případě akcelerometru například nemožnost výpočtu inklinace při nerovnoměrném či nepřímočarém pohybu. Pokud by byla konečná data výsledkem propojení dat z obou senzorů, mohly by být tyto nevýhody do značné míry potlačeny. Jednou z možností spojování dat z různých senzorů je právě komplementární filtr [34].

Komplementární filtr pochází z teorie řízení a umožní nám fúzi dat z více senzorů s malým nárokem na výpočetní výkon.

Ze schématu 2.6 je možné vidět způsob fúze dat z akcelerometru a gyroskopu. Na vstupu filtru jsou nezpracovaná data těchto senzorů. Blok `Výpočet odhadované orientace` znázorňuje přepočet dat akcelerometru na zrychlení a výpočet orientace popsaný v sekci 2.1.3. Blok `Výpočet naměřených úhlových rychlostí` představuje přepočet nezpracovaných dat gyroskopu na úhlové rychlosti. Blok `Integrace dat` představuje výpočet orientace popsaný v sekci 2.1.3. Dále je každá větev filtru vynásobena konstantou, kde součet konstant




Obrázek 2.6: Schéma komplementárního filtru pro akcelerometr a gyroskop.

obou větví je 1. Tyto konstanty popisují důležitost jednotlivých dat senzorů v celkovém výpočtu. Poslední blok provádí sečtení hodnot orientací obou větví a výsledkem této operace jsou konečná filtrovaná data [29, 34].

Tento základní filtr je sice pro výpočet nenáročný, ovšem jeho filtrační schopnosti nejsou převratné. Ve stavu, kdy se pozice senzorů nemění nárazově, je filtr účinný a efektivní. Při komplexním pohybu senzorů ovšem tento filtr nemusí dostačovat, což se může projevit například velmi vysokou odchylkou naměřených hodnot od hodnot reálných. V případě, že filtrační schopnosti tohoto filtru nejsou dostačující, je potřeba zvolit jiný komplexnější filtr, například Kálmánův filtr [22], ovšem za cenu vyššího výkonu [27].

## 2.2 MEMS

Mikroelektromechanické systémy (MEMS) jsou malé zařízení vyrobené pomocí mikrofabrikačních technik. MEMS senzor je typ senzoru, který používá mikroelektromechanickou technologii k detekci a měření změn v okolním prostředí.

MEMS senzory existují v mnoha různých typech, včetně akcelerometrů, gyroskopů, tlakových senzorů, teplotních senzorů a magnetických senzorů, mezi jinými. Tyto senzory jsou často používány v různých aplikacích jako jsou chytré telefony, nositelná zařízení, automobilové systémy a lékařské přístroje.

Tyto senzory jsou složeny z komponent o velikosti mezi 1 a 100 mikrometrů, samotné MEMS zařízení má běžně velikost od 20 mikrometrů do 1 milimetru. Takové zařízení běžně obsahuje centrální jednotku, která zpracovává naměřené hodnoty, a několik MEMS komponent, které interagují s okolním světem [17].

### 2.2.1 Materiály používané pro výrobu MEMS senzorů

Důležitý aspektem tvorbu MEMS senzorů je správná volba materiálů, která ovlivní výsledné vlastnosti daného zařízení. Mezi nejpoužívanější materiály patří například křemík–karbid, kovové slitiny, jako je například zlato–nikl, diamant, polymerní materiály, nebo sklo [38, 15].

**Křemík–karbid** Křemík–karbid (SiC) poskytuje vysokou odolnost proti teplotám, korozivním látkám a mechanickému opotřebení. Senzory z tohoto materiálu jsou často používány v náročných průmyslových aplikacích.

**Slitina zlato–nikl** Slitinu zlato–nikl lze použít pro výrobu MEMS senzorů s vysokou pevností a nízkou elektrickou rezistencí. Senzory z tohoto materiálu jsou často používány v elektronických zařízeních s vysokou hustotou výkonu.

**Diamant** Diamantové MEMS senzory se používají převážně v náročných aplikacích jako jsou výbušniny, kosmické aplikace a průmyslové výrobní procesy. Diamantové senzory mají vynikající vlastnosti jako je vysoká tvrdost, chemická inertnost a vynikající tepelná vodivost.

**Polymerní materiály** Určité polymerní materiály lze použít pro výrobu MEMS senzorů s vysokou flexibilitou a nízkou hmotností. Takovéto senzory jsou hojně využívány v medicínských aplikacích a v zařízeních pro sledování pohybu.

**Sklo** Sklo dodává MEMS senzorům vysokou chemickou a tepelnou stabilitu a vynikající optickou transparentnost. Takový materiál je vhodný pro použití v mikrofluidních aplikacích a v senzorech pro sledování pohybu.

### 2.2.2 Výrobní procesy

Procesy využívané pro tvorbu MEMS senzorů představují kombinaci klasických procesů pro tvorbu integrovaných obvodů a jiných sofistikovaných procesů, které využívají vlastností použitých materiálu pro dosažení mechanických vlastností. Nejvyužívanější výrobní procesy pro MEMS senzory jsou například: *Optická litografie*, *Mokré leptání*, *Hloubkové reaktivní iontové leptání* nebo *LIGA* (německý akronym pro „Lithographie, Galvanoformung, Abformung“ – litografie, galvanické pokovování a lisování.) [18, 25].

**Optická litografie** Optická litografie se používá při výrobě MEMS senzorů k vytvoření přesných obrazů struktur na substrátu. Proces začíná nanesením vrstvy fotorezistu na substrát, která se následně vystaví UV světlu skrze masku. Světlo způsobí rozložení fotorezistu a umožní odstranění materiálu podle vzoru masky. Tento proces umožňuje velmi přesnou tvorbu struktur na substrátu a je jedním z nejčastěji používaných procesů pro výrobu MEMS zařízení.

**Mokré leptání** Mokré leptání je chemický proces, který se používá při výrobě MEMS senzorů k odstraňování materiálu z povrchu substrátu pomocí kyselinového roztoku. Tento proces umožňuje vytvoření přesných a opakujících se struktur na substrátu, ale vyžaduje správné nastavení času a koncentrace roztoku pro dosažení požadovaného tvaru a hloubky struktury.

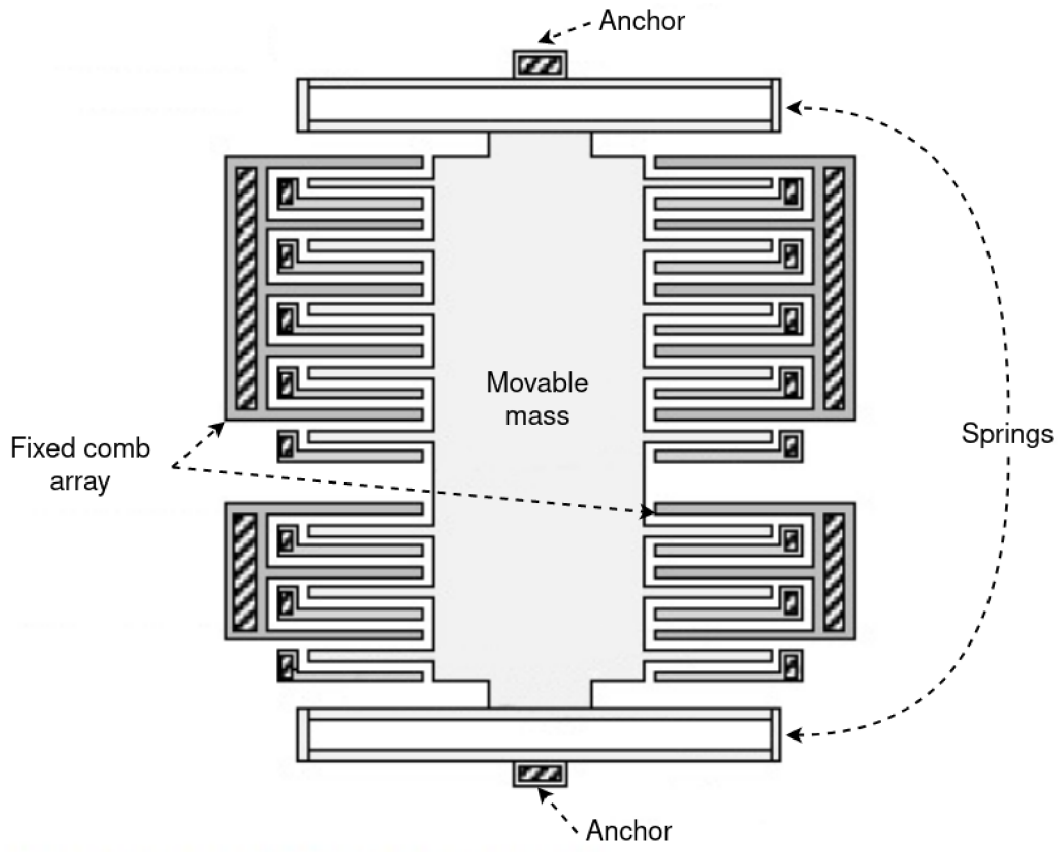
**Hloubkové reaktivní iontové leptání** Proces je odvozen od optické litografie, kde je substrát po vytvoření masky s pomocí fotorezistu umístěn do vysokofrekvenčního elektromagnetického pole, kde se ionty srazí s povrchem substrátu, čímž jsou nemaskované části substrátu odstraněny.

**LIGA** LIGA je proces, který využívá rentgenovou litografii, galvanické pokovování a formování. Tento proces umožňuje vytváření velmi přesných a opakovatelných struktur na substrátu, což je ideální pro výrobu mikrofluidních zařízení.

### 2.2.3 MEMS akcelerometr a gyroskop

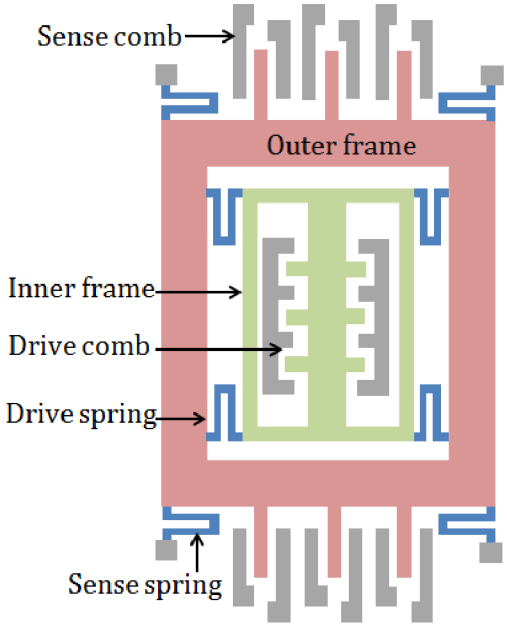
Oba tyto senzory využívají pro měření výchozích hodnot změnu kapacity mezi částmi MEMS komponenty.

Na obrázku 2.7 je vidět schéma MEMS akcelerometru, který sestává ze dvou hlavních částí. Fixní část je napevno připevněna k substrátu a druhá pohyblivá část (Movable Mass) je přes pružné části (Springs) připevněna ke kotevním bodům (Anchor). V případě, že se tato komponenta bude hýbat se zrychlením v daném směru, bude na ni působit síla, která způsobí ohyb pružných částí a posun pohyblivé části. Změna polohy pohyblivé části způsobí změnu kapacity mezi touto a fixní částí, kde změna této kapacity je závislá na síle působící na tuto komponentu, a tedy i závislá na zrychlení, se kterým se tato komponenta pohybuje. Samotná komponenta je schopna měřit zrychlení pouze v jednom směru, proto výsledný akcelerometr většinou obsahuje více těchto komponent [19].




Obrázek 2.7: Schéma MEMS akcelerometru [41].

Na obrázku 2.8 je zobrazeno schéma komponenty MEMS gyroskopu. Na první pohled je zřejmé, že konstrukce MEMS gyroskopu je mírně složitější, než tomu bylo u akcelerometru. Tento MEMS gyroskop využívá Koriolisova efektu, který popisuje vznik Koriolisovy síly. Jedná se o setrvačnou sílu působící na tělesa, která se pohybují v rotující neinerciální vztažné soustavě [44]. MEMS gyroskop měří tuto sílu obdobně jako akcelerometr. Stejně jako u akcelerometru, gyroskop často obsahuje více těchto komponent pro měření úhlové rychlosti kolem více os [19].




Obrázek 2.8: Schéma MEMS gyroskopu [31].

## 2.3 Bluetooth Low Energy (BLE)

Bluetooth Low Energy je bezdrátový komunikační protokol, který jak vyplývá z názvu, je zaměřen na nízkou spotřebu energie. Díky této skutečnosti je vhodný pro zařízení napájená malými bateriemi, které potřebují pracovat po delší časovou periodu. Jedná se o rozšíření standartu Bluetooth, které bylo představeno s verzí Bluetooth 4.0 [39].

BLE operuje na frekvenci 2.4 GHz s teoretickým dosahem až 100 metrů, ovšem reálná vzdálenost je menší z důvodu velkého množství faktorů jako například překážky, přes které signál prochází [10].

### 2.3.1 Protokol stack

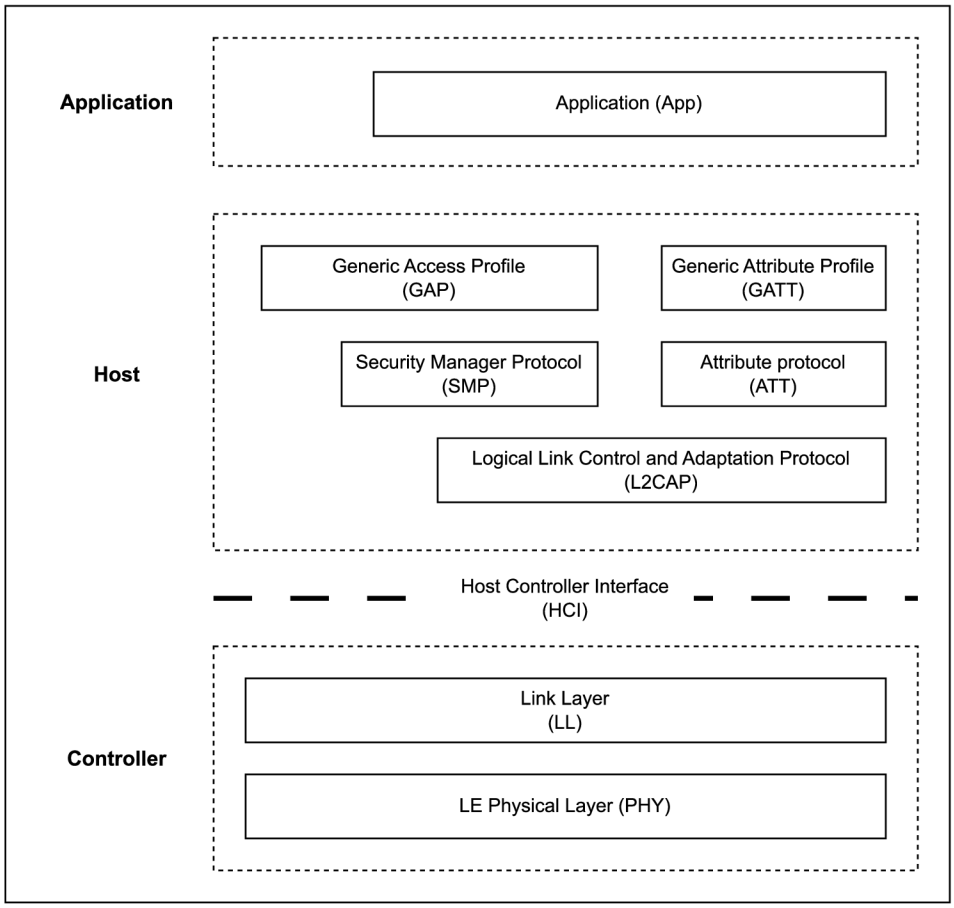
Bluetooth Low Energy protocol se skládá z několika vrstev, které pracují dohromady a zajišťují komunikační vlastnosti toho protokolu.

Na nejvyšší úrovni je protokol rozdělen na tři části, *Application*, *Host* a *Controller*, které se dále větví do menších logických celků. *Application* část představuje uživatelovu aplikaci, která zpracovává potřebná přenesená data. *Host* část představuje skupinu protokolů, běžně implementovaných v rámci operačních systémů, které implementují standardy BLE. A *Controller* část běžně reprezentuje hardwarový BLE modul [39].

Následující sekce blíže rozebírají jednotlivé vrstvy BLE protokolu znázorněné na diagramu na obrázku 2.9.

#### Physical Layer (PHY)

Jedná se o fyzickou vrstvu tohoto protokolu, která zodpovídá za odesílání a přijímání rádiových signálů mezi jednotlivými BLE zařízeními. Tato vrstva je schopna modulovat a demo-




Obrázek 2.9: BLE protokol stack.

dulovat zmíněný radiový analogový signál a transformovat ho do potřebné digitální podoby [39].

BLE radiové signály využívají 2.4 GHz IMS[3] frekvenční pásmo. Toto frekvenční pásmo je rozděleno do 40 kanálů na frekvencích 2.4000 GHz až 2.4835 GHz. Z toho 37 kanálů je používáno na přenos dat a zbylé 3 kanály jsou využívány jako *Advertising* kanály pro přenos informací o daných zařízeních [10].

### Link Layer (LL)

*Link layer* je běžně implementovaná jako kombinace hardwaru a softwaru. Jedná se o jedinou real–time vrstvu celého BLE protokolu. Také z tohoto důvodu bývá společně s PHY vrstvou běžně izolovaná od ostatních vrstev BLE protokolu pomocí speciálního rozhraní zvaného HCI, popsaného v následující sekci [39].

### Host Controller Interface (HCI)

HCI je rozhraní sloužící pro oddělení *Controller* části BLE protokolu, který je zaměřen primárně na real–time operace, od ostatních částí tohoto protokolu. Bluetooth specifikace definuje HCI jako soubor příkazů a událostí, které umožňují interakci zmíněných částí BLE

[3]Industrial, Scientific, and Medical. Viz https://en.wikipedia.org/wiki/ISM_radio_band.

protokolu. Dále jsou také součástí tohoto rozhraní definovány správné formáty datových paketů a soubory pravidel pro kontrolu zařízení [39].

### Logical Link Control and Adaptation Protocol (L2CAP)

Tato vrstva má za úkol dvě hlavní úlohy. Za prvé slouží jako multiplexor, který zapouzdřuje protokoly vyšších vrstev do standardních BLE paketů a také naopak příchozí standardní BLE pakety rozděluje jednotlivým odpovídajícím protokolům vyšších vrstev.

Druhým úkolem této vrstvy je fragmentace a rekombinace paketů. Odchozí pakety vyšších vrstev jsou rozděleny do celků o maximální velikosti 27 bajtů, které mohou být následně předány nižším vrstvám. A naopak příchozí pakety jsou spojovány do odpovídajících celků a následně předány vyšším vrstvám [10].

Chování této vrstvy je obdobné jako chování internetového protokolu TCP definovaného v RFC 973 [32, 39].

### Attribute Protocol (ATT)

ATT vrstva reprezentuje bezstavový klient–server protokol na základě atributů daného zařízení.

Každý server obsahuje data organizovaná do atributů, kde jeho každý atribut je reprezentován jedinečným 16–bitovým identifikátorem (universally unique identifier – UUID), souborem oprávnění a také samotnou hodnotou.

V případě, že si klient přeje pracovat s daty na serveru, vyšle serveru *read* nebo *write* příkaz. Následně server odpoví zprávou s vyžádanými daty v případě *read* příkazu nebo potvrzující (ACK) zprávou v případě *write* příkazu. V obou případech je na klientovi, aby porozuměl a ohlídal datové typy zpráv [39].

### Security Manager (SM)

SM vrstva poskytuje zabezpečovací služby BLE protokolu. Zajišťuje autentizaci a šifrování dat přenášených mezi účastníky komunikace. Dále také obsahuje bezpečnostní algoritmy pro předávání klíčů a jejich správu [39].

### Generic Attribute Profile (GATT)

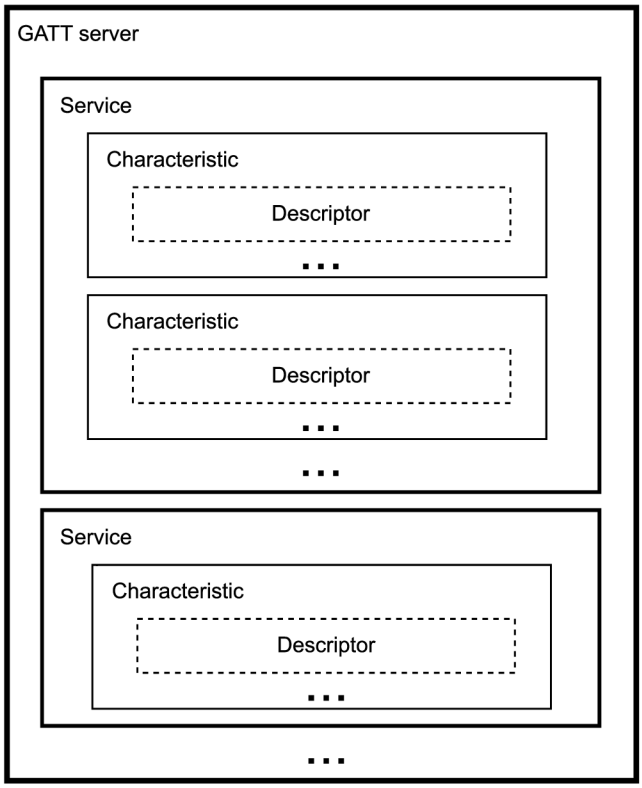
Jedná se o nástavbu nad již popsanou ATT vrstvou, kterou tato vrstva využívá jako transportní. Dále k ní přidává hierarchii dat a abstraktní datový model. Je to jedna z nejdůležitějších vrstev BLE protokolu, jelikož definuje, jak jsou data organizována a také předávána mezi aplikacemi.

GATT definuje stejně jako ATT dvě role, klient a server. Tyto role odpovídají rolím definovaným v sekci popisující ATT vrstvu. Účel klienta je získávat, případně zapisovat, hodnoty atributů na serveru, ovšem nejprve je nutné provést sken serveru, aby klient věděl, jakými atributy a s jakými oprávněními server disponuje. Server odpovídá na příkazy klienta a poskytuje data jednotlivých atributů. Tyto role jsou zcela nezávislé na rolích GAP vrstvy popsané v následující sekci.

Jak už bylo naznačeno dříve, atributy jsou jednou z nejdůležitějších částí GATT vrstvy. Jedná se o nejmenší adresovatelnou datovou strukturu definovanou na této vrstvě. Každý atribut se povinně skládá z pěti částí, Identifikátor (*Handle*), Typ (*Type*), Oprávnění (*Permissions*), Hodnota (*Value*) a Délka hodnoty (*Value length*). Identifikátor, jak vyplývá z názvu, jedinečně identifikuje atribut v rámci jednoho GATT serveru. Jedná se o 16 bitovou

hodnotu, která se atributům přiřazuje vzestupně, ovšem inkrement následujícího identifikátoru nemusí být 1. Typ atributu definuje, jaký typ dat daný atribut může obsahovat. Jeho hodnota je univerzálně jedinečný identifikátor o velikosti 128 bitů. Jelikož ale 16 bajtů zabírá značnou část 27 bajtového BLE paketu probraného v dřívější sekci, BLE specifikace definuje také 16 a 32 bitové identifikátory pro nejpoužívanější typy atributů. Oprávnění atributu jsou metadata, které popisují jaké operace je možné s daným atributem provádět a s jakými bezpečnostními požadavky. Hodnota atributu představuje datový obsah tohoto atributu, BLE specifikace nijak neomezuje jaký typ dat může být uložen v této části, ovšem omezuje jejich délku, a to na 512 bajtů. A poslední část atributu, délka hodnoty, označuje reálnou délku datového obsahu tohoto atributu [10].

Tato vrstva také definuje striktní hierarchii těchto atributů, které jsou rozděleny do služeb (*Services*), charakteristik (*Characteristics*) a popisovačů (*Descriptors*), viz digram na obrázku 2.10.




Obrázek 2.10: Datová hierarchie GATT vrstvy.

Služby (*Services*) jsou kolekce jedné a více příbuzných charakteristik, které definují určitou funkcionalitu. *Services* jsou označovány univerzálně jedinečnými identifikátory o velikost 126 bitů, případně 16 nebo 32 bitů, pokud jsou definované v BLE specifikaci.

Charakteristika je skupina atributů představující jednu uživatelskou hodnotu. Popis této hodnoty, jako vlastnosti, datový typ, nebo umístění hodnoty, je uložen v charakteristiku deklarujícím atributu. Možné vlastnosti GATT charakteristiky jsou:

**Broadcast** Umožňuje dané charakteristice být součástí *Advertising* paketů

**Read** Umožňuje GATT klientům číst hodnotu dané charakteristiky.

**Write without response** Umožňuje GATT klientům zapisovat hodnotu do dané charakteristiky bez očekávání odpovědi.

**Write** Umožňuje GATT klientům zapisovat hodnotu do dané charakteristiky.

**Notify** Umožňuje GATT serveru asynchronně rozesílat nové hodnoty dané charakteristiky bez očekávání potvrzení ze strany GATT klienta.

**Indicate** Umožňuje GATT serveru asynchronně rozesílat nové hodnoty dané charakteristiky s očekáváním potvrzení ze strany GATT klienta.

Uživatelská data dané charakteristiky jsou uložena v samotném atributu. Identifikátor tohoto atributu lze získat právě z charakteristiky deklarujícího atributu.

Popisovače (*Descriptors*) poskytují dodatečná data o určité charakteristice či službě. Jsou složena z jednoho atributu, který je vždy umístěn v rámci definice dané charakteristiky či služby. Popisovače mohou poskytovat například uživatelsky čitelný název charakteristiky, nebo strukturu dat dané charakteristiky a mnoho dalších údajů [39].

### Generic Access Profile (GAP)

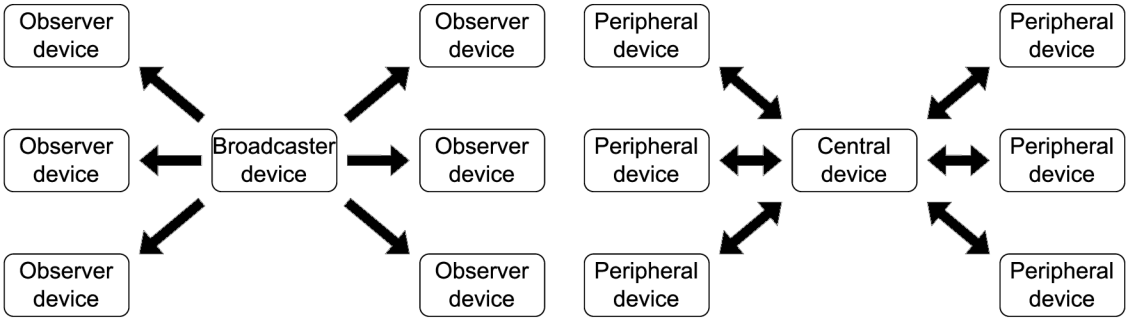
Tato vrstva BLE protokolu definuje jak mezi sebou jednotlivá BLE zařízení interagují při připojovací fázi. Definuje, jakým způsobem se jednotlivá zařízení mezi sebou hledají, navazují spojení a mnoho jiných standardních aktivit.

Tato vrstva specifikuje čtyři role zařízení, *Broadcaster*, *Observer*, *Central* a *Peripheral*. Jak již bylo řečeno dříve, tyto role jsou zcela nezávislé na rolích definovaných GATT vrstvou. Každé zařízení může nabývat jedné nebo i více GAP rolí současně. *Broadcaster* je role, která pouze vysílá data v pravidelných intervalech. Data jsou odesílána v *Advertising* paketech a okolní zařízení mohou tedy tato data přijímat bez nutnosti navázání spojení s daným zařízením. *Observer* role je optimalizovaná pouze naslouchání. Veškerá data, která tato role může přijímat, musí být součástí *Advertising* paketů. Je tak ideální pro přijímání dat z několika *Broadcasterů* současně. *Central* role je schopna navazovat spojení s jinými zařízeními, která najde pomocí jejich *Advertising* paketů. Právě tato role vždy navazuje spojení mezi dvěmi zařízeními. *Peripheral* role vysílá *Advertising* pakety, aby informovala okolní zařízení o její existenci a umožnila s nimi navázat spojení. BLE protokol má asymetrický charakter a z tohoto důvodu vyžaduje role *Central* více výpočetního výkonu než role *Peripheral* [10].

Od těchto rolí lze také odvodit topologie BLE sítí, viz obrázek 2.11. Jelikož ovšem jedno zařízení může nabývat více rolí najednou, tyto topologie mohou být navzájem kombinovány a proplétány.

Na obrázku 2.11a je vidět topologie v případě použití zařízení pouze s rolemi *Broadcaster* a *Observer*. Lze vidět, že data vysílaná z jednoho *Broadcasteru* mohou být čtena více *Observery*. Důležitá je také jednosměrnost komunikace v této topologii. Pkud by chtěl *Observer* také rozesílat data, musel by se navíc stát on sám *Broadcasterem* a naopak.

Dále je na obrázku 2.11b vidět topologie složená z *Central* a *Peripheral* rolí. Oproti předchozí topologii umožňuje tato topologie obousměrný přenos dat, ovšem pro přenos dat musí být daná zařízení k sobě připojena a déle přenos dat probíhá pouze mezi dvěma zařízeními najednou.

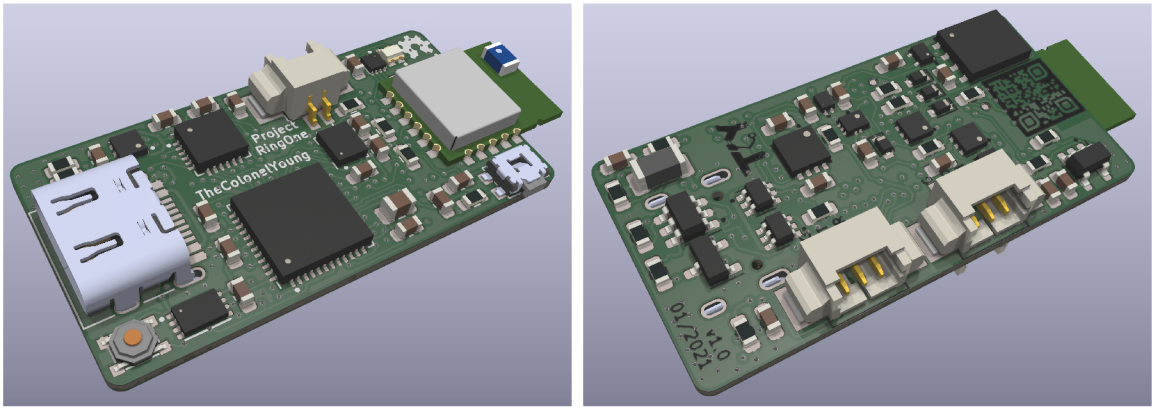



(a) Broadcast topologie (b) Central topologie

Obrázek 2.11: BLE topologie.

## 2.4 Popis zařízení sloužící pro sběr prostorových dat

Tato práce navazuje na již existující hardwarové řešení RingOne [26]. Jedná se o zařízení určené pro tvorbu 3D myši. Jeho velikost je přibližně $4 \times 2$ centimetry. Na obrázku 2.12 je vyobrazen render tohoto zařízení a jsou zde vidět všechny jeho komponenty. RingOne je navržen primárně pro práci v bezdrátovém režimu a obsahuje všechny k tomu potřebné komponenty popsané v následující sekci.




(a) Pohled shora (b) Pohled zespoda

Obrázek 2.12: Render zařízení RingOne, převzato z internetového zdroje [26].

### 2.4.1 Komponenty zařízení RingOne

Tato sekce popisuje nejdůležitější použité komponenty. Celkové schéma zařízení RingOne je k nalezení v příloze A.

#### Mikrokontrolér

Procesor je hlavní výpočetní jednotka celého zařízení, tento mikrokontrolér konkrétně je vyráběn firmou STMicroelectric, přesněji jde o verzi `STM32L452CEU6` [7]. Jedná se o mikrokontrolér naložený na 32-bitovém jádru Arm Cortex-M4 a patří do řady STM32L4 zmíněné firmy STMicroelectronics. Tato řada je optimalizována pro nízkou spotřebu energie při

zachování vysokého výkonu, což ho činí ideálním pro širokou škálu bateriově napájených aplikací.

STM32L452CEU6 nabízí operační frekvenci až 80 MHz, zabudovanou paměť Flash o velikosti 512 KB a SRAM paměť o velikosti 160 KB. Je také vybaven matematickým koprocesorem (floating-point unit – FPU), což umožňuje efektivnější zpracování dat s plovoucí desetinnou čárkou. Konkrétně se jedná o hardwarově implementovanou jednotku FPU s podporou pro datový typ IEEE 754 float a double. Dále disponuje různými komunikačními rozhraními, jako jsou USB, SPI, $I^2C$ a USART, což mu umožňuje snadno interagovat s jinými zařízeními.

STM32L452CEU6 je také vybaven funkcí Direct Memory Access (DMA), jež umožňuje rychlý a efektivní přenos dat mezi periferiemi a pamětí bez zatížení jádra procesoru. DMA je užitečné zejména pro aplikace s vysokou propustností dat, jako jsou přenosy obrazových nebo zvukových dat.

Procesor také podporuje přerušení, což je mechanismus, který umožňuje mikrokontroléru reagovat na události v reálném čase. Přerušení mohou být vyvolána různými periferiemi nebo událostmi a procesor pak okamžitě přepne své zdroje na obsluhu přerušení. Tato funkce umožňuje snadnou a efektivní reakci na změny v systému.

### IMU

Inerciální měřící jednotka (Inertial measuring unit – IMU) je velmi důležitá komponenta, jelikož je zdrojem všech prostorových dat. RingOne používá komponentu IMU s označením `LSM6DSL` [5], tato jednotka obsahuje akcelerometr i gyroskop a je navržena pro použití v mobilních zařízeních, přenosných počítačích, fitness monitorech a dalších podobných aplikacích.

LSM6DSL poskytuje přesné měření 3D akcelerace a úhlové rychlosti, což umožňuje sledovat pohyb zařízení a získávat informace o jeho orientaci v prostoru. Tento senzor je vybaven pokročilými funkcemi, jako je například detekce kroků, detekce pohybu, detekce pádu a detekce klesnutí.

Tento senzor je také velmi úsporný a nabízí nízkou spotřebu energie, což umožňuje jeho použití v bateriemi napájených zařízeních. Navíc tato komponenta podporuje různé komunikační rozhraní, jako je například $I^2C$, SPI a rozhraní s jedním drátem, což zjednodušuje jeho integraci do různých typů zařízení.

Tato komponenta dále umožňuje vyvolávat přerušení procesoru za určitých podmínek. Jednou z těchto podmínek může být například poklepání prstu na zařízení nebo zachycení volného pádu. Možné je použít i implicitní filtrování dat pomocí horních a dolních propustí pro omezení šumu v datech [14].

### Bluetooth modul

Přesněji Bluetooth Low Energy modul použitý v tomto zařízení nese označení RN4871 [6]. Tento modul vyrobený společností Microchip Technology je navržený pro snadnou integraci do různých typů zařízení, jako jsou například senzory, přístroje pro sběr dat, zařízení pro sledování polohy a mnoho dalších.

Modul má malé rozměry (pouze $9 \times 11,5 \times 2$ mm) a je založen na čipu CSR1010 firmy Qualcomm. Podporuje Bluetooth 4.2 a je kompatibilní s různými operačními systémy včetně iOS a Android.

RN4871 je vybaven řadou funkcí jako jsou například integrovaný anténní filtr, plně programovatelný firmware, podpora rozhraní UART pro komunikaci s externími zařízeními

a možnost konfigurace přes sériovou linku nebo bezdrátově pomocí aplikace v mobilním zařízení.

Jelikož modul implementuje BLE, má chování klasického BLE zařízení, tedy obsahuje GATT služby a jejich charakteristiky. Výchozí služba, kterou tento modul poskytuje, nese název *Transparent UART Service*. Tato služba vytváří abstraktní vrstvu nad samotným BLE protokolem a umožňuje propojeným zařízením komunikovat přes protokol UART [2], téměř jako by byly propojeny vodiči. Tato GATT služba dále obsahuje dvě GATT charakteristiky, kde každá z nich reprezentuje jeden směr toku dat. První charakteristika je pojmenovaná *Transparent UART TX*. Tato charakteristika reprezentuje přenos dat z GATT serveru (tento modul) do GATT klienta (připojené zařízení) a přenos je realizován využitím vlastnosti GATT charakteristiky `Notify`, viz 2.3.1. Druhá charakteristika s názvem *Transparent UART RX* naopak reprezentuje přenos dat ze strany GATT klienta na GATT server. Tato charakteristika realizuje přenos dat s pomocí vlastnosti GATT charakteristiky `Write` a `Write without response`. Následující tabulka 2.1 ukazuje identifikátory (UUIDs) zmíněné služby a charakteristik i s jejich GATT vlastnostmi.

| **Název služby/ charakteristiky** | **UUID** | **GATT vlastnosti** |
|---|---|---|
| Transparent UART Service | `49535343-FE7D-4AE5-8FA9-9FAFD205E455` | |
| Transparent UART TX | `49535343-1E4D-4BD9-BA61-23C647249616` | Notify, Write, Write without response |
| Transparent UART RX | `49535343-8841-43F4-A8D4-ECBE34729BB3` | Write, Write without response |

Tabulka 2.1: Popis GATT služby a jejích charakteristik.

### Signalizační dioda

RingOne také disponuje RGB LED diodou, kterou je možné použít například pro signalizaci stavu zařízení. Tuto diodu ovšem neřídí procesor přímo, ale o její ovládání se stará budič LED diod s označením `KTD2026BEWE` [4]. Tento budič komunikuje s procesorem pomocí protokolu $I^2C$ a dokáže nastavit vlastnosti LED diody podle potřeby. Možné je nastavení velikosti elektrického proudu jdoucího do jednotlivých kanálů RGB diody, díky tomu je možné nastavit specifickou barvu a jas připojené diody. Tato komponenta také umožňuje jednotlivé kanály připojit k některému ze dvou interních časovačů, pomocí kterých je možné upravit frekvenci blikání, či dobu zhasínání nebo rozsvěcení. Kombinace všech těchto nastavení může vést k vytvoření zajímavých a užitečných efektů.

### EEPROM paměť

EEPROM paměť je nevolatilní typ paměti, což znamená, že zapsaná data zůstanou v dané paměti i po odpojení napájení. Tato skutečnost dělá z EEPROM paměti ideální místo pro ukládání například konfigurace zařízení nebo jiných hodnot, které mají životnost delší než samotný běh aplikace.

RingOne využívá paměťový modul s označením M24C32-FMC6TG [3]. Tento modul disponuje kapacitou 32 KB a komunikace je realizována pomocí protokolu $I^2C$.

### 2.4.2 Napájení

Všechna elektrická zařízení potřebují energii. A jinak tomu není ani u zařízení RingOne, které je napájeno primárně z portu USB-C. Dále je možné pomocí připraveného konektoru připojit jednočlánkovou li-po baterii, která umožní používat zařízení opravdu bezdrátově. O tuto baterii se starají obvody popsané ve třetí části schématu v příloze A. Dále RingOne obsahuje skupinu obvodů starající se o celkové napájení, které jsou popsány ve čtvrté části schématu v příloze A. Tyto obvody umožňují také například vypnout napájení některých periférií a ponechat napájení procesoru pro úsporu energie.

V případě, že se procesor nachází v módu nízké spotřeby, je možné ho vzbudit například zmáčknutím tlačítka nacházejícím se u Bluetooth modulu, nebo poklepáním na samotné zařízení pokud jsou IMU i procesor nastaveny na vytváření přerušení v případě zachycení tohoto poklepání.

Zařízení RingOne také v určitých případech umožňuje *MCU offloading*, což je proces, při kterém se určité úkoly procesory přenáší na jiné komponenty. Toto má za následek ušetření cenného procesorového času, ale také například ušetření energie, jelikož komponenty, na které se dané úkoly přenesou, mohou být daleko více optimalizovány pro tyto úkoly. Příkladem pro *MCU offloading* pro toto zařízení může být například zmíněný LED driver, který oprostí procesor od složitého ovládání RGB LED diody, nebo také samotný procesor umožňuje použití dříve zmíněného DMA, které umožní jiným komponentám zapisovat přímo do určené paměti bez nutnosti použití výpočetního jádra.

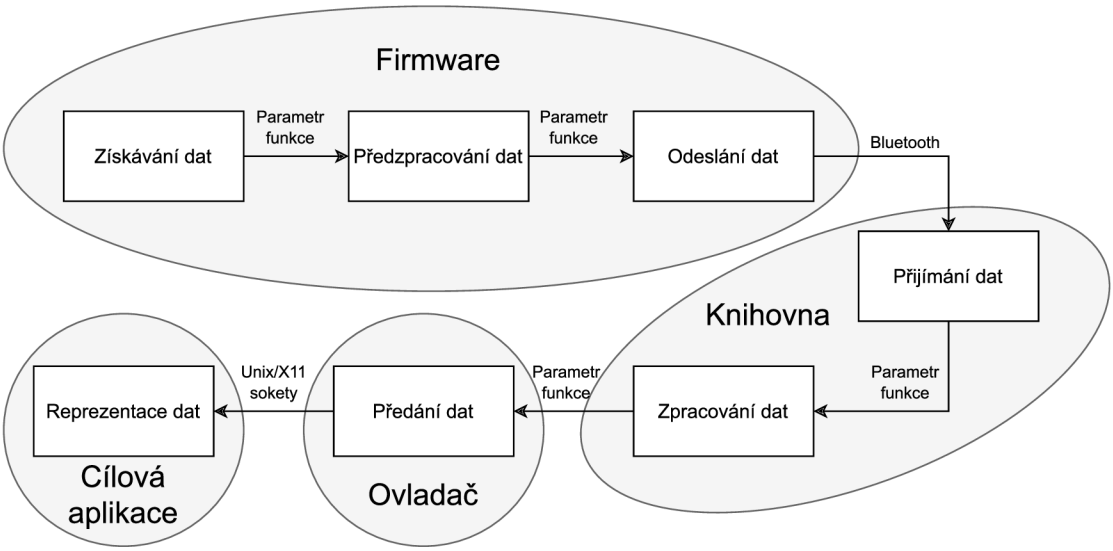
# Kapitola 3

# Návrh řešení rekonstrukce pohybu

Tato kapitola popisuje návrh systému pro získávaní a zpracování prostorových dat. Získávání těchto dat je zprostředkováno již zmíněným zařízením RingOne. Pro širší zpracování dat je navržena knihovna, která navíc bude poskytovat jednoduchou komunikaci se zdrojovým zařízením.

## 3.1 Životní cyklus dat systému

Na obrázku 3.1 je zobrazen diagram popisující životní cyklus dat celkového navrhovaného systému. Digram popisuje data od jejich vzniku až po konečnou reprezentaci.




Obrázek 3.1: Životní cyklus dat.

Z diagramu je patrné rozdělení na čtyři hlavní části, `Firmware`, `Knihovna`, `Ovladač` a `Cílová aplikace`. Tyto celky naznačují, v jaké části implementace budou jednotlivé body realizovány. `Firmware` představuje program na již představeném zařízení RingOne a je nejdůležitější částí této práce. `Knihovna` označuje softwarovou knihovnu jazyka C, podrobněji bude probrána v sekci 3.3. Celek `Ovladač` popisuje část implementace, která bude umožňovat posílat data do různých 3D aplikací na úrovni uživatelského ovladače. Poslední částí je `Cílová aplikace`, která představuje klientskou aplikaci, která získaná

data zobrazuje. Pomocí již zmíněného ovladače bude jednoduše možné zobrazovat data v CAD a jiných aplikacích. Dále také bude možné s využitím knihovny získávat data v jiném programu přímo a tedy obejít cestu ovladače.

Šipky v digramu naznačují směr postupu dat. Tyto šipky také obsahují popisek, který naznačuje, jakým způsobem jsou data přenášena do další komponenty. `Parametr funkce` značí, že jednotlivá data jsou předávána jako parametr určité funkce, která reprezentuje následující komponentu, nebo alespoň její část. `Bluetooth` popisek naznačuje, že data jsou do další komponenty předávána přes Bluetooth, přesněji se jedná o jeho rozšíření BLE, popsané v sekci 2.3. Pro přenos dat přes BLE jsem navrhl speciální protokol, který je blíže popsaný v sekci 3.5. Poslední způsob přenosu dat, který se v diagramu vyskytuje, je pojmenován `Unix/X11 sokety`. Tento způsob přenosu využívá ovladač pro přeposílání dat do cílových aplikací. Jeho využití a chování bude dále rozebráno v sekci 4.3.

### Získávání dat

Data budou získávána ze senzorů popsaných v sekci 2.4.1. Senzory nevytvářejí data spojitě, ale s určitou nastavitelnou frekvencí. Využití této skutečnosti bude popsáno v následujících sekcích této kapitoly.

### Předzpracování dat

Předzpracování dat umožňuje aplikovat určité výpočty a filtry na získaná data jako například transformace naměřených dat na data vyjadřující naměřenou hodnotu v základních jednotkách měřené veličiny, nebo využití komplementárního filtru pro přesnější výpočet absolutní polohy zařízení.

### Odesílání dat

Odesílání dat je zajištěno technologií BLE s využitím BLE modulu popsaného v sekci 2.4.1. Data jsou odesílána ve specializovaném binárním formátu definovaném protokolem popsaným v sekci 3.5.

### Přijímání dat

Tento bod je součástí knihovny popsané v části 3.3. Umožňuje přijatá data ze zařízení analyzovat a transformovat je zpět na data prostorová.

### Zpracování dat

Tento krok není pro životní cyklus dat povinný, popisuje pouze možnosti již zmíněné knihovny na transformaci prostorových dat na jiná užitečná data.

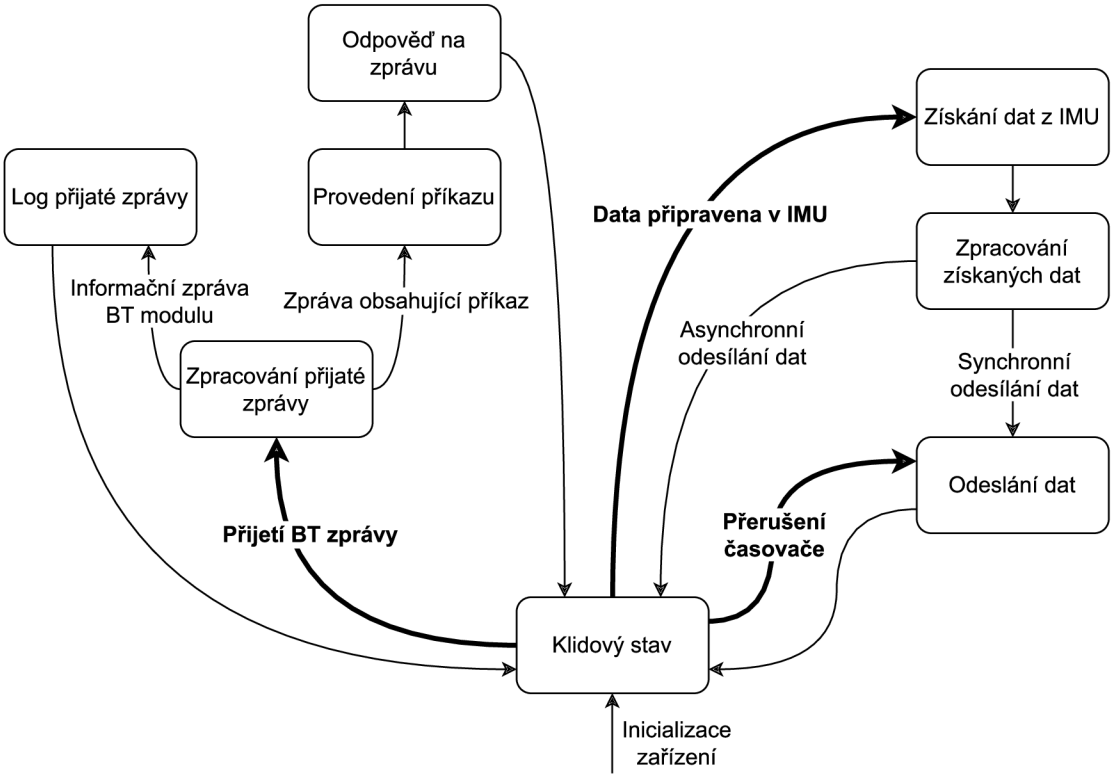
### Předání dat a jejich reprezentace

Bod předávání dat je uskutečněn pomocí speciálně upraveného ovladače pro komunikaci se zařízením RingOne. Tento ovladač je blíže popsán v sekci 3.4. Díky tomuto ovladači bude možné data vizualizovat v běžných 3D programech bez problému kompatibility. Bod reprezentace dat tedy není součástí programového řešení této práce, ale představuje jednu z možností využití naměřených dat.

## 3.2 Firmware

Jednou z hlavních částí této práce je právě návrh a následná implementace firmware pro zařízení RingOne. Jak bylo zmíněno v sekci 2.4, procesor tohoto zařízení umožňuje zpracovávat hardwarová přerušení a na této skutečnosti je založen celý návrh firmwaru.

V rámci návrhu jsem se rozhodl využít *Event–driven* přístup, který mi umožní rychle reagovat na jednotlivé události systému [16].




Obrázek 3.2: konečný automat popisující zpracování událostí v návrhu firmwaru

Na obrázku 3.2 je zobrazen diagram konečného automatu, který popisuje zjednodušený pohled na zpracování událostí. Systém zpracovává celkově tři hlavní události, `Data připravena v IMU`, `Přerušení časovače` a `Přijetí BT zprávy`. První událost nese název `Data připravena v IMU` a je generována komponentou IMU v případě, že jsou připravena naměřená data pro čtení. Následně jsou data zpracována a odeslána, bližší popis této části je v sekci 3.2.2. Druhá událost s názvem `Přerušení časovače` označuje, jak vyplývá z názvu, událost vytvořenou časovačem. Tato událost slouží k asynchronnímu odesílání dat a bude více probrána také v sekci 3.2.2. Poslední událostí, nacházející se v konečném automatu, je `Přijetí BT zprávy`. Tato událost je vygenerována při zaznamenání zachycení zprávy BT modulem. Její využití bude blíže popsáno v sekci 4.1.2.

### 3.2.1 Konfigurace zařízení

Zařízení také umožňuje přizpůsobit konfiguraci tak, aby vyhovovala potřebné aplikaci. Díky této možnosti změny konfigurace, bez nutnosti změny firmwaru, je zařízení univerzálním

nástrojem pro měření různých typů pohybu. Celková konfigurace zařízení by se dala popsat jako struktura. Její možná deklarace je naznačena následující částí kódu:

```
typedef struct device_config {
    uint8_t output_device;
    uint8_t output_timing;
    uint8_t data_source;
    struct {
        uint8_t frequency;
        uint8_t sensitivity;
    } gyro_config;
    struct {
        uint8_t frequency;
        uint8_t sensitivity;
    } accel_config;
    uint8_t data_processing;
    uint8_t data_filter;
    uint16_t async_period;
    bool run_at_startup;
    bool double_tap_enabled;
    int filter_data;
} device_config_t;
```

Každá možnost v konfiguraci má určité povolené hodnoty. Tyto povolené hodnoty se také mohou měnit v závislosti na předcházejících možnostech. Všechny povolené hodnoty a jejich omezení jsou popsané v sekci 4.1.3.

**output_device** Toto nastavení umožňuje zvolit, jakým způsobem budou data odesílána ze zařízení. Data ze zařízení je možné odesílat přes Bluetooth, UART rozhraní nebo obě možnosti najednou.

**output_timing** Tato možnost umožňuje uživateli zvolit mezi synchronním a asynchronním odesíláním dat, v případě synchronní možnosti jsou data odesílána v okamžiku, kdy jsou přečtena z IMU a v případě asynchronní možnosti jsou data odesílána při přerušení časovačem.

**data_source** Tato možnost umožňuje výběr senzorů, ze kterých budou data získávána. Vybírat je možné z gyroskopu, akcelerometru, nebo mohou být zvoleny oba senzory současně.

**gyro_config** Jedná se o strukturu, která popisuje nastavení gyroskopu. Struktura popisuje nastavení frekvence a senzitivity senzoru.

**accel_config** Stejně jako u předchozí možnosti se jedná o strukturu popisující nastavení senzoru, tentokrát ale nastavení akcelerometru.

**data_processing** Tato možnost popisuje, jak jsou naměřená data předzpracována před odesláním do cílového zařízení. Na výběr jsou zde tři hodnoty: žádné předzpracování, transformace naměřených dat na data vyjadřující naměřenou hodnotu v základních jednotkách měřené veličiny, nebo akumulace naměřených dat do absolutní orientace zařízení.

**data_filter** Tato možnost umožňuje uživateli vybrat filtr, který bude aplikován na naměřená a předzpracovaná data. Uživatel nemusí zvolit žádný filtr, nebo si může vybrat mezi komplementárním a mediánovým filtrem.

**async_period** Tato možnost popisuje periodu časovače pro asynchronní odesílání dat.

**run_at_startup** Tato možnost umožňuje spustit měření ihned po inicializaci firmwaru bez nutnosti čekání na externí příkaz.

**double_tap_enabled** Tato možnost umožňuje zapnout detekci dvojitého poklepání.

**filter_data** Jedná se o dodatečná data pro některé filtry.

### 3.2.2 Práce s naměřenými daty

Způsob, jakým jsou naměřená data zpracovávána, je plně závislý na konfiguraci zařízení popsané v předchozí sekci. Data procházejí jednotlivými úseky předzpracování a po dokončení každého z těchto úseků je podle aktuální konfigurace rozhodnuto, zda budou data předána následujícímu úseku, nebo budou odeslána a celkové předzpracování pro aktuální data bude ukončeno. První úsek reprezentuje načítání dat z IMU, druhý úsek reprezentuje transformaci naměřených dat na data vyjadřující naměřenou hodnotu v základních jednotkách měřené veličiny. Poslední úsek využívá tato data pro výpočet absolutní orientace zařízení. Po konci posledního úseku může být výsledná orientace odeslána, nebo je, v případě asynchronního odesílaní dat, uložena v zařízení. V případě asynchronního odesílání dat, je tato orientace odesílána pravidelně s periodou nastavenou v konfiguraci zařízení.

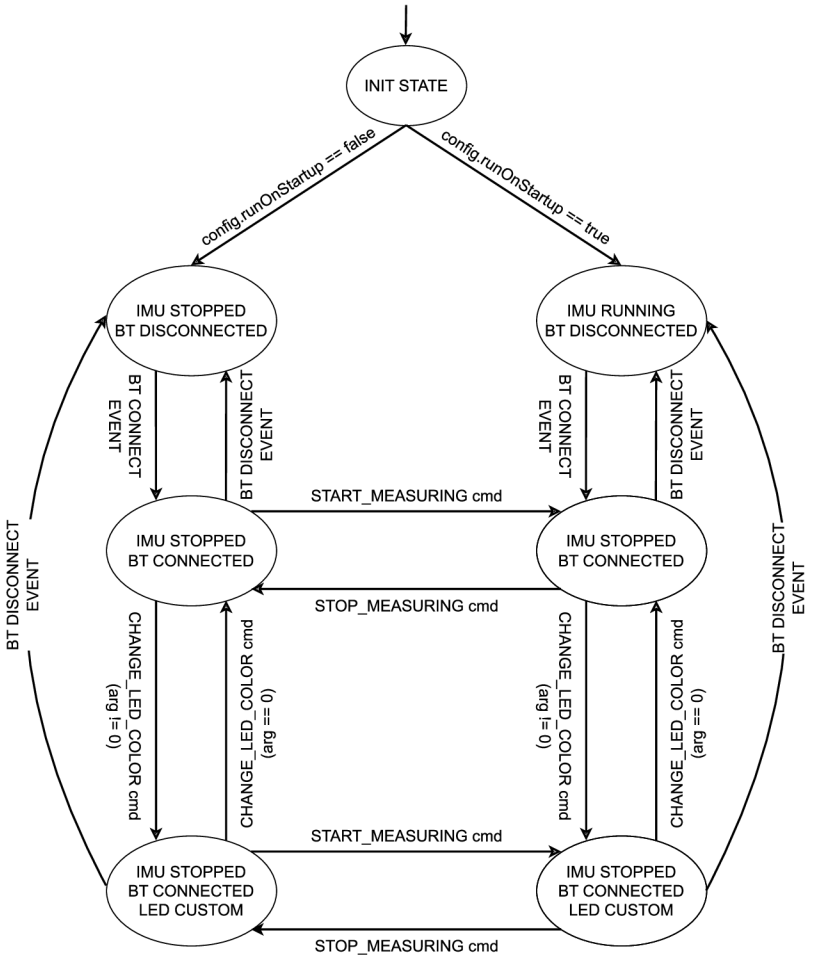
### 3.2.3 Indikace stavu

Jelikož je zařízení schopno pracovat zcela bezdrátově, může být těžké určit jeho aktuální stav. Proto je součástí návrhu také indikace stavu realizovaná pomocí RGB LED diody popsané v sekci 2.4.1. To umožní rychle a jednoduše identifikovat aktuální stav zařízení RingOne.

Na obrázku 3.3 je zobrazen diagram konečného automatu popisující změny stavu zařízení. Stavy zařízení, které jsou součástí identifikace stavu, jsou zaměřeny na stav Bluetooth připojení a na stav měření dat. Každý stav, kromě počátečního, se skládá se dvou podstavů. První podstav popisuje stav Bluetooth připojení a druhý podstav popisuje stav měření dat.

Ihned po zapnutí se zařízení dostává do počátečního stavu, z tohoto stavu ovšem odejde hned po inicializaci zařízení. V závislosti na aktuálním nastavení se zařízení dostane do stavu, kde je Bluetooth v nepřipojeném stavu a měření spuštěno nebo pozastaveno. Z těchto dvou stavů se zařízení může dostat pouze při události `BT CONNECTED EVENT`, která značí připojení externího zařízení přes Bluetooth. Následující změny stavů jsou závislé primárně na příkazech, které pošle externí připojené zařízení přes Bluetooth.

Počáteční stav je indikován zhasnutou LED diodou. Dále, pokud zařízení měří data, LED dioda svítí zeleně, pokud naopak měření není aktivní, LED dioda svítí modře. Pokud je připojeno externí zařízení přes Bluetooth, LED dioda svítí konstantně, pokud je Bluetooth modul v odpojeném stavu, LED dioda bliká. Toto ovšem neplatí pro případ explicitního nastavení barvy LED diody přes příkaz odeslaný přes Bluetooth. Pokud je LED dioda nastavena explicitně příkazem, svítí konstantně zvolenou barvou. Do původního stavu se dostane po resetování barvy příslušným příkazem, nebo odpojením externího Bluetooth zařízení.




Obrázek 3.3: Konečný automat indikace stavu zařízení.

## 3.3 Knihovna pro komunikaci se zařízením

Knihovna pro práci se zařízením RingOne (RingOne Library – ROL) plní úkol abstrakce komunikace. Díky této knihovně je umožněno jednoduché připojení k zařízení, rozsáhlá možnost nastavení konfigurace RingOne zařízení bez nutnosti přepisu jeho firmware a hlavně jednoduchý bezdrátový přenos naměřených data.

Prvním úkolem knihovny je umožnit jednoduchou identifikaci a rozpoznání správného zařízení a jeho následné připojení. Další důležitou částí je samotné získávání dat. Knihovna rozumí použitému protokolu a umí analyzovat příchozí data a interpretovat z nich data o pozici v prostoru. Tato data mohou být dále využita v jiných částech programu, který tuto knihovnu bude využívat. Dále je také možné naměřená data jednoduše exportovat do souboru ve formátu CSV.

Přeposílání dat bude umožněno i v opačném směru, tedy z knihovny do zařízení RingOne. Data posílaná v tomto směru budou sloužit primárně na konfiguraci tohoto zařízení, případně na kalibraci senzorů. Tato možnost je velmi důležitá primárně pro ROL protokol použitý pro komunikaci mezi zařízeními.

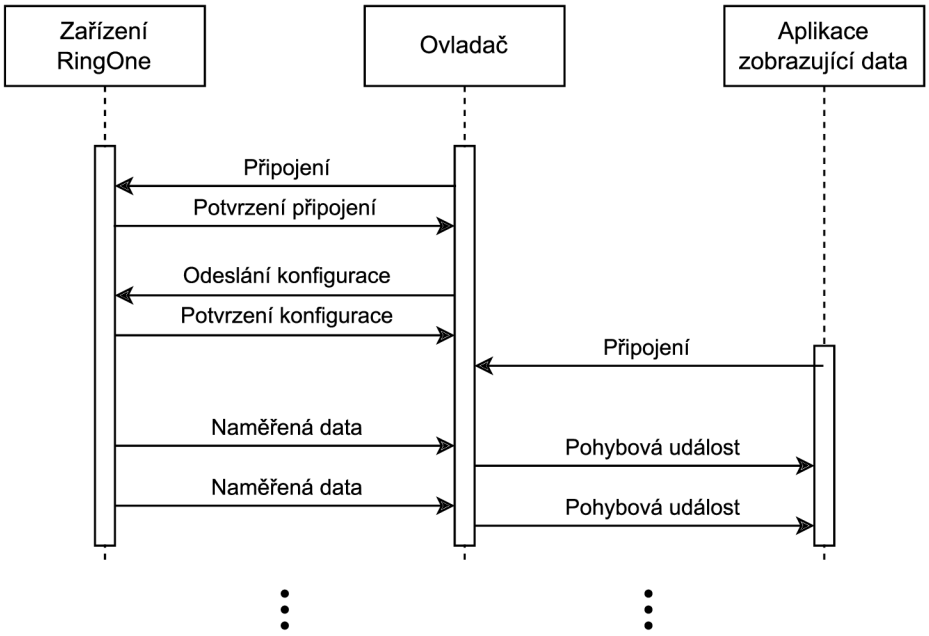
Tato knihovna tedy slouží jako pomyslný most mezi programy na uživatelově počítači a samotným zařízením RingOne.

Jelikož jsem během řešení této práce zjistil, že maximální množství dat přenesených přes BLE modul za jednotku času je značně více omezující, než výkon samotného procesoru zařízení RingOne, rozhodl jsem se určování základních odvozených dat z dat senzorických v části knihovna, dané bodem 4 v zadání práce, přesunout do části firmware. Díky této změně není nutné posílat všechna senzorická data přes Bluetooth, ale je možné vyžádat si pouze odvozená data, jako například inklinaci.

## 3.4 Ovladač

Tato část práce navazuje na již existující projekt Spacenav [40]. Tento projekt přináší open-source alternativu ovladače pro 3D vstupní zařízení a umožňuje připojení některých běžně využívaných CAD programů. S využitím knihovny popsané v předchozí sekci je tento ovladač upraven tak, aby umožňoval připojení a komunikaci se zařízením RingOne. Díky této úpravě bude možné jednoduše získaná data prezentovat v běžných CAD programech.

Na obrázku 3.4 je zobrazen sekvenční digram tohoto ovladače. V prvním kroku se ovladač připojí k zařízení RingOne a požádá o přepnutí jeho konfigurace na předpřipravenou konfiguraci určenou právě pro tento ovladač. Jakmile je konfigurace nastavena, zařízení RingOne začíná odesílat naměřená data. Od této doby se mohou začít připojovat externí aplikace, které chtějí využívat naměřená data. Ovladač naměřená data transformuje na speciální události definované v rámci projektu Spacenav a následně tyto události odešle všem připojeným externím aplikacím. Externí aplikace se mohou v průběhu běhu ovladače libovolně připojovat a odpojovat dle potřeby.




Obrázek 3.4: Sekvenční diagram ovladače.

## 3.5 ROL protokol

Protokol knihovny pro komunikaci se zařízením RingOne (RingOne library protocol – ROL protokol) je navrhnut pro rychlý a nenáročný přenos dat ze zařízení RingOne na cílové

zařízení a také pro jednoduchou možnost konfigurace zařízení RingOne. Tento protokol je zaměřen primárně na syntaktickou stránku zasílání dat, ovšem přináší i pár sémantických pravidel, které drží celý protokol pohromadě. Protokol je dělen na dvě části, kde první část popisuje formát odesílání naměřených dat z RingOne zařízení do cílového zařízení. Druhá část protokolu popisuje formát a chování příkazů posílaných ve směru opačném, které slouží ke konfiguraci a ovládání zařízení RingOne.

### 3.5.1 Formát protokolu

Při návrhu jsem se rozhodl využít binární formát pro posílání příkazů i zpráv namísto textového formátu. Tato volba umožní nejen ve většině případů menší velikost dat, ale také zajistí konstantní délku stejných typů dat usnadňující jejich zpracování. Například při odesílání tří číselných hodnot datového typu *integer* (je uvažován *integer* o velikost 4 bajty). V případě textového formátu při zápisu v CSV formátu s přidáním jednoho znaku pro identifikaci typu dat by výsledné zprávy mohly vypadat takto:

```
G,1,2,3
G,1000,2000,3000
G,123456,234567,345678
```

Z těchto příkladů je na první pohled vidět jejich proměnná délka. Data v takovémto formátu mohou mít délku od 7 bajtů (1 bajt pro identifikaci typu dat, 3 bajty pro symboly oddělující jednotlivé části a 3 bajty pro 3 jednociferná čísla) až po 34 bajtů (při maximální hodnotě datového typu *integer* s 10 ciframi).

Naopak při použití binárního formátu dat by data se stejným obsahem mohla vypadat takto (každá dvojice čísel představuje hodnotu jednoho bajtu v šestnáctkové soustavě):

```
01    00 00 00 01    00 00 00 02    00 00 00 03
01    00 00 03 E8    00 00 07 D0    00 00 0B B8
01    00 01 E2 40    00 03 94 47    00 05 46 4E
```

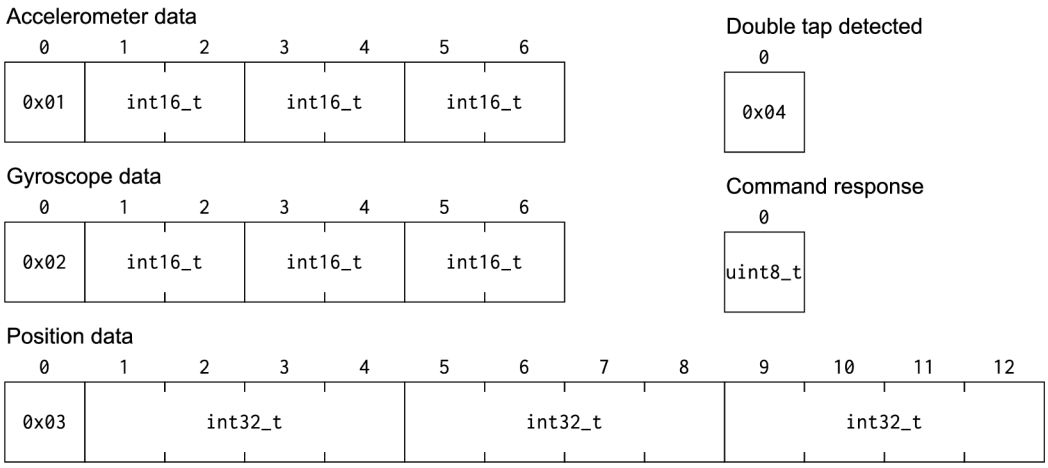
Na první pohled je zřejmé, že data s různými hodnotami mají stejnou velikost. Tato velikost je přesně 13 bajtů. Jelikož všechna data stejného typu budou mít konstantní velikost, není potřeba doplňovat symboly oddělující jednotlivé části dat.

Za předpokladu, že průměr přenášených hodnot je v hodnotách tisíců a vyšší, binární formát se z hlediska délky vyplatí. Ovšem i jednoduchost jeho zpracování, díky konstantní délce dat, je nezanedbatelnou výhodou.

### 3.5.2 Odesílání dat ze zařízení RingOne

Zařízení RingOne umožňuje odesílat několik různých typů prostorových dat podle potřeby a také odesílá odpovědi na přijaté příkazy. Na obrázku 3.5 je zobrazen binární formát jednotlivých zpráv, které umí RingOne odesílat.

`Accelerometer data` a `Gyroscope data` obsahují data naměřená jednotlivými senzory, tato data již mohou být upravena v závislosti na konfiguraci zařízení. Data popisující celkovou pozici zařízení musí mít speciální formát pro omezení ztráty přesnosti dat, na obrázku 3.5 je označený jako `Position data`. Poslední formát naměřených dat je označený jako `Double tap detected` a označuje detekci dvojitého poklepání na zařízení. Dále je na obrázku ještě jeden formát označený jako `Command response`. Tento formát zprávy slouží pro




Obrázek 3.5: Formát dat odesílaných ze zařízení RingOne.

odeslání statusu zpracování příkazu. V tabulce 3.1 jsou zobrazeny všechny podporované hodnoty, které se mohou vyskytnout ve zprávě tohoto formátu.

| **Hodnota** | **Název** |
|---|---|
| 0x80 | ACK |
| 0x81 | ERR_INVALID_OPCODE |
| 0x82 | ERR_INVALID_ARGUMENT_LENGTH |
| 0x83 | ERR_INVALID_ARGUMENT_VALUE |
| 0x84 | ERR_INVALID_CONFIG_COMBINATION |
| 0x85 | ERR_INTERNAL_ERROR |
| 0x86 | ERR_ALREADY_MEASURING |
| 0x87 | ERR_NOT_MEASURING |
| 0x88 | ERR_NOT_SAVED_IN_EEPROM |

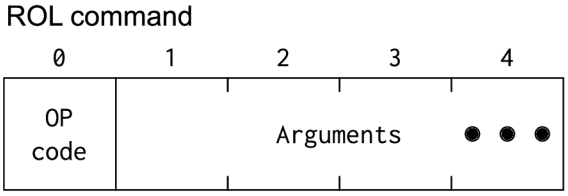
Tabulka 3.1: Tabulka možných odpovědí na přijatý příkaz.

Důležitá vlastnost všech zmíněných formátů zpráv je pevně definovaná hodnota prvního bajtu. Tato hodnota umožňuje rychlou a jednoduchou identifikaci daného formátu. Důležitá je také skutečnost, že u všech formátů obsahujících naměřená data, je nejvýznamnější bit (MSB – Most Significant Bit) roven 0 a naopak u zpráv odesílající status zpracovaného příkazu je MSB roven 1. Díky tomuto pravidlu je možné rychle a jednoduše odlišit tyto dva typy zpráv.

### 3.5.3 Odesílání příkazů do zařízení RingOne

Hlavním důvodem vzniku tohoto protokolu bylo umožnit uživateli jednoduše a strukturovaně ovládat a konfigurovat zařízení RingOne bez nutnosti změny jeho firmwaru. Pro tyto úkoly obsahuje ROL knihovna sadu příkazů. Všechny příkazy mají, stejně jako zprávy z předešlé sekce, prevně stanovený formát. Tento formát je zobrazen na obrázku 3.6.

První bajt zprávy obsahuje kód daného příkazu a následující bajty obsahují argumenty zvoleného příkazu. Délka argumentů příkazu není konstantní, ale mění se v závislosti na zvoleném příkazu.




Obrázek 3.6: Obecný formát příkazu ROL knihovny.

Podobně jako v předchozí sekci i zde je důležitá hodnota MSB prvního bajtu zprávy. Všechny příkazy ROL protokolu musí mít tento bit nastaven na hodnotu 1. Díky této skutečnosti zařízení RingOne pozná, že se opravdu jedná o zprávu obsahující některý z příkazů. Jelikož použitý Bluetooth modul, popsaný v sekci 2.4.1, posílá procesoru informace o stavu BLE spojení přes stejné rozhraní, přes které procesoru zasílá i přijaté zprávy, nebylo by bez nastavení MSB prvního bajtu zprávy na hodnotu 1 možné tyto dva typy od sebe odlišit. Všechny odeslané informace o stavu BLE zařízení obsahují pouze základní znaky ACII kódování, mají tedy vždy MSB každého znaku roven 0.

Celkově ROL protokol definuje 16 příkazů. Jednotlivé příkazy, jejich popis, operační kód, argumenty a omezení, jsou popsané v příloze B.

### 3.5.4 Komunikace pomocí ROL protokolu

Průběh komunikace mezi zařízením RingOne a aplikací využívající knihovnu, popsanou v sekci 3.3, zobrazuje sekvenční digram v příloze C. Průběh komunikace se dělí na 4 základní části: `Připojení`, `Konfigurace zařízení`, `Přenos měřených dat` a `Ukončení komunikace`.

Část komunikace označená jako `Připojení` musí vždy komunikaci začínat. Funkcionalita této části je přenesena na BLE protokol a ROL protokol nad ní vytváří pouze jednoduchou abstrakci pro snažší použití.

V části `Konfigurace zařízení` je s pomocí příkazů tohoto protokolu upravována konfigurace zařízení RingOne dle potřeby uživatele.

`Přenos naměřených dat` představuje část, kde zařízení RingOne posílá naměřená data do uživatelovy aplikace.

Části `Konfigurace zařízení` a `Přenos naměřených dat` jsou volitelné a nemusí se v průběhu komunikace objevit vůbec. Tyto sekce také nemusí následovat za sebou tak, jak je znázorněno v příloze C, ale mohou být v pořadí opačném, nebo se mohou dokonce prolínat mezi sebou. Vše závisí pouze na potřebě uživatele.

Poslední část komunikace nese označení `Ukončení komunikace`. Tato část se musí nacházet v každé komunikaci se zařízením RingOne a musí se nacházet vždy na konci komunikace. Stejně jako u části `Připojení`, i zde je většina této části jednoduchá abstrakce nad protokolem BLE. Součástí této části může být i volitelný příkaz `STOP_MEASURING` pro ukončení měření dat.

# Kapitola 4

# Implementace

Tato kapitola pro změnu představuje práci z hlediska praktického. Stejně jako při popisu návrhu bude tato kapitola rozdělena na 3 části, *Firmware*, *Knihovna* a *Ovladač*.

Všechny tyto části jsou implementovány v jazyce C a navíc část *Firmware* kombinuje jazyk C s jazykem C++. Při strukturování kódu jsem zvolil přístup monorepositáře, kde všechny části projektu jsou součástí jednoho repositáře, jsou ovšem logicky členěny a odděleny pro vyšší přehlednost.

Tato kapitola popisuje nejdůležitější části implementace této práce. Přesný popis všech funkcí ve všech částech implementace je k nalezení v programové dokumentaci, která je součásti repositáře. Součástí repositáře jsou také stromově členěné soubory *README.md*, dokumentující ostatní důležité informace o této práci, jako například způsob instalace, použití a správné spuštění všech částí projektu.

Jelikož vývoj vestavěných systémů vyžaduje často speciální nástroje a programy, součásti této práce bude také obraz Docker[1] kontejneru, který bude obsahovat všechny potřebné nástroje pro kompilaci zdrojových kódů nebo provedení jiných potřebných operací.

## 4.1 Firmware

Tato sekce popisuje nejdůležitější části implementace firmwaru zařízení RingOne. Celková implementace firmwaru by se dala rozdělit na 2 části, kde první část přestavuje nejnižší softwarovou vrstvu, v případě tohoto projektu řešenou knihovnou HAL, popsanou v sekci 4.1.1. A druhá část přestavuje hlavní funkcionalitu navrhnutou v rámci kapitoly 3.

Jak bylo zmíněno dříve tato část implementace je napsána v kombinaci jazyků C a C++. Podle popsaného rozdělení firmwaru by se dalo říct, že první část firmwaru je napsána v jazyce C a druhá část je napsána v jazyce C++. Druhá část je také implementována objektově orientovaným přístupem, kde jednotlivé komponenty jsou vždy reprezentovány odpovídajícím objektem pro jednodušší zpracování a přehlednější kód.

### 4.1.1 HAL knihovna

Při programování mikrokontrolerů existuje několik přístupů k řízení periferií MCU, kterými se může programátor vydat. Mezi nejpoužívanější přístupy programování mikrokontrolerů patří: *bare–metal programming*, použití *Hardware Abstraction Layer*, *Low–Level programming* nebo programování s použitím *Real-Time Operating System (RTOS)*.

[1] Viz https://www.docker.com/.

První zmíněná cesta nevyužívá žádné připravené knihovny či frameworky, ale programuje mikrokontroler „napřímo“. Tato cesta je populární z důvodu vysoké kontroly nad mikroprocesorem a flexibility. Při použití na větších projektech může ovšem být velmi náročná, programátor musí znát velmi dobře strukturu procesoru, jeho mapování paměti a mnoho dalších věcí. Implementace je většinou realizována s využitím nízkoúrovňových jazyků jako je příklad C nebo assembler [24].

Další zmíněnou možností je využití *Hardware Abstraction Layer (HAL)*. Tato abstraktní vrstva bývá běžně dostupná jako programová knihovna. Jelikož je každý procesor jiný, potřebuje mít svou HAL implementaci. Implementaci této vrstvy běžně poskytuje výrobce daného procesoru. HAL je vrstva abstrakce hardware, která poskytuje jednotné rozhraní pro komunikaci s periferiemi a dalšími hardwarovými funkcemi procesoru. Programátor využívá předdefinovaných funkcí v HAL knihovně pro inicializaci a ovládání různých periferií jako jsou UART, SPI, $I^2C$, GPIO a další. Tento přístup umožňuje programátorovi psát kód, který je nezávislý na konkrétním hardwarovém designu procesoru a umožňuje snadnou přenositelnost kódu mezi různými procesory určitého výrobce [47].

Možnost *Low–level programming* představuje programování mikrokontrolerů na úrovni mezi prvními dvěma zmíněnými způsoby. Stejně jako *Bare–metal programming* přístup zahrnuje tato možnost psaní nízkoúrovňového kódu, který přímo interaguje s hardwarem, ovšem s rozdílem možnosti použití vyšší úrovně abstrakce nebo knihovny, aby se proces programování zjednodušil. LL knihovna může například poskytovat funkce pro konfiguraci periferií a registrů mikrokontroleru nebo pro implementaci běžných operací, jako jsou vstup a výstup nebo přerušení. Na rozdíl od *Hardware Abstraction Layer* ale tento přístup neobsahuje standardizovanou sadu funkcí a abstrakcí, které umožňují vývojářům psát kód, který je nezávislý na hardware [30].

Poslední zmíněná možnost programování mikrokontrolerů využívá způsob RTOS. Jedná se o operační systém, který umožňuje vícenásobné úlohy běžet na mikrokontroleru současně. Programátor vytváří jednotlivé úlohy a určuje jejich prioritu. RTOS umožňuje lepší správu úloh a časování a může být velmi užitečný v aplikacích s vysokými nároky na reakční dobu. Tento přístup může být složitější, ale zároveň umožňuje vytvářet sofistikované aplikace s vyšší úrovní abstrakce [20].

Implementace této práce využívá cesty vrstvy abstrakce hardwaru. Přesněji, je použita HAL knihovna firmy STMicroelectronics [8]. Tato knihovna přináší všechny důležité funkce potřebné pro implementaci této práce.

### 4.1.2 Přerušení a DMA

Jak již bylo několikrát zmíněno v předešlých kapitolách, přerušení je velmi důležitá vlastnost pro tuto práci. Celý program je řízen různými přerušeními, vyvolanými vybranými akcemi některých periferií.

Celkově mikrokontroler reaguje na několik vybraných událostí. První událost nastává při změně logické hodnoty z log. 0 do log. 1 na mikrokontrolorovém pinu `PB12`. Tuto změnu provádí periferie IMU a označuje s ní, že má připravena prostorová data pro čtení. Při zachycení této změny mikrokonterolerem je vyvoláno přerušení a při jeho obsluze je volána vybraná funkce. Tato funkce je začátkem rutiny, která zajišťuje čtení dat z IMU, jejich předzpracování a případné odeslání v závislosti na aktuální konfiguraci zařízení. Druhou událost, kterou může IMU vyvolat, je událost zachycení dvojitého poklepání. Tuto událost ovšem IMU vyvolává pouze v případě, že je to vyžádáno konfigurací zařízení. Událost je vyvolána změnou logické hodnoty z log. 0 do log. 1 na mikrokontolorovém pinu `PB1`. Při

zachycení této události je vyvolána rutina, která dle konfigurace zařízení odešle na vybrané výstupní rozhraní informaci o zaznamenání dvojitého kliknutí.

Další možné přerušení je vyvoláváno interním časovačem, který jej vyvolává s nastavenou konstantní periodou. Toto přerušení je vyvoláváno pouze při specifickém nastavení konfigurace a to při nastavení hodnoty `output_timing` na asynchronní hodnotu. Vyvoláním tohoto přerušení je spuštěna rutina, která na konfigurací vybraná výstupní rozhraní odešle informace o aktuální orientaci.

V rámci popisu zařízení RingOne v sekci 2.4.1 bylo zmíněno, že zvolený procesor podporuje funkci DMA. Tato funkce umožňuje periferiím zapisovat data přímo do paměti procesoru bez nutnosti zapojení výpočetního jádra mikrokontroleru. Data jsou zapisována do paměti na předem stanovenou adresu a periférie mohou pro zápis použít některý z hardwarových protokolů, jako například UART, SPI a další.

Funkce DMA je v rámci implementace využita pro neblokující čtení dat z BLE modulu. Data přečtená z BLE modulu obsahují informace o změně stavu spojení, ale hlavně také obsahují zprávy přijaté z připojených externích zařízení. Jelikož je čekání na data z BLE modulu neblokující, v době čekání se může procesor soustředit na jiné události, například číst data z periférie IMU . Při zápisu posledního bajtu zprávy z BLE modulu do paměti je vygenerováno přerušení označující dokončení přenosu dané zprávy a je zavolána funkce začínají rutinu obsluhy přijaté zprávy.

Při práci s přerušením je také důležitá jejich priorita, která umožňuje více prioritním přerušením být obsloužena dříve, než méně prioritním přerušení. V případě implementace této práce mají všechny události označující přerušení stejnou prioritu. Díky této skutečnosti je zajištěno, že v případě výskytu více různých přerušení ve stejný čas, budou tato přerušení obsloužena postupně, jedno po druhém. Což má za následek zamezení výskytu vyhladovění některých přerušení, nebo také zamezení výskytu uváznutí při přístupu ke zdrojům, jelikož mají jednotlivé obslužné rutiny vždy zajištěno, že všechny zdroje jsou volné. Aby tento přístup fungoval, je také důležité, aby ve funkci `main` byla kromě inicializace zařízení pouze prázdná nekonečná smyčka nebo byl procesor po dokončení akce vyvolané přerušením uspán.

### 4.1.3 Konfigurace zařízení

Implementovaná konfigurace dává uživateli možnost změnit chování zařízení RingOne bez nutnosti změny jeho firmwaru. To zajišťuje univerzálnost zařízení a možnost jeho použití v různých aplikacích.

Jak již bylo zmíněno v návrhu řešení v sekci 3.2.1, konfigurace má podobu struktury v jazyku C. Struktura použitá při implementaci je stejná jako struktura předvedená v návrhu. Jednotlivé prvky jsou seřazeny pro nejefektivnější paměťové uspořádání vzhledem k *structure alignment* jazyka C. *Structe alignment* je technika používaná k zarovnání datových prvků v rámci struktur, aby byly adresy těchto prvků násobky velikosti datového typu.

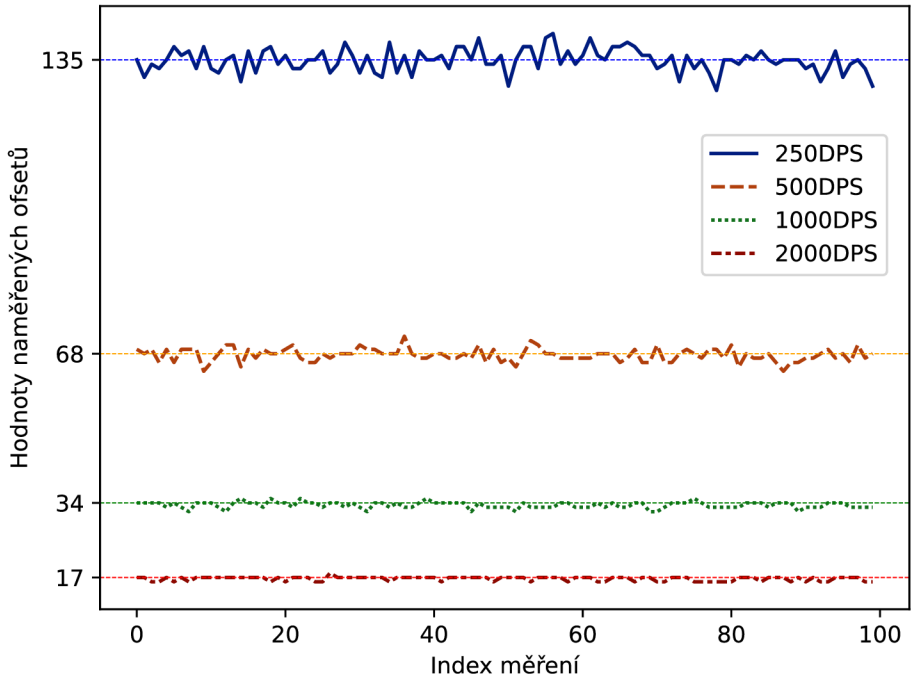
Většina prvků struktury konfigurace je datového typu `uint8_t`. Tento datový typ může nabývat hodnot od 0 do 255, ovšem povolené hodnoty prvků této struktury jsou značně omezenější. Proto by se mohlo zdát, že použití datového typu `uint8_t` je značně neefektivní. Jedná se ale o nejmenší možný adresovatelný datový typ v jazyce C a byl vybrán pro zjednodušení práce s konfigurací zařízení v rámci firmwaru.

Všechny prvky struktury mají jasně definované hodnoty, kterých mohou nabývat. Tyto hodnoty jsou shrnuty v tabulce nacházející se v příloze D. Na některé prvky konfigurační

struktury se také vztahují určitá omezení. Tato omezení jsou často závislá na hodnotách předešlých prvků struktury, proto je velmi důležité nastavovat jednotlivé prvky v pořadí, v jakém jsou definovány v konfigurační struktuře.

Konfiguraci je také možné uložit do EEPROM paměti popsanou v sekci 2.4.1. Při inicializaci se zařízení vždy pokouší načíst konfiguraci uloženou v této paměti. Pokud je úspěšné, používá tuto konfiguraci, pokud ne, používá konfiguraci výchozí. Výchozí konfiguraci používá zařízení také pokud v EEPROM paměti žádná konfigurace není uložena. Zda je konfigurace v paměti uložena, zařízení pozná pomocí vybraných bajtů v EEPROM paměti, které indikují stav konfigurace uložené v paměti. Pro uložení konfigurace do EEPROM paměti ROL protokol implementuje příkaz `SAVE_CONFIG_TO_EEPROM`.

Důležitými daty pro RingOne zařízení jsou také hodnoty offsetů jednotlivých senzorů. Tyto hodnoty reprezentují posun hodnot naměřených senzory od hodnot reálných, po odečtení těchto hodnot od hodnot naměřených se stávají naměřená data daleko více přesnější. Konfigurace RingOne také umožňuje tyto offsety uložit do paměti EEPROM obdobně jako konfiguraci zařízení. Výchozí hodnoty offsetů všech os všech senzorů jsou 0. Proto je nutné před používáním měření tyto offsety nastavit. Pro nastavení offsetů senzorů implementuje ROL protokol příkaz `SET_IMU_OFFSETS` a také příkaz `SAVE_IMU_OFFSETS_TO_EEPROM` pro uložení offsetů do EEPROM paměti.




Obrázek 4.1: Hodnoty offsetů gyroskopu naměřených v ose X při různých citlivostech.

Určení offsetů je nutné provádět vždy při nejvyšší citlivosti senzorů (250 $DPS$ u gyroskopu a 2 $G$ u akcelerometru). Jelikož ofsety ostatních citlivostí senzorů jsou vždy násobky těch nejvyšších, je možné z offsetů pro nejnižší citlivosti odvodit i offsety pro citlivosti zbylé. Tato hypotéza je potvrzena grafem na obrázku 4.1, kde jsou zobrazeny offsety osy X v gyroskopu při různých citlivostech. Z tohoto grafu je jasně vidět, že hodnoty nejvyšších

offsetů jsou násobky těch nižších. Při měření offsetů bylo vždy použité 100 vzorků každé citlivosti a výsledná hodnota ofsetů je rovna mediánu naměřených hodnot v rámci jedné citlivosti. Tyto mediánové hodnoty jsou v grafu reprezentovány vodorovnými přerušovanými přímkami s odpovídající barvou.

Postup měření offsetů gyroskopu je velmi jednoduchý, stačí nechat zařízení v klidové poloze a pouze přečíst měřená nezpracovaná data z gyroskopu. Tato data přímo odpovídají hodnotám offsetů jednotlivých os gyroskopu. Příklad pro program měřící hodnoty offsetů se nachází mezi ukázkovými programy, které jsou součástí subrepozitáře obsahujícího implementaci knihovny ROL. Tyto ukázkové programy se nacházejí ve složce s názvem `examples` a program měřící hodnoty offsetů se nachází v její podsložce s názvem `MeasureGyroOffsets`.

Postup měření offsetů akcelerometru je mírně složitější, než tomu bylo u gyroskopu. Nezpracované hodnoty je potřeba měřit ve více polohách tak, aby při každém měření byla jedna osa akcelerometru svislá vzhledem k okolnímu světu. Po získání hodnot ze všech potřebných pozic je možné v každém jednom měření zjistit offsety těch os, které nebyly svislé. Následné nastavení a uložení hodnot offsetů v zařízení RingOne je stejné jako v případě offsetů gyroskopu.

## 4.2 Knihovna ROL

Jak bylo naznačeno v návrhu knihovny v sekci 3.3, nejdůležitější úkol knihovny je propojení a komunikace se zařízením RingOne. Proto se většina implementace této knihovny zaobírá Bluetooth komunikací. Knihovna je navržena pro operační systém Linux a byla vyvíjena na distribuci Ubuntu verze 20.04. Operační systém Linux přináší protokolový stack pro Bluetooth s názvem BlueZ, který umožňuje pracovat s Bluetooth zařízeními na tomto systému.

### 4.2.1 BlueZ

BlueZ [33] je oficiální protokolový stack pro Bluetooth na Linuxu, který poskytuje podporu pro jádro Bluetooth vrstev a protokolů.

Stack BlueZ poskytuje širokou škálu Bluetooth profilů jako jsou Audio/Video Remote Control Profile (AVRCP), Hands-Free Profile (HFP), Headset Profile (HSP), Advanced Audio Distribution Profile (A2DP) a mnoho dalších. Tyto profily umožňují zařízením založeným na Linuxu komunikovat s jinými zařízeními s Bluetooth, jako jsou chytré telefony, sluchátka a reproduktory.

BlueZ také poskytuje sadu nástrojů příkazového řádku, včetně hciconfig, hcitool a hcidump, které umožňují uživatelům spravovat Bluetooth zařízení a spojení z příkazové řádky. Tyto nástroje lze použít pro úkoly jako je skenování blízkých Bluetooth zařízení, párování a odpojování zařízení a monitorování provozu Bluetooth.

Pro automatizované využití Bluetooth protokolu poskytuje BlueZ rozhraní D–Bus pro komunikaci mezi aplikacemi a Bluetooth zařízeními. D-Bus je interprocesová komunikační sběrnice na Linuxu, která umožňuje aplikacím komunikovat s jinými aplikacemi a systémovými službami. V případě Bluetooth na Linuxu se D-Bus používá k propojení aplikací s BlueZ, což umožňuje aplikacím přístup k Bluetooth zařízením.

Při vývoji aplikací využívající Bluetooth na operačním systému Linux je také velmi užitečný nástroj příkazové řádky *bluetoothctl*. Bluetoothctl využívá rozhraní D-Bus poskytované BlueZ k ovládání a konfiguraci Bluetooth zařízení přímo z příkazové řádky.

### 4.2.2 SimpleBLE

Pro jednodušší a přehlednější implementaci jsem se rozhodl využít open–source knihovnu s názvem SimpleBLE [13]. Tato knihovna poskytuje jednoduchou abstrakci nad zmíněným protokolovým stackem BlueZ. Knihovna umožňuje prvotní stavení Bluetooth komunikace a následné naslouchání na *notify* události odeslané z připojeného zařízení. Při zachycení *notify* události je volána vybraná callback funkce, která se stará o zpracování přijatých dat. Také na tomto principu funguje knihovna ROL, která vytváří mezivrstvu mezi knihovnou SimpleBLE a uživatelovou implementací. Nejdůležitějším úkolem této mezivrstvy je implementace funkcí pro zpracovávání ROL protokolu.

Výsledná implementace ROL knihovny umožňuje uživateli nastavit a realizovat spojení se zařízením RingOne a pomocí připravených funkcí odesílat tomuto zařízení příkazy definované ROL protokolem. V průběhu jsou data přijatá ze zařízení RingOne zpracována a data o poloze zařízení jsou předána uživatelem definované callback funkci pro následné zpracovaní.

## 4.3 Ovladač

Poslední implementační část upravuje dříve zmíněný Spacenav ovladač pro možnost komunikace se zařízením RingOne. Většinu této části tvoří samotný Spacenav ovladač, který je několika novými a několika upravenými funkcemi upraven na komunikaci s RingOne.

V Linuxových systémech se používají dva typy ovladačů – ovladače pro Linuxové jádro (kernel drivers) a uživatelské ovladače (user-space drivers). Ovladače pro Linuxové jádro jsou součástí jádra operačního systému a mají přímý přístup k hardwaru počítače. Tyto ovladače poskytují jádru operačního systému rozhraní pro komunikaci s hardwarovými zařízeními, jako jsou například periferní zařízení, disky nebo síťové karty. Ovladače pro Linuxové jádro se obvykle provozují v režimu jádra (kernel mode) a jsou výkonné a rychlé. Avšak vzhledem k tomu, že jsou tyto ovladače součástí jádra operačního systému, jejich vývoj a úpravy jsou náročné a vyžadují hlubší znalosti operačního systému.

Uživatelské ovladače jsou umístěny v uživatelském prostoru operačního systému a jsou zprostředkovatelem mezi aplikacemi a jádrovými ovladači. Tyto ovladače poskytují rozhraní pro komunikaci aplikací s hardwarovými zařízeními a používají jádrové ovladače pro přístup k hardwaru. Uživatelské ovladače se obvykle provozují v uživatelském režimu (user mode) a jsou bezpečnější, protože chyby v ovladači nemohou ovlivnit celý systém. Nicméně uživatelské ovladače mohou být pomalejší než jádrové ovladače, protože vyžadují komunikaci s jádrem operačního systému [12].

### 4.3.1 Spacenavd

Spacenavd je démon pro Linux, který poskytuje ovladače zařízení 3Dconnexion pro Linux. Zařízení 3Dconnexion jsou specializovaná vstupní zařízení používaná pro 3D modelování a CAD aplikace. Spacenavd umožňuje použití těchto zařízení s Linuxovými systémy a poskytuje podporu pro různá zařízení 3Dconnexion, včetně SpaceNavigator, SpaceMouse a SpacePilot. Projekt spacenavd je open source a k dispozici pod licencí GPL.

Ovladač spacenavd používá Unix sokety (UNIX domain sockets) nebo X11 sokety pro distribuci událostí informující o změně polohy ve 3D prostoru. Tento ovladač žádným způsobem nezasahuje do jádra systému a pro komunikaci využívá výhradně zmíněných soketů. Z tohoto důvodu může být Spacenav považován za uživatelský ovladač [40].

### 4.3.2 Unix/X11 sokety

Unix sokety jsou typem komunikačního kanálu, který umožňuje dvěma nebo více procesům komunikovat přímo mezi sebou. Tyto sokety jsou typicky vytvářeny na základě souborového systému, konkrétně jsou vytvářeny jako soubory v adresáři `/tmp` nebo jiném speciálním adresáři. Unix sokety jsou běžně používány pro komunikaci mezi procesy na stejném počítači, například pro komunikaci mezi serverovým a klientským procesem [35].

X11 sokety jsou specifickým typem Unix soketů používaných v X Window Systému, což je grafický subsystém používaný v mnoha Unixových systémech, včetně Linuxu. Tyto sokety slouží k vytvoření komunikačního kanálu mezi X serverem a X klientem. X klienti jsou běžně grafické aplikace, jako například okenní manažery nebo aplikace pro kreslení. X11 sokety umožňují X klientům přistupovat ke grafickému výstupu a vstupu, který je poskytován X serverem. X11 sokety jsou také vytvářeny jako soubory v adresáři `/tmp`, ale mají speciální název, například `/tmp/.X11-unix/X0` [28].

# Kapitola 5

# Testování

Testování je důležitá část této práce, která slouží k ověření správnosti návrhu a také implementace. V případě pozitivních výsledků testování těchto částí jsou následně testovány vlastnosti celkového systému jako například jak naměřená data odpovídají realitě nebo zpoždění přenosu dat ze zařízení RingOne do uživatelské aplikace.

Testování návrhu a implementace je provedeno v rámci ukázkových příkladů nacházejících se v repositáři při implementaci ROL knihovny. Jednotlivé ukázkové příklady testují vybrané části sytému a zároveň názorně ukazují způsob použití knihovny budoucím uživatelům. Implementace upraveného Spacenav ovladače také do značné míry testuje systém jako celek. Všechny tyto testovací aplikace prokazují, že systém funguje dle očekávání.

## 5.1 Správnost naměřených dat

Jedním z nejdůležitějších ohodnocení systému je, jak naměřená data odpovídají realitě. Pokud by data byla zcela nesmyslná, tak i se všemi ostatními částmi funkčními by toto zařízení nemělo žádné využití.

Testy využívají vlastnosti zařízení RingOne umožňující odesílat data reprezentující absolutní orientaci zařízení. Data byla měřena v několika různých pozicích a následně porovnávána s referenčními hodnotami.

Testovány byly pozice otočené kolem os X a Y. Jednotlivé pozice měly úhel otočení od 0° do 90° s krokem 5°. Každé měření tedy obsahuje 19 hodnot.

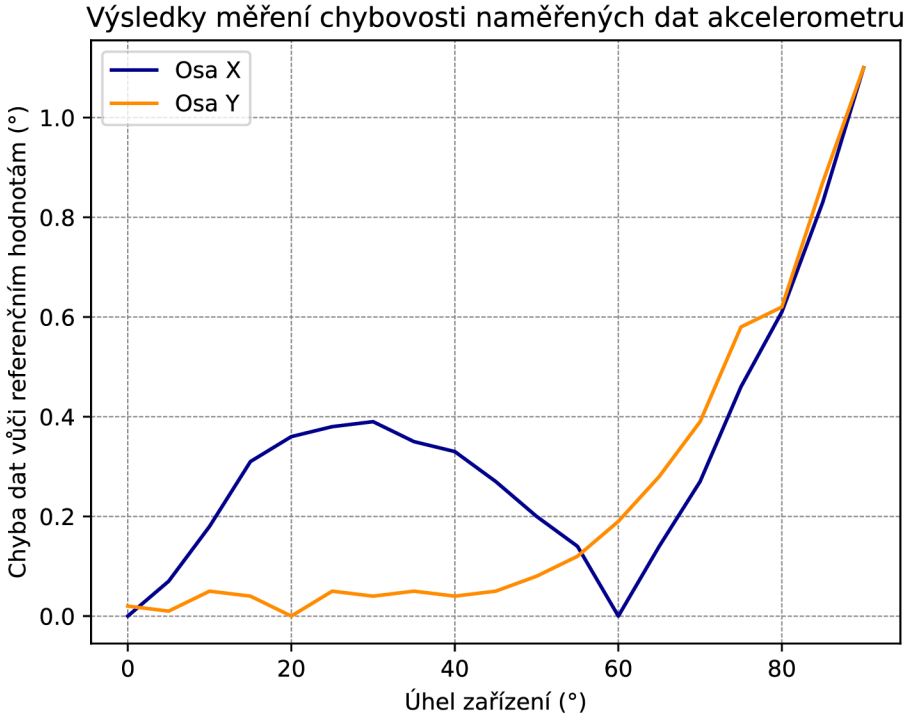
Dále byly testy rozděleny do dvou skupin, kde první skupina reprezentovala určování polohy výpočtem inklinace z dat akcelerometru a druhá skupina reprezentovala měření gyroskopu, kde je poloha určená integrací naměřených hodnot. V rámci testů polohových dat nebyl uvažován komplementární filtr, jelikož z pohledu funkčnosti reprezentuje okamžité změny jako hodnoty z gyroskopu a trvalé pozice určuje hodnotami akcelerometru. Výsledná data by tedy měla neurčitý poměr těchto hodnot a vyhodnocení testů by bylo velmi náročné.

Jako referenční hodnoty jsou v rámci všech testů naměřených hodnot použity hodnoty IMU jednotky mobilního telefonu. Zařízení RingOne je k mobilnímu telefonu napevno přichyceno tak, aby jednotlivé osy senzorů byly rovnoběžné. Následně je celá tato soustava otáčena do dříve popsaných měřících pozic. IMU jednotka mobilního telefonu byla před měřením kalibrována jeho interním algoritmem, ovšem i tak není zaručeno, že referenční hodnoty jsou zcela správné, z výsledků testů se však pro manuální ověření prostorových dat zdají být dostačující.

Na obrázku 5.1 je zobrazen graf výsledků testování hodnot inklinace vypočtené z dat akcelerometru. Oranžově vyobrazená data reprezentují měření osy Y. Z tvaru grafu jde jasně vidět, že s hodnotami blížícími se 90 stupňům se zvětšuje také chyba naměřených dat. Tato chyba může být zapříčiněna způsobem výpočtu dat, jelikož je pro výpočet úhlu otočení kolem této osy využita inverzní goniometrická funkce arkus sinus. Tato funkce rychle roste při vstupních hodnotách blížících se 1 a tedy při výstupních hodnotách blížících se 90°, tyto hodnoty se tedy musí mapovat na menší úsek, což má za následek velkou změnu výstupní hodnoty při malé změně vstupní hodnoty.

Obdobně při hodnotách naměřených při otáčení kolem osy X je zde možné vidět jasný trend zvětšování chyby měření se zvyšujícím se úhlem polohy. Zde ovšem nejsou hodnoty získávány výpočtem používající funkci arkus sinus, ale funkci arkus tangens. Tato funkce má asymptotu na ose y v hodnotě $2\pi$, reprezentující hodnotu 90°, znamená to tedy, že zde stejný problém jako u předchozího měření nenastane. Ovšem argument této funkce je zlomek, jehož jmenovatel dosahuje při náklonu 90° k nule se blížících hodnot a naopak jmenovatel dosahuje hodnot největších. A jelikož je jmenovatel velmi malá hodnota a naopak čitatel je hodnota velká, jakákoliv změna některé z těchto hodnot má za následek velkou změnu na výstupu.

Důležité je také zmínit, že zaznamenávání jednotlivých hodnot měření bylo prováděno v klidových stavech zařízení, pokud by byla inklinace vypočítávána z hodnot při pohybu zařízení, který není rovnoměrný a přímočarý, výsledné hodnoty by se značně lišily od reality.

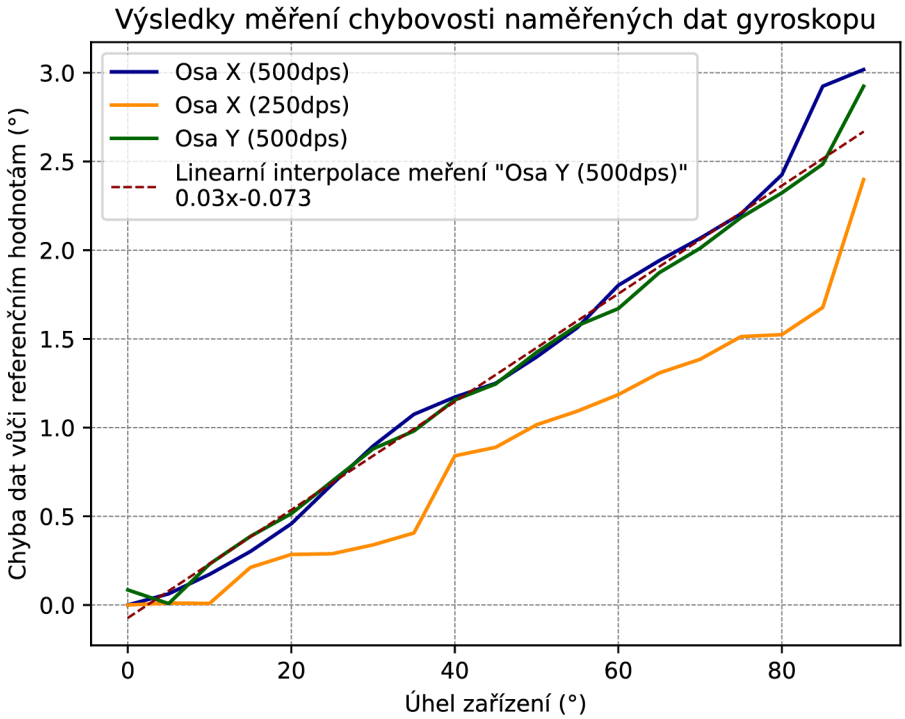



Obrázek 5.1: Hodnoty chybovosti dat akcelerometru.

Hodnoty měření testující gyroskop jsou zobrazeny na grafu 5.2. Zde jde vidět jasný rostoucí trend chyby měření. Tento trend je zapříčiněn způsobem výpočtu pozice z dat gyroskopu. Jednotlivá měření jsou integrována do výsledné pozice, ovšem s těmito daty

je integrována i chyba měření, která je postupně akumulována a stává se stále větší. Tuto chybu jde snížit zvolením větší citlivosti jak jde vidět z grafu 5.2 u hodnot osy X při citlivosti 500 dps a 250 dps i přesto ovšem chyba nezmizí a je stále akumulována s následkem zvětšující se absolutní chyby. Dalším problémem využití nižší citlivosti měření je menší maximální zaznamenatelná rychlost otáčení, což v některých případech aplikace nemusí být žádoucí. Dále je také z tohoto grafu vidět, že chyby měření při osách X a Y při stejné citlivosti jsou téměř stejné, což naznačuje konzistenci v rámci jednotlivých čidel gyroskopu.

Jelikož naměřené hodnoty tvoří téměř lineární funkci, při lineární interpolaci například měření „Osa Y (500dps)“ má koeficient stoupání výsledné přímky hodnotu 0.03, tato interpolační přímka je také vidět na grafu 5.2. Zjištěná hodnota koeficientu stoupání označuje, že na každý jeden stupeň otočení je chyba 0.03 stupně, neboli se jedná o 3 % chybu.




Obrázek 5.2: Hodnoty chybovosti dat gyroskopu.

Z výsledků měření jsou vidět jednotlivé výhody i nevýhody využitých senzorů popsané v teoretické části této práce. Jak bylo také zmíněné v teoretické části, tyto nevýhody jednotlivých senzorů mohou být potlačeny využitím komplementárního filtru, kde náhlé změny orientace jsou zachyceny gyroskopem a dlouhodobá pozice je korektována inklinací vypočítanou z dat akcelerometru.

## 5.2 Zpoždění přenosu

Další důležitou vlastností systému, kterou jsem měřil, je zpoždění přenosu, neboli latence. Latence se skládá z několika faktorů jako je zpoždění přenosu dat, zpracování signálu v počítači a reakční doba výstupního zařízení. Aby mohlo být zařízení RingOne využíváno opravdu

univerzálně, je důležité, aby latence měla nízké hodnoty. Pokud by latence byla vysoká, například jednotky až desítky sekund, použitelnost zařízení by byla značně omezena.

Pro testování latence jsem si připravil dva typy testů, první test využívá příkazu `PING`, který je součástí ROL protokolu, a druhý test využívá pro zjištění latence vysokorychlostní kameru.

### Test 1.

Tento test využívá příkazu `PING`, který definuje ROL protokol. Latence je určena jako rozdíl mezi časem odeslání příkazu a časem přijetí odpovědi na příkaz ze zařízení RingOne. Program, pomocí kterého je toto měření uskutečněno, se nachází v repositáři při implementaci ROL knihovny ve složce `examples` a má název `TestLatency`. Nejdůležitější část kódu tohoto programu vypadá následovně:

```
1    clock_t start, end;
2    double latency;
3
4    start = clock();
5    ROL_sendCmd(PING, NULL, 0, 100);
6    end = clock();
7    latency = (double)(end - start) / CLOCKS_PER_SEC *1000;
```

Na řádku 4. je uložen počet taktů procesoru od začátku spuštění programu reprezentující čas odeslání příkazu. Následně je na řádku 5. volána funkce `ROL_sendCmd()`. Jelikož je tato funkce blokující, skončí až ve chvíli, kdy je přijata odpověď od zařízení RingOne. Na řádku 6. je uložena hodnota taktů procesoru od začátku programu, reprezentující čas přijetí odpovědi příkazu. Na posledním řádku je převeden rozdíl dvou změřených časů a převeden na hodnotu vyjadřující počet milisekund.

Na grafu 5.3 jsou zobrazeny hodnoty sta měření latence pomocí příkazu `PING`. Z grafu je dobře vidět, že většina měření má hodnotu kolem 2 milisekund, ovšem jednou za několik měření je výsledná hodnota přibližně 32 ms. Tyto výkyvy ovšem na celkovém průměru nejsou tolik znát. Průměrná latence má hodnoty 2.581 ms a medián je 1.981 ms.

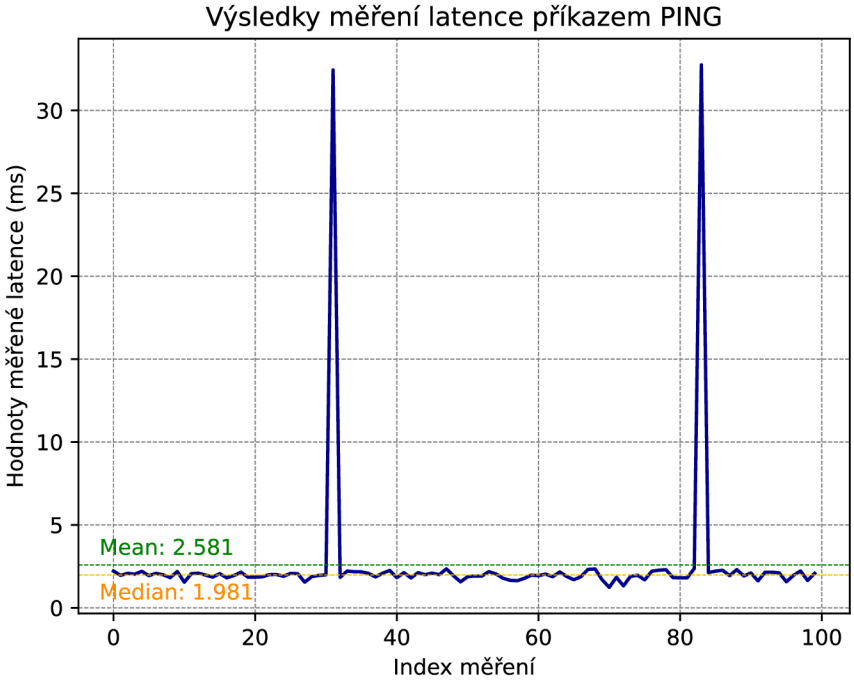
Zmíněné výkyvy v měření latence mohou být zapříčiněny například nevhodně zvoleným BLE GATT profilem s kombinací s jeho prací s vyrovnávací pamětí BLE modulu. Řešením tohoto problému by mohlo být například vytvoření vlastního GATT profilu určeného pro efektivní odesílání naměřených dat.

Důležité je také uvědomit si, že naměřené hodnoty reprezentují dva přenosy dat, první přenos přenáší příkaz na zařízení RingOne a druhý přenos přenáší odpověď do uživatelovy aplikace. Výsledná latence pro prostorová data bude tedy poloviční.

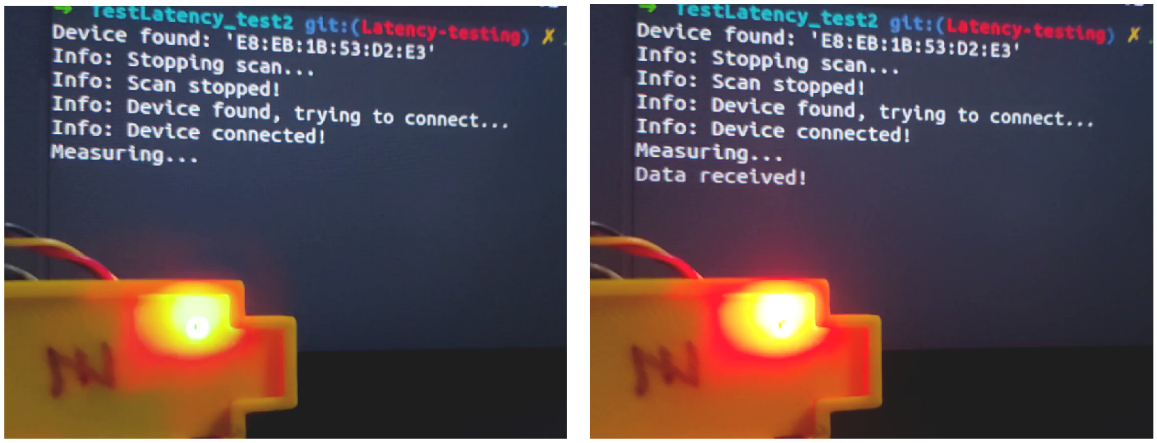
### Test 2.

Tento test má za cíl změřit celkovou latenci systému od začátku čtení dat z IMU jednotky po zobrazení naměřených dat na uživatelově monitoru. Jelikož toto měření obsahuje více událostí než jen samotný přenos dat přes Bluetooth, dá se očekávat, že výsledné hodnoty naměřené latence budou o něco větší, něž tomu bylo u předešlého testu.

Při tomto měření byl využit upravený firmware zařízení RingOne, který při začátku čtení dat z IMU komponenty rozsvítí LED diody červenou barvou jako indikaci začátku měření. Dále je využita jednoduchá aplikace využívající ROL knihovny, která při přijetí prostorových dat vypíše zprávu na standardní výstup. Celý tento proces je měřen vysokorychlostní kamerou s frekvencí 960 snímků za sekundu.




Obrázek 5.3: Hodnoty latencí pomocí příkazu PING.




(a) Snímek číslo 64.

(b) Snímek číslo 157.

Obrázek 5.4: Vybrané snímky vysokorychlostní kamery zachycující hlavní události testu.

Na snímcích 5.4a a 5.4b jsou vidět hlavní události popsaného testu. Snímek 5.4a zobrazuje rozsvěcení diody při počátku čtení dat z IMU komponenty a snímek 5.4b zobrazuje událost zobrazení zprávy oznamují příjem dat na monitor počítače.

První snímek má číslo 64 a druhý 157, tato čísla označují index snímku v rámci natočeného videa. Znamená to tedy, že se mezi zmíněnými událostmi stihlo zaznamenat celkem 93 snímků. Pokud má kamera frekvenci 960 snímků za vteřinu, snímek je zaznamenáván přibližně každých 1.04 ms. 96 snímků by tedy bylo zaznamenáno za 100 ms.

Latence 100 ms je několikanásobně více než výsledek z prvního testu, je ovšem důležité si uvědomit, že za tuto dobu jsou navíc data přečtena z IMU komponenty, předzpracována a mají také větší velikost, než data přeposílaná v prvním testu. Dále také přijatá data na uživatelově počítači nejsou na monitoru zobrazeny ihned, ale se zpožděním několika milisekund, v extrémních případech i desítek milisekund, pokud se jedná o monitor s nízkou obnovovací frekvencí.

### Vyhodnocení testů latence

Z testování latencí počítačových myší z [43] vyplývá, že Bluetooth myši určené pro hraní her mají latenci kolem 10 ms a myši kancelářské mají průměrnou latenci do 60 ms. V těchto testech je také vidět vysoká nekonzistence mezi jednotlivými opakovanými měřeními Bluetooth latence.

Pokud porovnáme naměřenou latenci přenosu dat zařízení RingOne s výsledky latencí Bluetooth myší, je jasné, že toto zařízení není vhodné pro hraní počítačových her, jelikož jeho latence je několikrát větší než u herních Bluetooth myší. Ovšem jako vstupní zařízení pro ovládání běžných programů má RingOne latenci téměř dostačující. Je také důležité uvědomit si, že pro testování celkové latence systému bylo využito nejhoršího možného případu, kdy odesílaná data mají maximální možnou velikost, jelikož se přeposílá informace o celkové orientaci. Pro měření bylo také využito monitoru s obnovovací frekvencí 60 Hz a výrobcem určenou odezvou, který mohl výsledek do značné míry zhoršit.

Testy měřící latenci dat odeslaných zařízením RingOne prokázaly, že její hodnota je přívětivá a zařízení je tedy možné použít například i jako vstupní zařízení počítače. Tuto aplikaci zařízení RingOne umožňje právě Spacenav ovladač a na základě uživatelského testovaní systému s využitím tohoto ovladače bylo zjištěno, že celková latence je opravdu dostačující pro běžné použití s CAD programy.

# Kapitola 6

# Závěr

Tato práce měla za cíl vytvořit univerzální systém pro měření dat o pohybu ve 3D prostoru. Výsledný systém se skládá z firmwaru pro zařízení RingOne a knihovny ROL, která umožňuje uživateli jednoduše komunikovat s RingOne zařízením. Pro komunikaci mezi těmito dvěma částmi byl navržen speciální protokol ROL, který umožňuje efektivně odesílat naměřená prostorová data a také konfigurovat měřící zařízení bez nutnosti aktualizace jeho firmwaru. Poslední součást práce je upravený ovladač Spacenavd, který umožňuje komunikovat se zařízením RingOne a také zjednodušuje prezentaci dat v různých CAD programech.

Vytvořený systém podporuje širokou škálu konfigurace měřícího zařízení RingOne, což umožňuje použití v mnoha různých aplikacích, nejen pro měření polohy prstu, ale také například měření orientace kamery využívané pro detekci laserového paprsku a určení pozice jeho zdroje nebo záznam letových dat kluzáku pro vyhodnocení efektivity pilotova řízení. Zařízení RingOne také podporuje indikaci stavu s pomocí zabudované RGB LED diody. Barvu této diody je možné změnit i z uživatelské aplikace, jelikož tuto funkci podporuje ROL protokol.

Knihovna ROL umožňuje jednoduchou komunikaci s RingOne zařízením a také intuitivní zpracovaní přijatých dat a možnost jejich exportu ve formátu CSV. Díky skutečnosti, že knihovna podporuje ROL protokol, je uživatel oproštěn od analýzy přijatých dat a může se tak soustředit pouze na jejich využití.

Testování naměřených prostorových hodnot prokázalo, že hodnoty do značné míry odpovídají realitě. Proto je toto zařízení možné využít ve většině běžných aplikací. Naměřená latence přenosu dat je také dle testů dostatečná.

Při úpravě zmíněného ovladače Spacenavd, jsem narazil na ne zcela intuitivní chování otáčení objektů, kde zobrazovaný objekt není otáčen zcela stejně jako měřící zařízení. Toto chováni je zapříčiněno pravděpodobně využitím Eulerových úhlů pro reprezentaci orientace a mohlo by být do určité míry potlačeno využitím například kvaternionů pro popis výsledné orientace zařízení. Jelikož ovšem i samotný ovladač používá ve svém jádru Eulerovy úhly, není možné toto chování potlačit úplně.

Dalším krokem ve vývoji tohoto systému by mohlo být přidání některých senzorů do zařízení RingOne, jako jsou například magnetometr nebo GPS modul. Magnetometr by umožnil přesnější měření otočení kolem osy Z, které je nyní vypočítáváno pouze z dat gyroskopu. Dalším senzorem pro rozšíření funkcionality by mohl být GPS modul, který by umožnil naměřená data o orientaci doplnit o souřadnice uskutečněného měření. Toto rozšíření by mohlo být vhodné například při měření letových dat. Další možností rozšíření by mohlo být například přidání modulu reálného času pro možnost spojení naměřených prostorových dat s aktuálním časem.

# Literatura


[1] *Volume 1. Performance Flight Testing. Chapter 13. Equations of Motion I.* U.S. Air Force Test Pilot School, 1993 [cit. 2023-04-29]. Dostupné z: https://apps.dtic.mil/sti/citations/ADA320205.

[2] *Universal Asynchronous Receiver/Transmitter* [online]. SPRUGP1. Keystone Electronics Corp., 2010 [cit. 2023-01-12]. Dostupné z: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf.

[3] *32-Kbit serial I²C bus EEPROM M24C32-F* [online]. DocID4578. STMicroelectronics N.V., 2017 [cit. 2023-01-12]. Rev. 30. Dostupné z: https://www.st.com/resource/en/datasheet/m24c32-f.pdf.

[4] *KTD2026/KTD2027* [online]. KTD2026-KTD2027 Full Datasheet. Kinetic Technologies, 2017 [cit. 2023-01-12]. Rev. 04h. Dostupné z: https://www.kinet-ic.com/ktd2026/.

[5] *LSM6DSL* [online]. DocID028475. STMicroelectronics N.V., 2017 [cit. 2023-01-12]. Rev. 7. Dostupné z: https://www.st.com/resource/en/datasheet/lsm6dsl.pdf.

[6] *RN4870/71* [online]. DS50002489C. Microchip Technology, 2017 [cit. 2023-01-12]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/50002489C.pdf.

[7] *STM32L452xx* [online]. DS11912. STMicroelectronics N.V., 2020 [cit. 2023-01-12]. Rev. 7. Dostupné z: https://www.st.com/resource/en/datasheet/stm32l452re.pdf.

[8] *Description of STM32L4/L4+ HAL and low-layer drivers.* UM1884. STMicroelectronics N.V., 2021 [cit. 2023-04-05]. Rev. 9. Dostupné z: https://www.st.com/resource/en/user_manual/um1884-description-of-stm32l4l4-hal-and-lowlayer-drivers-stmicroelectronics.pdf.

[9] Beard, R. W., Beard, R. W. a McLain, T. W. *Small unmanned aircraft: theory and practice.* Princeton: Princeton University Press, 2012. ISBN 0691149216.

[10] Bluetooth Special Interest Group. *Bluetooth Core Specification version 5.4* [online]. 2023 [cit. April 29, 2023]. Dostupné z: https://www.bluetooth.com/specifications/specs/core-specification-5-4/.

[11] Bonnet, V., Mazzà, C., McCamley, J. a Cappozzo, A. Use of weighted Fourier linear combiner filters to estimate lower trunk 3D orientation from gyroscope sensors data. *Journal of neuroengineering and rehabilitation.* LONDON: Springer Nature. 2013, sv. 10, č. 1, s. 29–29. ISSN 1743-0003.

[12] Corbet, J. *Linux device drivers.* 3rd ed. Sebastopol: O´Reilly, 2005. ISBN 0-596-00590-3.

[13] Dewald, K. SimpleBLE. *Github* [online]. [cit. 2023-04-23]. Dostupné z: https://github.com/OpenBluetoothToolbox/SimpleBLE.

[14] Edwards, L. A. R. W. *Embedded system design on a shoestring.* Vyd. 1. Boston: Newnes, 2003. ISBN 0-7506-7609-4.

[15] Esashi, M. a Ono, T. From MEMS to nanomachine. *Journal of Physics D: Applied Physics.* jun 2005, sv. 38, č. 13, s. R223, [cit. 2023-04-06]. DOI: 10.1088/0022-3727/38/13/R01. Dostupné z: https://dx.doi.org/10.1088/0022-3727/38/13/R01.

[16] Etzion, O. a Niblett, P. *Event Processing in Action.* Manning Publications, 2010. ISBN 9781935182214.

[17] Gad-el-Hak, M. E. *The MEMS handbook. [Volume 1], MEMS: introduction and fundamentals.* 2nd ed. Boca Raton: CRC Press, 2006. Mechanical engineering series. ISBN 0-8493-9137-7.

[18] Gad-el-Hak, M. E. *The MEMS handbook. [Volume 2], MEMS: design and fabrication.* 2nd ed. Boca Raton: CRC Press, 2006. Mechanical engineering series. ISBN 0-8493-9138-5.

[19] Gad-el-Hak, M. E. *The MEMS handbook. [Volume 3], MEMS: applications.* 2nd ed. Boca Raton: CRC Press, 2006. Mechanical engineering series. ISBN 0-8493-9139-3.

[20] Gerstlauer, A., Yu, H. a Gajski, D. D. RTOS Modeling for System Level Design. In: *Embedded Software for SoC.* Boston, MA: Springer US, s. 55–68. ISBN 1402075286.

[21] Inductiveload. *Spherical coordinate system* [online]. [cit. 2023-01-24]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/b/b3/Coord_LatLong.svg.

[22] Kim, P. *Kalman filter for beginners : with MATLAB examples.* Charleston: CreateSpace, 2011. ISBN 978-1-463648350.

[23] Korn, G. A. *Mathematical Handbook for Scientists and Engineers.* Mineola: Dover Publications, 2000. ISBN 0-486-41147-8.

[24] Lal, S. Chapter 15 - Bare-Metal Systems. In: *Real World Multicore Embedded Systems.* Elsevier Inc, 2013, s. 517–560. ISBN 0124160182.

[25] Leondes, C. *MEMS/NEMS : handbook techniques and applications. Volume 2, Fabrication techniquies.* New York: Springer, 2006. ISBN 0-387-24520-0.

[26] Malaník, P. RingOne. *Github* [online]. [cit. 2023-01-24]. Dostupné z: https://github.com/TheColonelYoung/RingOne.

[27] Mitchell, H. *Multi-Sensor Data Fusion: An Introduction.* 1. Aufl. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN 9783540714637.

[28] Mui, L. *X Window system administrators guide*. Vyd. 1. London: O´Reilly, 1992. ISBN 0-937175-83-8.

[29] Narkhede, P., Poddar, S., Walambe, R., Ghinea, G. a Kotecha, K. Cascaded Complementary Filter Architecture for Sensor Fusion in Attitude Estimation. *Sensors*. 2021, sv. 21, č. 6, [cit. 2023-01-15]. DOI: 10.3390/s21061937. ISSN 1424-8220. Dostupné z: https://www.mdpi.com/1424-8220/21/6/1937.

[30] Noviello, C. *Mastering STM32: A Step-by-step Guide to the Most Complete ARM Cortex-M Platform, Using a Free and Powerful Development Environment Based on Eclipse and GCC*. Vyd. 0.26. Leanpub, 2018.

[31] Patel, C. a McCluskey, P. Modeling and simulation of the MEMS vibratory gyroscope. In: *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. IEEE, 2012, s. 928–933. ISBN 9781424495337.

[32] Postel, J. *Transmission Control Protocol* [Internet Requests for Comments]. STD 7. RFC Editor, September 1981 [cit. 2023-04-15]. Dostupné z: http://www.rfc-editor.org/rfc/rfc793.txt.

[33] Project, B. *BlueZ* [online]. [cit. 2023-04-23]. Dostupné z: http://www.bluez.org/.

[34] Smith, W. S. *Digital signal processing : scientist and engineer´s guide*. Vyd. 1. California: California Technical Publishing, 1997. ISBN 0-9660176-3-3.

[35] Stevens, W. R. *Advanced programming in the UNIX environment*. 1st ed. Boston: Addison-Wesley, 1993. ISBN 0-201-56317-7.

[36] Stolfi, J. *Cartesian coordinate system* [online]. [cit. 2023-01-24]. Dostupné z: https://en.wikipedia.org/wiki/Cartesian_coordinate_system#/media/File:Coord_system_CA_0.svg.

[37] Stolfi, J. *Cylindrical coordinate system* [online]. [cit. 2023-01-24]. Dostupné z: https://en.wikipedia.org/wiki/Cylindrical_coordinate_system#/media/File:Coord_system_CY_1.svg.

[38] Tilli, M., Paulasto Krockel, M., Petzold, M., Theuss, H., Motooka, T. et al. *Handbook of silicon based mems materials and technologies*. Third edition. Amsterdam: Elsevier, 2020. Micro and nano technologies series. ISBN 978-0-12-817786-0.

[39] Townsend, K., Davidson, R. a Cufí, C. *Getting Started with Bluetooth Low Energy*. O'Reilly, 2014. ISBN 9781491949511.

[40] Tsiombikas, J. *Spacenav* [online]. [cit. 2023-01-24]. Dostupné z: https://spacenav.sourceforge.net/.

[41] Villegas, F. J. I., Cisneros, S. O., Ibarra, F. S., Panduro, J. J. R. a Domínguez, J. R. Design of capacitive MEMS transverse-comb accelerometers with test hardware. *Superficies y Vacío*. 2013, sv. 26, č. 1, [cit. 2023-04-13]. Dostupné z: https://superficiesyvacio.smctsm.org.mx/index.php/SyV/article/view/173.

[42] Vince, J. *Quaternions for Computer Graphics*. London: Springer London, Limited, 2021. ISBN 9781447175087.

[43] Vodden, G. *Our Mouse Control Tests: Click Latency* [online]. 2022 [cit. 2023-05-03]. Dostupné z: https://www.rtings.com/mouse/tests/control/latency.

[44] Wang, Y., Hou, J., Li, C., Wu, J., Jiang, Y. et al. Ultrafast Mode Reversal Coriolis Gyroscopes. *IEEE/ASME transactions on mechatronics*. New York: IEEE. 2022, sv. 27, č. 6, s. 1–12. ISSN 1083-4435.

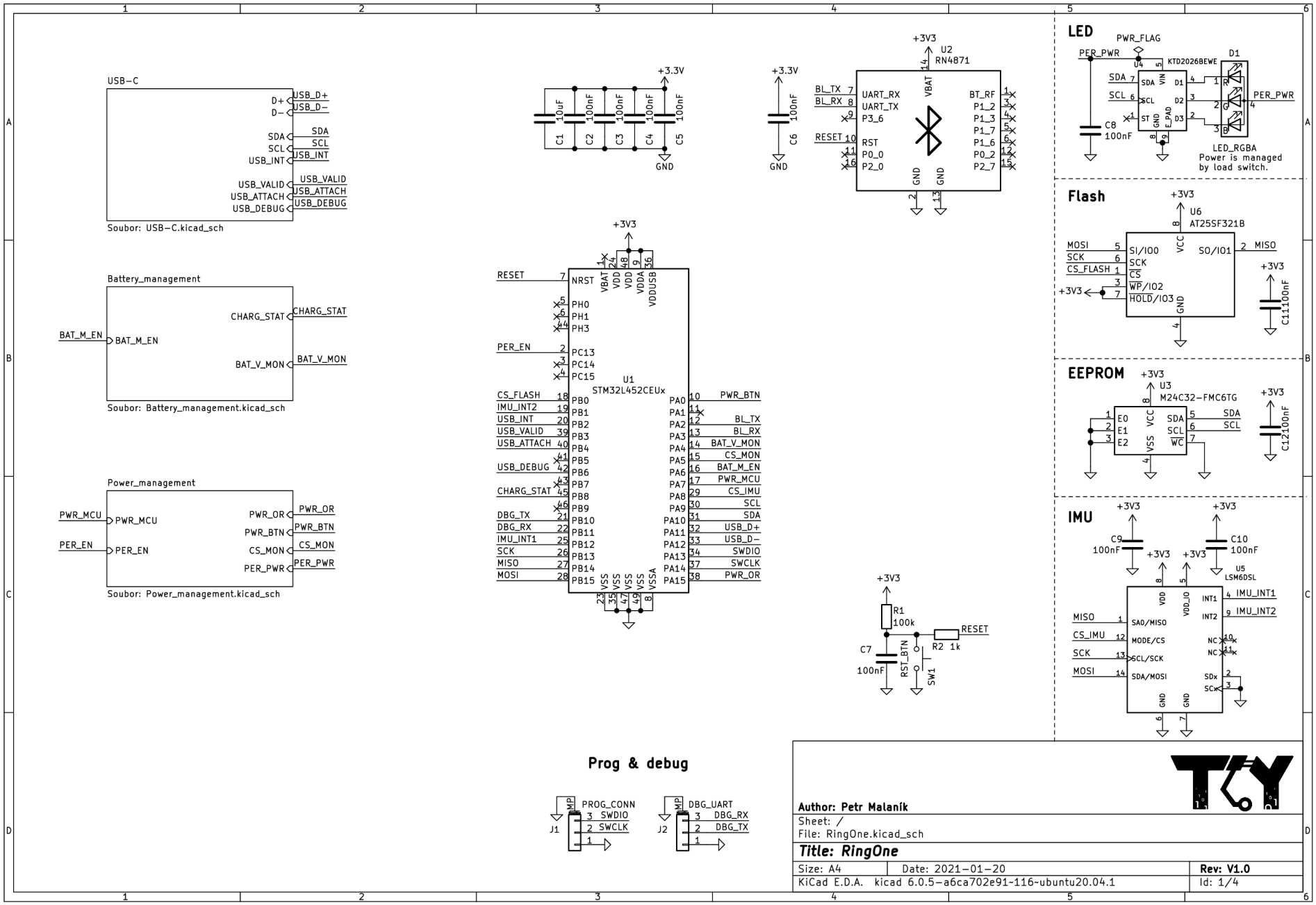
[45] Weisstein, E. W. *Euler Angles*. MathWorld–A Wolfram Web Resource [cit. 23-04-30]. Dostupné z: https://mathworld.wolfram.com/EulerAngles.html.

[46] Weisstein, E. W. *Rotation Formula*. MathWorld–A Wolfram Web Resource [cit. 23-04-30]. Dostupné z: https://mathworld.wolfram.com/RotationFormula.html.

[47] Yoo, S. a Jerraya, A. A. Introduction to Hardware Abstraction Layers for SoC. In: *Embedded Software for SoC*. Boston, MA: Springer US, 2003, s. 179–186. ISBN 1402075286.

# Příloha A

# Schéma zařízení RingOne

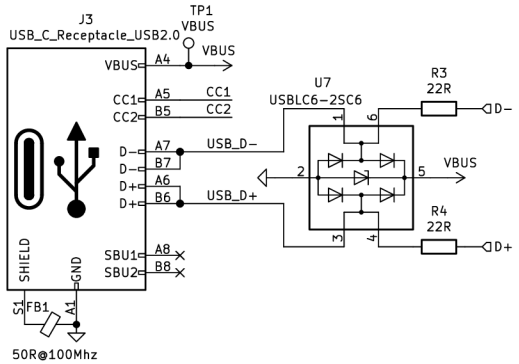


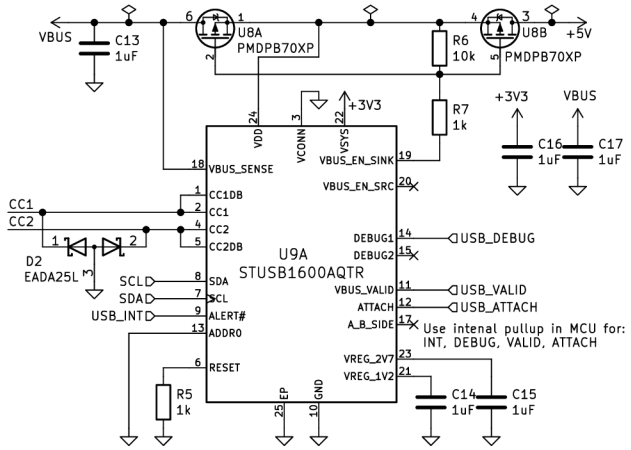

Author: Petr Malaník
Sheet: /
File: RingOne.kicad_sch

**Title: RingOne**

| Size: A4 | Date: 2021-01-20 | Rev: V1.0 |
|---|---|---|
| KiCad E.D.A. kicad 6.0.5-a6ca702e91-116~ubuntu20.04.1 | | Id: 1/4 |

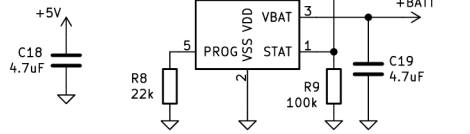
## USB type-C input



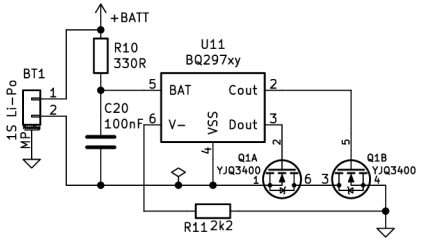

| | | |
|---|---|---|
| Sheet: /USB-C/ | | |
| File: USB-C.kicad_sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad 6.0.5-a6ca702e91~116~ubuntu20.04.1 | | Id: 4/4 |

## Charging

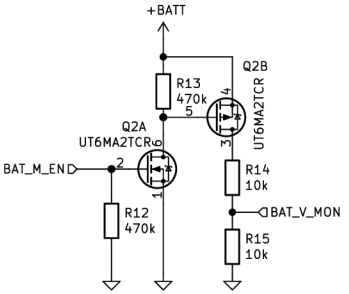



Charging current: 45 mA

## Battery protection

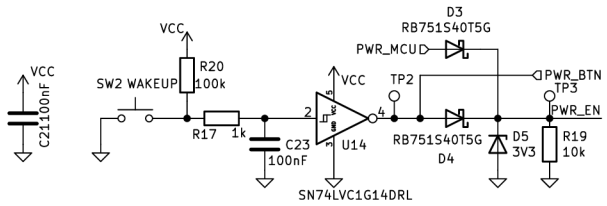



## Battery voltage measuring

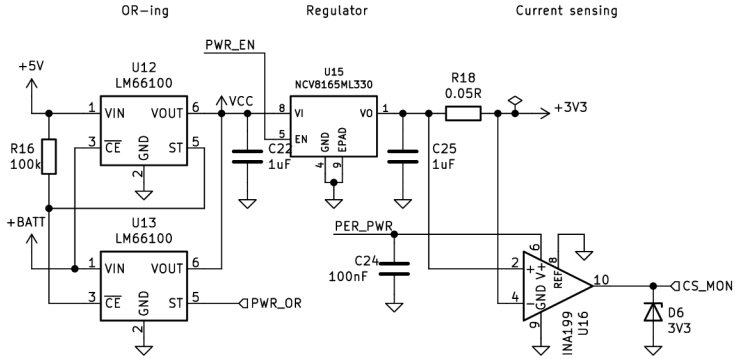



| | | |
|---|---|---|
| Sheet: /Battery_management/<br>File: Battery_management.kicad_sch | | |
| ***Title:*** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad 6.0.5-a6ca702e91~116~ubuntu20.04.1 | | Id: 5/4 |

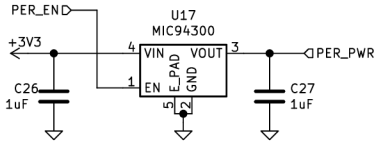
## Power enable button




## Power path & Current sensing




## Peripheral power switch




| | | |
|---|---|---|
| Sheet: /Power_management/<br>File: Power_management.kicad_sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad 6.0.5-a6ca702e91~116~ubuntu20.04.1 | | Id: 6/4 |

# Příloha B

# Příkazy ROL protokolu

**PING**

**Kód** 0x80

**Popis** Příkaz slouží pro testování spojení.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

**SET_OUTPUT_DEVICE**

**Kód** 0x81

**Popis** Příkaz slouží k nastavení výstupního rozhraní pro naměřená data.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| OUTPUT_NONE | 0x00 | Žádný výstup |
| OUTPUT_BT | 0x01 | Vydávání dat přes Bluetooth rozhraní |
| OUTPUT_UART | 0x02 | Vydávání dat přes UART debug rozhraní |
| OUTPUT_BOTH | 0x03 | Vydávání dat přes Bluetooth i UART |

**Omezení** Žádné

**SET_OUTPUT_TIMING**

**Kód** 0x82

**Popis** Příkaz slouží pro nastavení časování výstupu dat.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| SYNC | 0x00 | Data jsou vydávána při každém načtení dat z IMU |
| ASYNC | 0x01 | Data jsou vydávána periodicky. |

**Omezení** Žádné

### SET_DATA_SOURCE

**Kód** 0x83

**Popis** Příkaz pro výběr senzorů, ze kterých budou čtena data.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| NONE | 0x00 | Žádná data nejsou čtena |
| SOURCE_GYRO | 0x01 | Data jsou čtena pouze z gyroskopu. |
| SOURCE_ACCEL | 0x02 | Data jsou čtena pouze z akcelerometru. |
| SOURCE_BOTH | 0x03 | Data jsou čtena z gyroskopu i akcelerometru. |

**Omezení** Žádné

### SET_GYRO_CONFIG

**Kód** 0x84

**Popis** Příkaz slouží pro nastavení konfigurace gyroskopu.

**Délka argumentů** 2 bajty

**Argumenty** První bajt obsahuje konfiguraci frekvence a druhý bajt obsahuje konfiguraci senzitivity.

| Povolené hodnoty prvního bajtu | | |
|---|---|---|
| Název | Hodnota | Popis |
| FREQ___OFF | 0x00 | Žádná data nejsou měřena. |
| FREQ___12_HZ | 0x01 | |
| FREQ___52_HZ | 0x02 | |
| FREQ___208_HZ | 0x03 | |
| FREQ___416_HZ | 0x04 | |
| FREQ___833_HZ | 0x05 | |

| Povolené hodnoty druhého bajtu | | |
|---|---|---|
| Název | Hodnota | Popis |
| SENS_G___250DPS | 0x00 | |
| SENS_G___500DPS | 0x01 | |
| SENS_G___1000DPS | 0x02 | |
| SENS_G___2000DPS | 0x03 | |

**Omezení** Žádné

### SET_ACCEL_CONFIG

**Kód** 0x85

**Popis** Příkaz slouží pro nastavení konfigurace akcelerometru.

**Délka argumentů** 2 bajty

**Argumenty** První bajt obsahuje konfiguraci frekvence a druhý bajt obsahuje konfiguraci senzitivity.

| Povolené hodnoty prvního bajtu | | |
|---|---|---|
| Název | Hodnota | Popis |
| FREQ___OFF | 0x00 | Žádná data nejsou měřena. |
| FREQ___12_HZ | 0x01 | |
| FREQ___52_HZ | 0x02 | |
| FREQ___208_HZ | 0x03 | |
| FREQ___416_HZ | 0x04 | |
| FREQ___833_HZ | 0x05 | |

| Povolené hodnoty druhého bajtu | | |
|---|---|---|
| Název | Hodnota | Popis |
| SENS_XL___2G | 0x00 | |
| SENS_XL___4G | 0x01 | |
| SENS_XL___8G | 0x02 | |
| SENS_XL___16G | 0x03 | |

**Omezení** Žádné

### SET_DATA_PROCESSING

**Kód** 0x86

**Popis** Příkaz slouží pro nastavení předzpracování dat.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| PROCESSING_RAW_DATA | 0x00 | |
| PROCESSING_INTO_BASE_UNITS | 0x01 | |
| PROCESSING_POSITION | 0x02 | |

**Omezení** Pokud je `OUTPUT_TIMING` nastaven na hodnotu `ASYNC`, tento příkaz musí mít hodnotu jedině `PROCESSING_POSITION`.

### SET_DATA_FILTER

**Kód** 0x87

**Popis** Tento příkaz umožňuje vybrat filtr, který bude aplikován na naměřená data před odesláním.

**Délka argumentů** 1 – 2 bajty

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| FILTER_NONE | 0x00 | |
| FILTER_FUSE | 0x01 | |
| FILTER_MEDIAN | 0x02 | |

**Omezení** Hodnota `FILTER_NONE` může být použita pouze pokud je `DATA_PROCESSING` nastaven na hodnotu `PROCESSING_RAW_DATA` nebo `PROCESSING_INTO_BASE_UNITS`.

Hodnota `FILTER_MEDIAN` může být použita pouze pokud je `DATA_PROCESSING` nastaven na hodnotu `PROCESSING_RAW_DATA` nebo `PROCESSING_INTO_BASE_UNITS`.

Hodnota `FILTER_FUSE` může být použita pouze pokud je `DATA_PROCESSING` nastaven na hodnotu `PROCESSING_POSITION`.

Hodnota `FILTER_MEDIAN` vyžaduje jeden bajt argumentu navíc. Tento bajt reprezentuje počet hodnot, ze kterých je medián vypočítáván.

### SET_RUN_AT_STARTUP

**Kód** 0x88

**Popis** Tento příkaz umožňuje nastavit, aby se po inicializaci zařízení automaticky spustilo měření.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| OFF | 0x00 | Zařízení nezačne měřit po inicializaci. |
| ON | 0x01 | Zařízení začne měřit po inicializaci. |

**Omezení** Aby toto nastavení bylo aplikováno, je nutné následně uložit konfiguraci zařízení příkazem `SAVE_CONFIG_TO_EEPROM`.

### SET_ASYNC_PERIOD

**Kód** 0x89

**Popis** Tento příkaz umožňuje nastavit periodu pro asynchronní výstup dat.

**Délka argumentů** 2 bajty

**Argumenty** `uint16_t` hodnota reprezentující zvolenou periodu v milisekundách.

**Omezení** Žádné

**SET_DEFAULT_CONFIG**

**Kód** 0x8A

**Popis** Nastaví konfiguraci zařízení na výchozí.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

**SET_DRIVER_CONFIG**

**Kód** 0x8B

**Popis** Nastaví konfiguraci zařízení na konfiguraci přednastavenou pro Spacenav driver.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

**SET_IMU_OFFSETS**

**Kód** 0x8C

**Popis** Příkaz pro nastavení ofsetů pro specifický senzor.

**Délka argumentů** 7 bajtů

**Argumenty** První bajt slouží k výběru senzoru. Následujících 6 bajtů reprezentuje tři `int16_t` hodnoty ofsetů jednotlivých os popořadě pro osu X, Y a Z.

| Povolené hodnoty prvního bajtu | | |
|---|---|---|
| Název | Hodnota | Popis |
| GYRO | 0x00 | Nastavení ofsetů gyroskopu |
| ACCEL | 0x01 | Nastavení ofsetů akcelerometru |

**Omezení** Žádné.

**SAVE_IMU_OFFSETS_TO_EEPROM**

**Kód** 0x8D

**Popis** Příkaz pro uložení nastavených ofsetů do EEPROM paměti.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

**LOAD_IMU_OFFSETS_FROM_EEPROM**

**Kód** 0x8E

**Popis** Příkaz pro nahrání nastavených ofsetů z EEPROM paměti.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Offsety jsou z EEPROM načítány automaticky při inicializaci, není tedy nutné tento příkaz používat při každém měření.

**SAVE_CONFIG_TO_EEPROM**

**Kód** 0x8F

**Popis** Příkaz pro uložení konfigurace zařízení do EEPROM paměti.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

**LOAD_CONFIG_FROM_EEPROM**

**Kód** 0x90

**Popis** Příkaz pro nahrání konfigurace zařízení z EEPROM paměti.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Konfigurace je z EEPROM načítána automaticky při inicializaci, není tedy nutné tento příkaz používat při každém spuštění.

**ENABLE_DOUBLE_TAP**

**Kód** 0x91

**Popis** Tento příkaz umožňuje nastavit zaznamenávání dojitého poklepání.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| OFF | 0x00 | Zařízení nebude zaznamenávat dvojité poklepání. |
| ON | 0x01 | Zařízení bude zaznamenávat dvojité poklepání. |

**Omezení** Aby mohlo být dvojité poklepání zaznamenáváno, `SET_DATA_SOURCE` musí být nastaveno na hodnotu `SOURCE_ACCEL` nebo `SOURCE_BOTH` a frekvence akcelerometru musí být nastavena minimálně na 208 Hz.

### CHANGE_LED_COLOR

**Kód** 0x92

**Popis** Příkaz sloužící pro změnu barvy LED diody.

**Délka argumentů** 1 bajt

**Argumenty**

| Název | Hodnota | Popis |
|---|---|---|
| NONE | 0x00 | |
| RED | 0x01 | |
| GREEN | 0x02 | |
| BLUE | 0x03 | |
| WHITE | 0x04 | |
| CYAN | 0x05 | |
| MAGENTA | 0x06 | |
| YELLOW | 0x07 | |

**Omezení** Žádné

### START_MEASURING

**Kód** 0x93

**Popis** Příkaz pro zapnutí měření prostorových dat.

**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

### STOP_MEASURING

**Kód** 0x94

**Popis** Příkaz pro vypnutí měření prostorových dat.

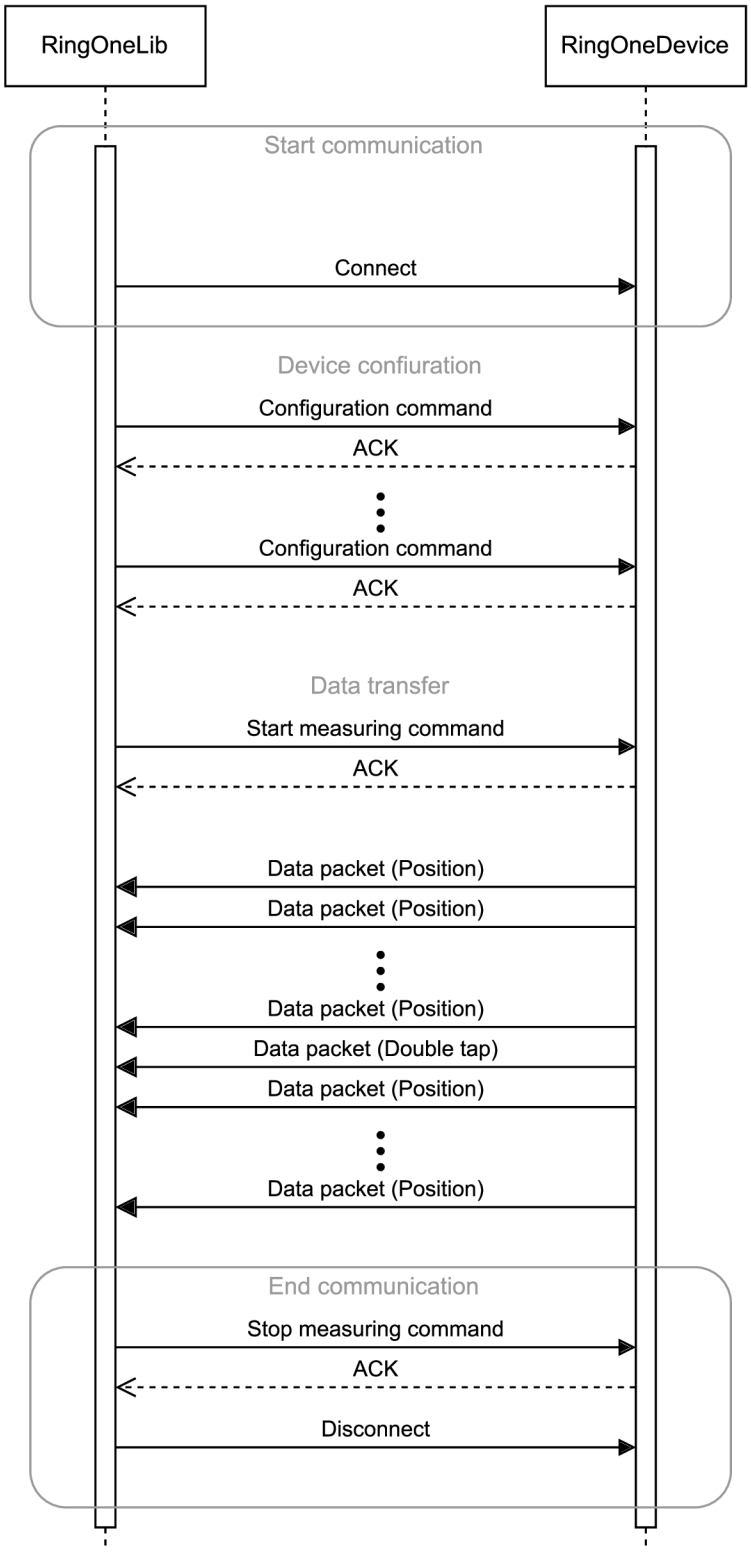
**Délka argumentů** 0 bajtů

**Argumenty** Žádné

**Omezení** Žádné

# Příloha C

# Sekvenční diagram použití ROL protokolu




Obrázek C.1: Sekvenční diagram komunikace mezi zařízením RingOne a knihovnou s použitím ROL protokolu.

# Příloha D

# Povolené hodnoty prvků konfigurační struktury zařízení RingOne

| Název prvku | Název hodnoty | Hodnota | Poznámka |
|---|---|---|---|
| output_device | OUTPUT_NONE | 0 | |
| | OUTPUT_BT | 1 | |
| | OUTPUT_UART | 2 | |
| | OUTPUT_ALL | 3 | |
| output_timing | SYNC | 0 | |
| | ASYNC | 1 | |
| data_source | SOURCE_NONE | 0 | |
| | SOURCE_GYRO | 1 | |
| | SOURCE_ACCEL | 2 | |
| | SOURCE_ALL | 3 | |
| gyro_config – frequency | CFG_FREQ___OFF | 0 | |
| | CFG_FREQ___12_HZ | 1 | |
| | CFG_FREQ___52_HZ | 2 | |
| | CFG_FREQ___208_HZ | 3 | |
| | CFG_FREQ___416_HZ | 4 | |
| | CFG_FREQ___833_HZ | 5 | |
| gyro_config – sensitivity | CFG_SENS_G___250DPS | 0 | |
| | CFG_SENS_G___500DPS | 1 | |
| | CFG_SENS_G___1000DPS | 2 | |
| | CFG_SENS_G___2000DPS | 3 | |
| accel_config – frequency | CFG_FREQ___OFF | 0 | |
| | CFG_FREQ___12_HZ | 1 | |
| | CFG_FREQ___52_HZ | 2 | |
| | CFG_FREQ___208_HZ | 3 | |
| | CFG_FREQ___416_HZ | 4 | |
| | CFG_FREQ___833_HZ | 5 | |

Tabulka D.1: Povolené hodnoty jednotlivých prvků konfigurační struktury. Část 1/2.

| Název prvku | Název hodnoty | Hodnota | Poznámka |
|---|---|---|---|
| accel_config – sensitivity | CFG_SENS_XL___2G | 0 | |
| | CFG_SENS_XL___4G | 1 | |
| | CFG_SENS_XL___8G | 2 | |
| | CFG_SENS_XL___16G | 3 | |
| data_processing | PROCESSING_RAW_DATA | 0 | Pouze pokud output_timing == SYNC |
| | PROCESSING_INTO _BASE_UNITS | 1 | Pouze pokud output_timing == SYNC |
| | PROCESSING_POSITION | 2 | |
| data_filter | FILTER_NONE | 0 | |
| | FILTER_FUSE | 1 | |
| | FILTER_MEDIAN | 2 | Pouze pokud output_timing == SYNC |
| async_period | | 1 – 65535 | Perioda asynchroinního odesílání dat v milisekundách. |
| run_at_startup | FALSE | 0 | Měření je nutné zapnout explicitně. |
| | TRUE | 1 | Měření se automaticky spustí po inicializaci zařízení. |
| double_tap_enabled | FALSE | 0 | Zaznamenávání dvojitého poklepání není zaplé. |
| | TRUE | 1 | Zaznamenávání dvojitého poklepání je zaplé. |
| filter_data | | -2147483647 – 2147483647 | Dodatečná data pro vybrané filtry. |

Tabulka D.2: Povolené hodnoty jednotlivých prvků konfigurační struktury. Část 2/2.

# Příloha E

# Obsah přiložené paměťové karty

- **/xhegrm00-pisemna_zprava.pdf** – tento dokument ve formátu PDF,
- **/text/*** – zdrojové soubory textu této práce,
- **/code/README.md** – hlavní Readme soubor, obsahující informace o této práci,
- **/code/Firmware/*** – zdrojové soubory firmwaru zařízení RingOne,
- **/code/Library/ROL_c/*** – zdrojové soubory knihovny ROL,
- **/code/Driver/*** – zdrojové soubory upraveného Spacenavd ovladače.