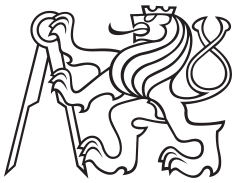
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Konstrukce nositelného zařízení pro měření počtu kroků

Wolfgang Mildner

Vedoucí práce: Ing. Alexandr Laposa, Ph.D.
Studijní program: Elektronika a komunikace
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mildner** Jméno: **Wolfgang** Osobní číslo: **499159**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Konstrukce nositelného zařízení pro měření počtu kroků

Název bakalářské práce anglicky:

Design of a Wearable Step Counting Device

Pokyny pro vypracování:

1. Prostudujte základní metody a technická řešení pro měření počtu kroků během chůze pomocí nositelné elektroniky.
2. Navrhněte a realizujte nositelné zařízení umožňující měření počtu kroků. Porovnejte různé akcelerometry, vzorkovací frekvence, algoritmy pro detekci kroků a jejich přesnost v určení počtu kroků, výpočetní náročnost a spotřebu energie.
3. Navrhněte a vytvořte laboratorní úlohu, která bude použitelná v rámci výuky.
4. Ověřte realizované řešení a zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

[1] M. Susi, V. Renaudin, and G. Lachapelle, “Motion mode recognition and step detection algorithms for mobile phone users,” Sensors (Switzerland), vol. 13, no. 2, pp. 1539–1562, Jan. 2013, doi: 10.3390/s130201539.
[2] P. Sadhukhan et al., “IRT-SD-SLE: An Improved Real-Time Step Detection and Step Length Estimation Using Smartphone Accelerometer,” IEEE Sens. J., vol. 23, no. 24, pp. 30858–30868, Dec. 2023, doi: 10.1109/JSEN.2023.3330097.
[3] X. Kang, B. Huang, R. Yang, and G. Qi, “Accurately counting steps of the pedestrian with varying walking speeds,” in Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, 2018, pp. 679–686, doi: 10.1109/SmartWorld.2018.00134.
[4] T. T. Pham and Y. S. Suh, “Walking Step Length Estimation Using Waist-Mounted Inertial Sensors with Known Total Walking Distance,” IEEE Access, vol. 9, pp. 85476–85487, 2021, doi: 10.1109/ACCESS.2021.3087721.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Alexandr Laposa, Ph.D. katedra mikroelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **08.02.2024** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Alexandr Laposa, Ph.D.	prof. Ing. Pavel Hazdra, CSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu své práce Ing. Alexandru Laposovi, Ph.D. za vstřícnost a konstruktivní připomínky k mému projektu. Také bych chtěl poděkovat své rodině za psychyckou a morální podporu.

Prohlášení

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

V Praze, dne 24. května 2024

Abstrakt

Bakalářská práce se zabývá metodami detekce kroků s použitím běžně dostupných akcelerometrů. Práce se zaměřuje na zpracování signálů z akcelerometrů a použití jeho matematických parametrů pro určení počtu kroků. Cílem bylo vytvořit aplikaci v Matlabu, která uživatele provede určením těchto parametrů a pomůže mu pochopit použití akcelerometrů. Součástí také bylo vytvoření zařízení, které je schopno zrychlení měřit a odesílat jej do této aplikace.

Klíčová slova: zrychlení, MEMS, ESP32, MATLAB, detekce chůze, senzory pohybu

Vedoucí práce: Ing. Alexandr Laposa, Ph.D.

Abstract

The bachelor thesis deals with step detection methods using commonly available accelerometers. The thesis focuses on the processing of accelerometer signals and the use of its mathematical parameters to determine the number of steps. The goal was to create an application in Matlab that guides the user through the determination of these parameters and helps the user understand the use of accelerometers. It also included creating a device that is capable of measuring acceleration and sending it to this application.

Keywords: acceleration, MEMS, ESP32, MATLAB, step detection, motion senzor

Title translation: Design of wearable step counting device

Obsah

Obrázky

Tabulky

Úvod

Detekce chůze je běžnou součástí našich životů. Dnes již každý člověk má na svém zápěstí nebo v kapse senzor, který je schopen měřit jeho zrychlení. Tato data se pak používají pro měření jeho fyzické aktivity případně je lze použít k detekci různých anomálií. Zaměřím se na to, jaké senzory se běžně používají. Jaké jsou jejich principy měření a metody výroby.

V této práci se snažím pochopit jakými způsoby lze ze signálu zrychlení získat užitečná data. Získání užitečných dat proběhne jednoduchou filtrací. Dále ze signálu budu získávat parametry jako je energie, variance či frekveční charakteristika. Tyto parametry posléze použiji pro základní detekci chůze.

Na závěr vytvořím aplikaci v Matlabu, která bude uživatelům prezentovat výše zmíněné parametry zrychlení. A kde si budou moci vyzkoušet právě takové parametry nastavit a sami si je ověřit.

Kapitola 1
Teoretický rozbor

V této kapitole nejprve popíši základní fyzikální vztahy související se zrychlením a jeho určením. Následně bych se zaměřil na zjednodušenou konstrukci akcelerometrů a vysvětlil na ní základní principy. Poté bych chtěl předvést jejich nejpoužívanější realizaci ve formě kapacitních senzorů. Nakonec přiblížím různé metody detekce chůze tj. metody detekce kroků.

1.1 Teorie zrychlení

Těleso může být ve dvou stavech – v klidu nebo v pohybu. Pokud hovoříme o pohybu, musíme zvážit úhel pohledu, jelikož vůči jedné soustavě může být těleso stacionární a oproti jiné v pohybu. V našem případě si zvolíme jako referenční soustavu povrch Země a veškeré pohyby budeme posuzovat vůči němu.

Lineární pohyb

Pro jednoduchost bych začal s nejjednodušším pohybem tělesa – přímočarým pohybem. Těleso se pohybuje po přímce vůči referenční soustavě. Pokud jako pozorovatel zaznamenáme, že se mění pozice tělesa, říkáme, že se těleso pohybuje s určitou rychlostí [1]. Rychlost podél přímé dráhy tedy můžeme zapsat jako změnu dráhy za změnu času

$$\bar{v} = \frac{\Delta x}{\Delta t}, \tag{1.1}$$

kde Δx je vzdálenost mezi dvěma referenčními body a Δt je doba, kterou za kterou tělesa vzdálenost mezi body urazí.

Okamžitou rychlost v daném čase definujeme jako

$$v = \frac{\mathrm{d}x}{\mathrm{d}t}. \tag{1.2}$$

Pokud pohyb není rovnoměrný, rychlost se mění v průběhu času, což způsobuje buď zrychlení nebo zpomalení. Podle Newtonova druhého zákona tato změna vyžaduje použití síly (v autě zmáčknu plyn nebo brzdu). Okamžité zrychlení

pak můžeme zapsat jako derivaci rychlosti respektive druhou derivaci polohy

$$a = \frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\mathrm{d}^2x}{\mathrm{d}t^2} \tag{1.3}$$

Pro zrychlení se používá jednotka $\mathrm{m/s^2}$. Ovšem u akcelerometrů se spíše používá g, které odkazuje na hodnotu tíhového zrychlení na povrchu Země ($g = 9,81\ \mathrm{m/s^2}$). Hlavně se s ním lze setkat v datasheetech senzorů, kde se v něm uvádí jeho rozsah nebo například citlivost měření která je ve $\mathrm{V/g^2}$.

1.2 Inerciální lineární senzory

Akcelerometry jsou používány pro měření zrychlení, které je součtem působení vnějších sil, včetně gravitačních. Zatímco tíhové zrychlení je běžně konstantní síla směřující do středu Země, ostatní síly jsou zpravidla proměnné v čase, velikosti i ve směru. Běžný akcelerometr by měl být citlivý na různé formy zrychlení – od konstantních a pomalých změn až po velké výchylky a vibrace.

Podle Newtonova druhého zákona definujeme zrychlení ve vektorech jako

$$\vec{F} = m \cdot \vec{a} \qquad \Rightarrow \qquad \vec{a} = \frac{\vec{F}}{m}, \tag{1.4}$$

kde $\vec{F}$ je síla působící na těleso o hmotnosti m. Vyplývá z něj, že hmota, síla a zrychlení jsou propojeny. Směr zrychlení a síly je totožný.

Rovnice 1.4 nastiňuje, že pro měření zrychlení je potřeba zajistit nějakou známou hmotnost m a měřit velikost síly, jíž tato hmotnost bude působit na senzor.

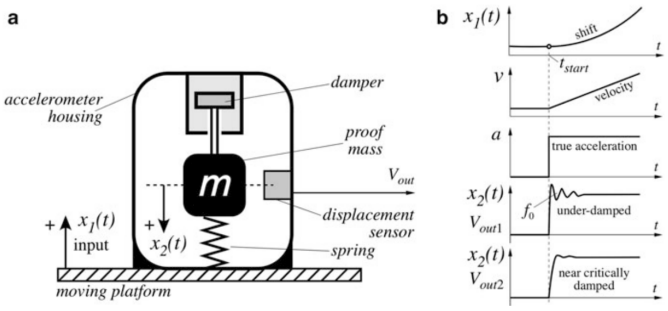
Model akcelerometru

Princip akcelerometru představím na jednoduchém modelu. Jedná se o zařízení s jedním stupněm volnosti (bude měřit zrychlení pouze podél jedné osy), které bude obsahovat *setrvačnou* respektive *seismickou hmotu*, odpružení této hmoty, tlumení kmitů seismické hmoty a také samotný senzor polohy. Na obrázku 1.1 je znázorněno rozmístění jednotlivých částí senzoru.

Setrvačná hmota (m) je umístěna na pružině (k), která umožňuje hmotě se pohybovat nahoru a dolů oproti základně (M). Pevnost pružiny ovlivňuje rozsah senzoru – čím je pružina tvrdší, tím má senzor větší rozsah, ale na úkor citlivosti. Jako základna je u senzoru použit podkladový materiál na kterém jsou naneseny vyhodnocovací a komunikační obvody. Z makroskopického hlediska za ní můžeme považovat destičku na níž je senzor připájen. Musí být pevně spojena s měřeným objektem, aby bylo měření vypovídající.

K hmotě jsou připojeny dvě další komponenty – tlumič (b) a senzor vychýlení. Tlumič zpomaluje pohyb hmoty a senzor určuje její polohu vůči neutrální pozici (bez zrychlení). Tlumení může být realizováno přímo pružinou, která bude sama o sobě tlumit rozkmity hmoty nebo můžeme mít jako tlumič nějaké prostředí (vzduch případně kapalinu). Tlumení zajišťuje, že setrvačná

hmota nebude kmitat moc dlouho, čímž by mohla nežádoucím způsobem zarušit následující měření. Je potřeba si však dát pozor, aby hmota nebyla naopak přetlumená, což by také negativně ovlivnilo měření. Základna, která je



Obrázek 1.1: Koncept lineárního mechanického akcelerometru (a) s průběhy signálu (b)

spojena s pouzdrem akcelerometru může být v klidu nebo se může pohybovat (v našem případě) podél osy x. Pokud budeme předpokládat, že změna polohy bude mít tvar parabolické funkce (viz 1.1 část b). Celé zařízení zrychluje s rychlostí v, která se lineárně zvyšuje zatímco zrychlení a je skoková funkce, jež se snažíme změřit. Na začátku pohybu se kvůli setvačnosti seismická hmota snaží zůstat na původním místě a tak vyvine sílu F na pružinu, kterou stlačí na vzdálenost $\Delta x = x_2 - x_1$. Pružina je charakterizována tuhostí k a snaží se působit proti síle F. Díky tomu můžeme odvodit následující rovnici

$$F = ma = k\Delta x = k(x_2 - x_1), \tag{1.5}$$

ze které můžeme odvodit vztah pro vychýlení seismické hmoty jako

$$x_2 - x_1 = \frac{m}{k}a. \tag{1.6}$$

1.3 Konstrukce akcelerometrů

Technologie MEMS

MEMS (Micro-Electro Mechanical System) je označení pro technologii ve které se vyrábějí mikroskopická zařízení jež propojují mechanickou a elektrickou složku na jednom čipu. Zatímco elektronické obvody, ať už analogové nebo digitální, jsou vyráběny technologiemi CMOS, BiPolar nebo BiCMOS, mikromechanické části jsou vytvářeny mikroobráběním [2]. Mikroobráběcí techniky odleptávají vybrané oblasti křemíkového wafferu, případně přidají nové strukturální vrstvy pro vytvoření potřebných mechanických a elektromechanických zařízení.

Mikromechanické senzory a aktuátory umožňují mikrosystému vnímat a ovládat okolní prostředí a zpracování signálů řeší okolní integrované obvody. MEMS umožňují použití stejných technologií, které byly vyvinuty pro výrobu elektronických obvodů. Díky snadné škálovatelnosti výroby MEMS zařízení je

lze vyrábět, stejně jako běžné čipy, s velmi vysokou přesností a za relativně nízkou cenu.

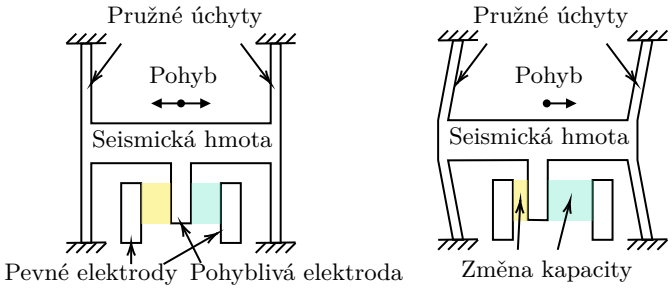
Kapacitní akcelerometry

Akcelerometr je ve své podstatě senzor, který monitoruje pohyby seismické hmoty a ty nějakým způsobem převádí na elektrický signál. Těchto převodníků je několik druhů. Jako příklad si můžeme uvést třeba piezorezistivní nebo termální akcelerometry [1]. V mé realizaci budu výhradně používat kapacitní akcelerometry jelikož to jsou osvědčené a spolehlivé prvky, které lze miniaturizovat s vysokou přesností ale zároveň zachovat nízkou cenu.

Kapacitní převodník se skládá alespoň ze dvou elektrod, kde první je umístěna na pevném pouzdře akcelerometru zatímco druhá je připevněna na seismickou hmotu, jež se hýbe uvnitř pouzdra. Tyto dvě vodivé plochy tvoří kondenzátor jehož kapacita je funkcí překrývajících se ploch (S) a jejich vzájemné vzdálenosti (d).

$$C = \varepsilon_0 \varepsilon_r \frac{S}{d} \tag{1.7}$$

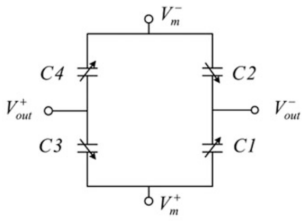
Design senzoru určuje, jestli bude záviset na změně plochy nebo vzdálenosti mezi elektrodami kondenzátoru. Výchylka seismické hmoty se pohybuje v jednotkách mikrometrů, což vyžaduje spolehlivou kompenzaci driftu a dalších nežádoucích interferencí. Proto se spíše volí rozdílové zapojení, kdy se vytvoří druhý kondenzátor na stejné MEMS struktuře. Kapacita druhého kondenzátoru musí být blízká prvnímu a při zrychlení musí být fázový posun signálu 180°. Potom může být zrychlení určeno jako rozdíl mezi těmito dvěma kondenzátory.



Obrázek 1.2: Ukázka principu kapacitního akcelerometru

Na obrázku 1.3 je ukázka principu rozdílového zapojení kondenzátoru. V klidu na setrvačnou hmotu nepůsobí síla, můstek je v rovnovážném stavu a napětí na obou svorkách V_{out}^{+} a V_{out}^{-} je vyrovnané. Pokud začne senzor zrychlovat, setrvačná hmota se vychýlí spolu s kapacitami připojených kondenzátorů.

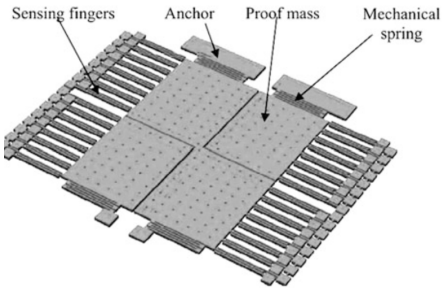
Výhodou tohoto zapojení je, že kapacita se mění na dvou kondenzátorech a díky tomu získáme dvakrát větší přesnost. Zároveň je odolný vlivům teploty protože budou ovlivněny všechny kondenzátory v zapojení čímž se negativní efekt teploty vyruší.



Obrázek 1.3: Rozdílové zapojení kondenzátorů

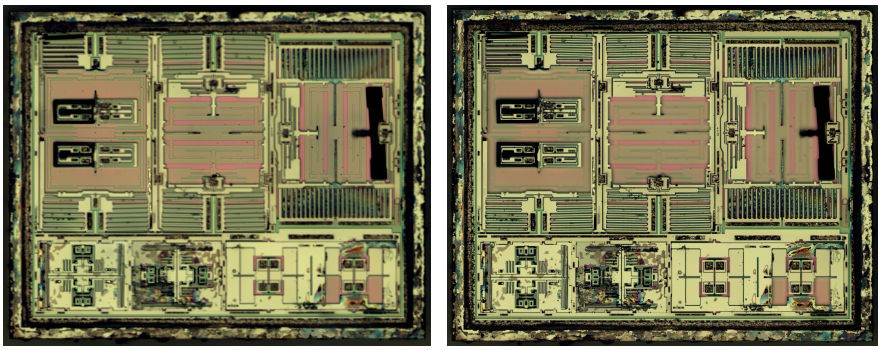
Realizace kapacitního akcelerometru

Na obrázku 1.4 je ukázka struktury kapacitního akcelerometru [1].



Obrázek 1.4: Ukázka relizace akcelerometru [1]

Reálné zapojení však bývá mnohem složitější. Na obrázku 1.5 je pohled na odkrytou strukturu akcelerometru MPU6050 se zaostřením na různé prvky. V horní části si lze povšimnout tří akcelerometrů, pro každou osu jeden. Ve spodní části se pak nacházejí gyroskopy pro detekci náklonu zařízení.



(a) : Zaostřeno na seismickou hmotu

(b) : Zaostření na odpružení a kondenzátory

Obrázek 1.5: Ukázka realizace senzoru MPU6050 [3]

1.4 1. Metoda detekce chůze

Jako první metodu detekce aplikuji [4], která ze signálu nejprve vypočítá energii, varianci a rychlou fourierovu transformaci (dále jen FFT). Tyto složky následně zpracuje v rozhodovacím stromě, kde určí do jaké kategorie signál spadá. V uvedené práci těchto kategorií mají několik, včetně rozpoznávání o jaký pohyb se jedná. Já se omezím pouze na tři základní kategorie:

- Statický - uživatel se takřka nehýbe, jediným výstupem z akcelerometru je tíhové zrychlení a šum
- Nepravidelný - uživatel se hýbe zařízením chaoticky, například ukazuje rukama nebo hledá věc v tašce
- Chůze/běh - uživatel provádí periodický pohyb rukou - můžeme předpokládat že chodí

V následujících kapitolách přiblížím zpracování signálu a výpočet složek které lze použít k rozpoznání chůze. Velikosti použitých oken a nastavení hodnot jednotlivých parametrů pak popíši v praktické části.

Předzpracování signálu

Pro zpracování signálu budu používat pouze velikost vektoru zrychlení. Vzhledem k tomu, že senzor není pevně uchycen k tělu, kde bychom mohli vybrat jednu osu podle které bychom signál zpracovávali, je zachování informace o směru zbytečné. Senzor bude uchycen na ruce, kde se neustále mění jeho orientace a měřené zrychlení. Využijeme tedy velikost zrychlení kterou spočítáme z jednotlivých složek s pomocí pythagorovy věty

$$s_{\mathrm{rms}}[n] = \sqrt{a_{\mathrm{x}}^2 + a_{\mathrm{y}}^2 + a_{\mathrm{z}}^2}. \tag{1.8}$$

Je potřeba podotknout, že přítomnost nenulové stejnosměrné složky bude negativně ovlivňovat frekvenční analýzu signálu a může skrýt některé charakteristiky signálu. Z toho důvodu ji následujícím způsobem odstraním

$$s_{\mathrm{rms0}}[n] = s_{\mathrm{rms}}[n] - \frac{1}{N}\sum_{l=0}^{N-1} s_{\mathrm{rms}}[n-l] \tag{1.9}$$

Druhý člen rovnice 1.9 vypočítává průměrnou hodnotu z okna signálu dlouhého N vzorků. N je velikost okna ze kterého průměrnou hodnotu počítáme. Čím delší bude toto okno, tím přesnější bude výpočet avšak na začátku měření bude déle trvat než se kompletně vyfiltruje stejnosměrná složka. Jedná se o horní propust provedenou metodou konečné impulzové odezvy (FIR) díky čemuž je stabilní a snadno aplikovatelná.

Energie signálu

Energie signálu nám pomůže rozlišit aktivity s nízkou a vysokou intenzitou. Energie je vypočítána podle vzorce

$$E_{\mathrm{s}} = \frac{1}{N} \sum_{l=0}^{N-1} s_{\mathrm{rms0}}^2 \left[n - l \right], \tag{1.10}$$

jež nejprve vypočítá kvadrát amlitudy daného vzorku a následně sečte N posledních vzorků. Z energie signálu lze velmi dobře rozlišit jestli se uživatel aktivně pohybuje nebo jen nečinně stojí. Je to primárně způsobeno umístěním měřicího zařízení na zápěstí, které se během stání prakticky nehýbe a naopak během chůze nebo běhu je výchylka veliká.

Rozptyl signálu

Dalším parametrem, jež budeme rozlišovat je rozptyl signálu. Jedná se o statistická hodnota, která je určena následujícím výpočtem:

$$\sigma_s^2 \left[n \right] = \frac{1}{N} \sum_{n=0}^{N-1} \left(s_{\mathrm{rms0}} \left[n \right] - \frac{1}{N} \sum_{n=0}^{N-1} s_{\mathrm{rms0}} \left[n \right] \right)^2 . \tag{1.11}$$

Je to součet kvadrátů odchylek od průměrné hodnoty signálu. Tento parametr nám umožní rozlišit nepravidelný pohyb, kdy rychle vzroste variance signálu.

Frekvenční analýza

Lidská chůze je periodická aktivita. Skládá se ze dvou kmitavých pohybů:

- tělo se pohybuje nahoru když dáváme nohy k sobě a dolů když je dáváme od sebe
- kmitavý pohyb rukou který nám pomáhá udržet během chůze rovnováhu

Tyto dva pohyby se projeví na výstupním signálu akcelerometru kde můžeme pozorovat periodicky opakující se nárůst a pokles zrychlení. Tento signál zanalyzujeme s pomocí Short Time Fourier Transform (STFT). Ta aplikuje FFT na krátké okno signálu a vypočítá amplitudy jeho frekvenčních složek. Využíváme STFT protože signál z akcelerometru bude velmi proměnný v čase, takže frekvenční analýza celého signálu by nebyla vypovídající. Avšak také předpokládáme, že signál bude v určitém krátkém okně neproměnný.

Velikost okna se zvolí jako mocnina 2, abychom mohli využít rychlého výpočtu FFT. To určuje rozlišení frekvenční charakteristiky – z čím více vzorků ji budeme počítat tím přesnější budeme mít údaj o frekvenci.

Na obrázku je spektrogram signálu který v první části zaznamenává uživatele který jen postává a v druhé části začne chodit. V první části je pouze šum, kdy žádná z frekvenčních složek signálu nevyniká nad ostatní. U chůze lze pozorovat vyniknutí frekvenčních složek okolo 2 Hz.

Pro analýzu signálu nebudeme pracovat s celým frekvenčním spektrem ale určíme tři frekvence na kterých se nachází lokální maxima amplitudy a ty budeme dále zpracovávat.

Vyhodnocení

Vyhodnocování signálu probíhá s pomocí rozhodovacího stromu. Nejprve se zjistí jestli se uživatel hýbe. To se určí podle variace a energie signálu které nesmí přesáhnou určitou hodnotu. Pokud ji přesáhnou rozhodování se přesune na následující blok kde se určí o jaký pohyb se jedná. Bude se rozhodovat podle variace a periodicity signálu. Variace musí opět spadat pod určitou hodnotu aby byl signál periodický a zvolila se tak chůze. Zároveň však musí být periodický. Tuto vlastnost určíme z rozptylu záznamu tří frekvencí s maximálními amplitudami, jež jsme získali při frekvenční analýze signálu. Opět, pokud je tento rozptyl nižší než daná hodnota, signál je dostatečně periodický na to, aby se jednalo o chůzi.

V uvedené práci ještě dále rozlišovali zda-li se jedná o kmitání rukou nebo jestli uživatel píše do mobilu nebo ho má v batohu. To je však nad rámec této práce kde se zabývám pouze rozpoznáním chůze oproti ostatním pohybům.

Výstupem této části je signál, který určuje v jakém módu se právě zařízení nachází:

- 0 – uživatel se nepohybuje
- 1 – nepravidelný pohyb
- 2 – uživatel chodí

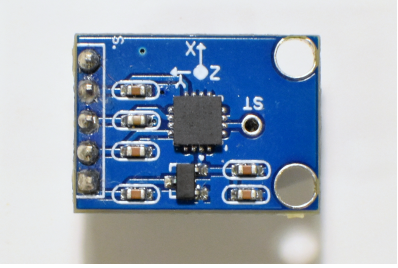
Detekce kroků

Detekce kroků se spustí až poté co je detekována chůze. Jako krok je detekováno lokální maximum signálu. Rozmit signálu při běžné chůzi je zhrube od 0.7 do 1.6 g. Střední hodnota se pohybuje kolem 1 g.

Na signálu je hledáno maximum s určitými vlastnostmi, které když jsou splněny tak je přičten další krok. Nejprve je potřeba nastavit minimální hodnotu pod kterou maximum nebudeme brát v potaz. Tímto odfiltruji falešná lokální maxima, která se nachází v minimu křivky a jež by způsobovala detekci vyššího počtu kroků. Lokální maximum musí splňovat další parametr a tím je prominence. Ta znamená, že maximum se vyvyšuje nad nejbližší sedlový bod alespoň určitou hodnotu. Tím zaručím, že nebude docházet k dvojím detekcím na stejném maximu. Pro své měření jsem používal hodnotu 0.5 g, což je zhruba půlka rozsahu zrychlení chůze. Detekci kroků ještě shrnu v následujícím pseudokódu.

```
prominence = 0.5
threshold = 1

if localMin > sample
    localMin = sample
if localMax < sample
    localMax = sample
if localMax - prominence > sample
    and localMax - localMIn > threshold
```



Obrázek 1.6: Ukázka směru os analogového senzoru

```
        steps++
        localMax = -Inf
        localMin = Inf
```

Sample je aktuální naměřený vzorek. Ten je porovnáván s hodnotami lokálního maxima a minima. Pokud je jeho hodnota větší v případě lokálního maxima nebo menší u minima je nový hodnota uložena do příslušné proměnné. Pokud je hodnota vzorku vzdálná od lokálního maxima víc než je definovaná prominence a zároveň rozdíl mezi lokálním maximem a minimem je dostatečná vzdálenost, pak je detekován krok. Hodnota lokálního maxima a minima je pak resetována na kladné a záporné nekonečno. Tento postup je spíše matematický, aby byl s jistotou načten další vzorek signálu. V reálném případě bych ji buď nastavil na maximální a minimální hodnotu používaného datového typu, nebo bych rovnou dosadil následující vzorek signálu.

1.5 2. Metoda detekce chůze

Tato metoda je používána v práci [5]. Je zde použita pro určení chybovosti různých metod detekce při různých vzorkovacích frekvencích.

Využívá přepočítání dat ze dvou os akcelerometru na úhel natočení. Tato data následně vyfiltruje dolní propustí a aplikuje metodu detekci kroků pokud signál přesáhne určitou hladinu.

Předzpracování signálu

Nejprve musím určit v jakých dvou osách se bude zrychlení nejvíce proměňovat. Ty určím z charakteristiky chůze a z architektury senzoru viz obrázek 1.6. Zařízení je umístěno rovnoběžně na rovinu zápěstí. V této rovině se také nacházejí osy senzoru X a Y. Tyto osy budou tedy hlavními detektory chůze.

Získání vertikálního zrychlení

Z vybraných os vypočítám úhel mezi zařízením a směrem tíhového zrychlení.

$$\phi = \arctan\left(\frac{a_x}{a_y}\right) \tag{1.12}$$

Následně zjistím velikost tohoto zrychlení, tím, že naprojektuji zrychlení v obou osách do vertikální roviny. Tím získám celkové vertikální zrychlení a_v.

$$a_v = a_x * \sin(\phi) + a_y * \cos(\phi) \tag{1.13}$$

Tímto jsem získal velikost zrychlení rovnoběžnou s tíhovým zrychlením. Toto vertikální zrychlení má výhodu v tom, že je v něm eliminován šum způsobený kmitáním více částí těla. Výsledné zrychlení je tedy z velké části tvořeno pouze tím, jak se tělo pohupuje nahoru a dolů během chůze.

Nalezení maxim

Pro určení, že daný pohyb je krok využijeme nejprve amplitudu signálu. Pokud má signál amplitudu nižší než jen určitá hranice, nebudeme signál dále zpracovávat a předpokládáme, že uživatel se nehýbe. V případě, že se uživatel začne pohybovat, signál přesáhne tuto hranici a můžeme pak začít počítat kroky. Kroky jsou počítány podobně jako u předchozí metody nalezením maxim. Když nalezneme lokální maximum signálu přípočteme další krok. demonstruji na následujícím pseudokódu.

```
threshold = 1
tol = 0.2

if localMax < sample
    localMax = sample
if localMax - tol > sample
    and localMax > threshold
        steps++
        localMax = -Inf
```

Kód nejpre načítá vzorky které jsou větší než lokální maximum, lokální maximum tak stále stoupá. Pokud však načítané vzorky klesnou pod určitou toleracni od lokálního maxima. Je přičten krok a hodnota maxima je zresetována.

Kapitola 2
Praktická část

V této kapitole se zaměřím na design měřicího zařízení. Nejprve vyberu senzory, které použiji pro měření zrychlení. Určím si jaké parametry pro mě byly klíčové a které nikoliv. Následně se zaměřím na výběr procesoru, který bude celé zařízení řídit. Proberu zvažované možnosti komunikace procesoru s počítačem a aplikací v Matlabu, která bude sloužit pro zobrazování dat. Na konec se zaměřím na naprogramování tohoto mikroprocesoru tak, aby byl schopen této komunikace a ukáži fungování navržené aplikace v Matlabu.

Cílem této části je vytvořit zařízení, kterým si budu moci ověřit metody detekce chůze. Zároveň chci aby výsledek mohl být použit při výuce a zvládnul odolat zvídavosti studentů. Vytvořím zařízení, které bude modulární – při poškození nebo nefunkčnosti nějaké části není problém ji vyměnit. Deska plošných spojů propojující tyto moduly by také měla být jednoduchá na výrobu pro případnou opakovatelnost měření. Vybíral jsem senzory a mikroprocesor, které jsou součástí modulu.

2.1 Senzory

Uvedu parametry které pro mě byly klíčové při výběru senzorů.

Rozhraní

Na výběr je hned několik běžně používaných rozhraní. Rozdělím je na dvě základní – digitální a analogové. V měřicí části porovnám digitalní senzor oproti analogovému. Alespoň jeden tedy musí mít analogový výstup, respektive tři analogové výstupy – pro každou osu jeden. Druhý pak musí mít možnost digitální komunikace. Na výběr je ze dvou nejběžnějších možností: I2C a SPI.

SPI je sběrnice pro synchronní sériovou komunikaci, založená na principu master–slave. Na sběrnici je přítomný řídicí obvod (master), ten kontroluje komunikaci na lince. Ostatní (slave) slouží pro příjem nebo odesílání dat. Zpravidla se jedná o senzory nebo displeje. Ke komunikaci využívá tři respektive čtyři vodiče.

- SDO – Serial Data Out (datový vstup)
- SDI – Serial Data In (datový výstup)

- SCK – Serial ClocK (hodinový signál)
- SS – Slave Select (výběr se kterým zařízením bude master komunikovat)

Každý slave obvod má výstup Slave Select pro výběr obvodu. Vstupy SS jednotlivých obvodů jsou samostatnými vodiči propojenými s řídícím obvodem. Každé další zařízení navíc blokuje jeden pin procesoru pro volbu SS, což je nevýhoda v aplikacích s více senzory. Naopak výhodou může být duplexní komunikace a její vysoká přenosová rychlost. Běžně se používá 15 MHz, rychlejší mikrokontrolery zvládají 25 MHz.

I2C je komunikační sběrnice vyvinutá firmou Philips pro komunikaci mezi integrovanými obvody. Sběrnice je typu multimaster – lze mít více řídících obvodů. Ostatní slave zařízení opět slouží pro příjem nebo pro odesílání dat. Komunikace probíhá po dvou vodičích.

- SDA - Datový vodiče
- SCL - Hodinový signál

Na tyto vodiče je potřeba připojit pull-up rezistory aby linka byla v klidovém stavu v logické jedničce. Výběr obvodu se kterým bude master komunikovat probíhá přes 7 bitovou (případně 10 bitovou) adresu, která je vysílána na začátku komunikace. Díky tomu lze připojit přes 120 zařízení na jednu linku. I2C komunikace je half-duplexní protože data přes SDA mohou proudit pouze jedním směrem. Nevýhodou je nižší rychlost hodinového signálu oproti SPI a tedy i nižší přenos dat. Běžně se používá kmitočet 100 kHz (standard mode) nebo 400 kHz (fast mode).

Pro svoji práci jsem si vybral sběrnici I2C. Výhodou je použití pouze dvou vodičů pro přenos informace. Zároveň lze za sebe takřka neomezeně řetězit zařízení aniž bych zabíral porty na mikroprocesoru. Přenosová rychlost není tak důležitá, jelikož je přenášeno pouze pár desítek bytů za sekundu.

Rozlišení

Dalším parametrem je rozlišení senzoru. Během testování jsem zjistil, že maximální zrychlení během chůze je cca 2 g. V případě, že bych chtěl detekovat i běh je potřeba toto maximum navýšit na cca 4 g pro běžný běh. Rozsahy senzorů se uvádějí v mocninách dvou, nejbližší hodnota je tedy 4 g.

U analogového senzoru však rozlišení nelze nastavit, u něj je nastaveno na pevnou hodnotu. Musím tedy použít senzor, který bude mít nastavené rozlišení kolem 4 g. U digitálního lze toto vyřešit pouhým přenastavením registrů. Běžně používané akcelerometry mají rozlišení od ± 2 g do ± 16 g. Nevýhodou však je, že si tímto zvýšením rozsahu snížím citlivost.

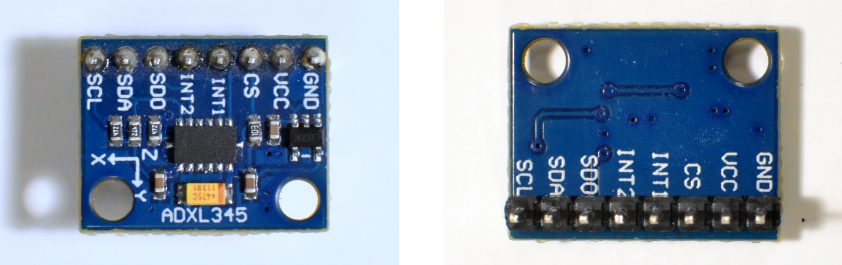
Spotřeba energie

Důležitým parametrem je také spotřeba energie. Pro úspěsnou detekci chůze je potřeba neustále odečítat vzorky zrychlení a ty následně vyhodnocovat.

Proto je potřeba nalézt senzory s co nejnižší spotřebou. Ovšem nesmí to být na úkor schopností samotného senzoru.

U analogového senzoru spotřeba nelze omezit jinak než výběrem správného čipu. U digitálního lze přes sběrnici I2C upravit registry, které snižují spotřebu senzoru. Lze například snížit vzorkovací frekvenci senzoru, případně vypnout určité komponenty jako je gyroskop. Po nastudování několika datasheetů [6], [7] jsem zjistil, že při použití akcelerometru a zároveň gyroskopu několikanásobně stoupne spotřeba energie. Používám tedy vyhodnocovací metody pouze s akcelerometrem.

Digitální senzor – ADXL345



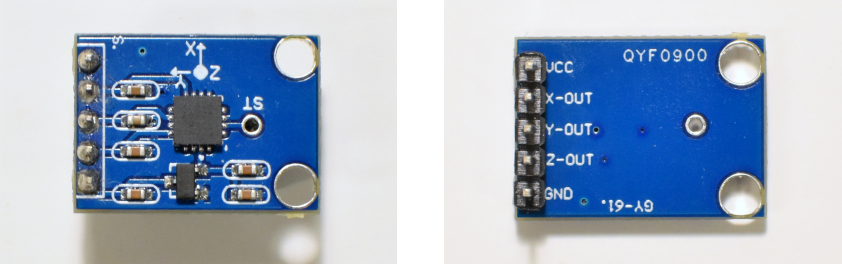
Obrázek 2.1: Senzor ADXL345 na modulu GY-291

Jako digitální senzory jsem si vybral dva – MPU6050 a ADXL345. Jedná se o běžně používané akcelerometry. V následující tabulce porovnám jejich základní vlastnosti podle kterých jsem se rozhodoval.

Parametr	**MPU6050**	**ADXL345**
Typ senzoru	Akcelerometr, gyroskop	Akcelerometr
Rozsah měření	± 2g, ± 4g, ± 8g, ± 16g	± 2g, ± 4g, ± 8g, ± 16
Citlivost	16384, 8192, 4096, 2048 LSB/g	256, 128, 64, 32 LSB/g
Hustota šumu	400 $\mu g/\sqrt{Hz}$	0.75 LSB rms (osy X,Y) 1.1 LSB rms (osa Z)
Napájecí napětí	2,4 V – 3.4 V	2 – 3,6 V
Napájecí proud	G + A = 3,8 mA A = 500 μA	140 μA (ODR $\geq$ 100 Hz) 0.1 μA (Standby)
Rozhraní	I2C, SPI	I2C, SPI

Tabulka 2.1: Porovnání parametrů digitálních senzorů

Všechny parametry byly docela podobné, ale nakonec používám senzor ADXL345. Důvodem jsou bonusové funkce rozpoznávání pohybu, tapu a double tapu. Zároveň je na stránkách výrobce senzoru MPU6050 [8] uvedeno ukončení podpory tohoto senzoru.



Obrázek 2.2: Senzor ADXL335

Analogový senzor – ADXL335

Analogový senzor jsem si vybral stejně jako digitální od firmy Analog Devices – ADXL335. V následující tabulce je ukázka základních parametrů senzoru.

Parametr	**ADXL335**
Typ senzoru	Akcelerometr
Rozsah měření	± 3,6
Citlivost	300 mV/g
Hustota šumu	150 $\mu g/\sqrt{Hz}$ (osy X,Y) 300 $\mu g/\sqrt{Hz}$ (osa Z)
Napájecí napětí	3 V
Napájecí proud	35 μ A
Rozhraní	Analog

Tabulka 2.2: Ukázka parametrů senzoru ADXL335

Senzor vyniká nízkým proudovým odběrem. Rozsah měření ±3.6 g není stejný jako u digitálního, ale je dost podobný.

2.2 Řídicí jednotka

Stejně jako u výběru senzorů i zde si nejdříve shrnu parametry které jsou klíčové při výběru vhodného procesoru. Jedná se o shrnutí možností, které mám k dispozici a snaha o zúžení možností výběru.

Periferie

Mikroprocesor musí obsahovat několik základních periferií potřebných pro zpracování vstupního signálu.

- ADC – převodník analogového signálu, čím vyšší bitové rozlišení bude mít, tím lepší data budu získávat z analogového akcelerometru
- I2C – potřebné pro komunikaci s digitálním senzorem

Jiné další periferie nebudu potřebovat.

Bezdrátová komunikace

Pro komunikaci s okolím je potřeba zajistit nějakou formu přenosu dat. Vzhledem k tomu, že se jedná o zařízení na ruku s následným zobrazení grafů na počítači nechci, aby se uživatelům všude pletly kabely. Využiji tedy několika nejznámějších druhů bezdrátové komunikace:

- Wifi – zařízení se buď připojí na nějakou lokální wifi síť nebo samo vytvoří vlastní access point. Komunikace pak probíhá nějakým vhodně zvoleným typem protokolu.
- Bluetooth LE – propojení na přímo spíše s mobilním zařízením. Výhodou je nízká spotřeba energie jak u odesílatele tak i u příjemce. Lze dosáhnou komunikace i na velké vzdálenosti (desítky metrů).

Zvolil jsem komunikaci přes wifi. Výhodou je její relativně jednoduché na nastavení a udržování.

Nevýhodou je možné zarušení okolních sítí.

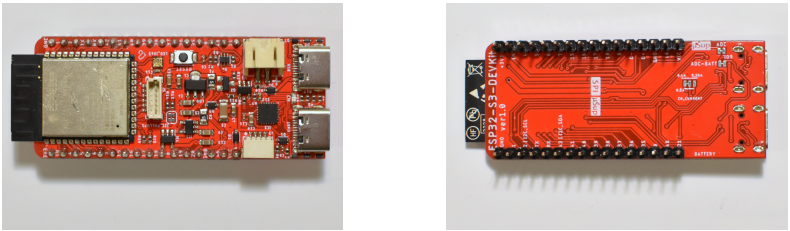
Pracovní napětí

Napájecí napětí obou senzorů je 3,3 V.

Mikroprocesor ESP32 S3

Zvažoval jsem procesory STM32, které vynikají svojí nízkou spotřebou. Jejich nevýhodou však je, že nejsou součástí modulů s wifi/bluetooth ve vhodné velikosti.

Další možností byly procesory od firmy Espressif ESP32. Narozdíl od STM32 jsou součástí modulů spolu s wifi anténou a celkovým příslušenstvím, které velmi zjednodušuje práci s tímto procesorem. Vybral jsem si modul ESP32-S3-DEVKit [9] od společnosti *Laskakit*.



Obrázek 2.3: LaskaKit ESP32-S3-DEVKit

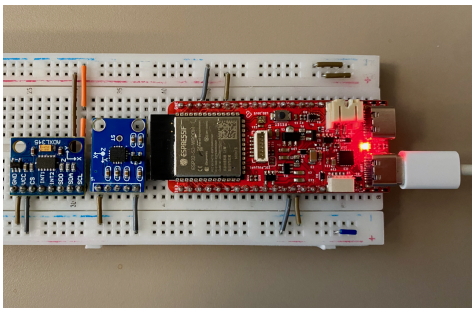
ESP32 disponuje Dual-core 32 bit cpu a dosahuje rychlostí až 240 MHz. Dokáže komunikovat jak přes wifi tak i přes BLE. Anténa je intagravaná na přímo na DPS. Obsahuje řadu periferií, pro mě nejdůležitější jsou IC2 a UART. ESP32 je osazené na DPS, která obsahuje programátor přímo pro ESP a možnost bateriové napájení Li-ion článků. Celá deska má rozměry 66x26 mm. Disponuje také velmi nízkou spotřebou. V deep sleep má spotřebu pouze 12 μA

2.3 Měřicí zařízení

V této kapitole popíšu jak jsem propojil jednotlivé komponenty na nepájivém poli kde jsem si nejprve otestoval funkčnost zapojení. Následně ukážu základní prvky knihoven, které jsem použil pro komunikaci se senzory. Proberu schémata přenosu dat z ESP32 do počítače. Zvážím kompromisy těchto zapojení a vyberu to jež bude pro mě nejvhodnější. Ukážu nastavení Raspberry pi 4 jako mqtt serveru pro přenos dat.

Testovací zapojení

Veškeré vybrané komponenty jsem zapojil do nepájivého pole pro otestování funkčnosti. Už od začátku jsem zaznamenal problém s nefunkčností 3,3 V



Obrázek 2.4: Testovací zapojení v nepájivém poli

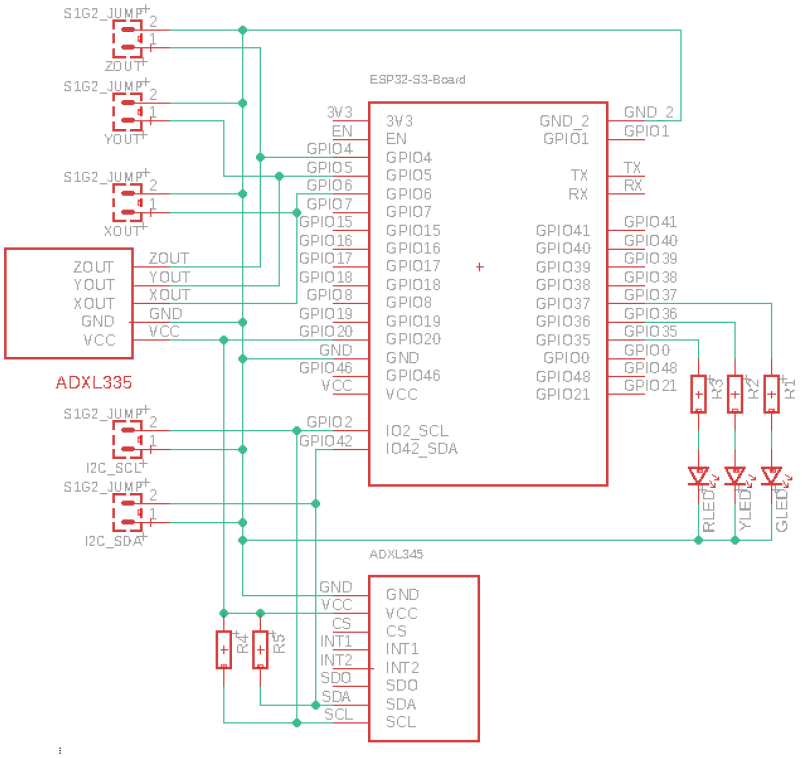
výstupu na ESP32. Výstup nedával dostatečné napětí jak při připojenému napájení přes USB C, tak i při připojené baterii. Zvolil jsem si tedy, že na začátku programu vždy zapnu pin 20 do logické jedničky.

Kontroloval jsem gerber soubory a jak jsou zapojené a zjistil jsem, že není propojený výstup z 3,3 V na mikroprocesoru s 3,3 V na desce. Dál vede cesta k I2C portu tzv. uŠup, ale tam také není nijak dál napojené.

Po otestování funkčnosti na nepájivém poli jsem navrhl desku plošných spojů **??**. Cílem bylo dodržet modulárnost zapojení ale také zároveň rozměr, který je vhodný na ruku.

Původní záměr bylo napájet zařízení z malé powerbanky. Avšak na testovacím zapojení jsem zaznamenal problém s USB C porty na ESP32. Při pohybu se zařízením se stávalo, že se kabel propojující powerbanku a ESP32 také pohnul, což způsobilo výpadek napájení a ESP se restartovalo. Výhodou této desky je, že obsahuje port pro připojení Li-Ion článku a také jeho nabíječku. Využil jsem tedy této možnosti a k ESP jsem připojil 2,5 Ah článek, který udrží zařízení v provozu na cca den. Díky tomu se již ESP nerestartovalo.

Pro uložení komponent jsem navrhl krabičku kterou jsem spojil všechny komponenty 2.7. Při měření lze tuto krabičku buď držet v ruce nebo si ji pomocí suchých zipů připevnit na zápěstí.



Obrázek 2.5: Schéma zapojení modulů

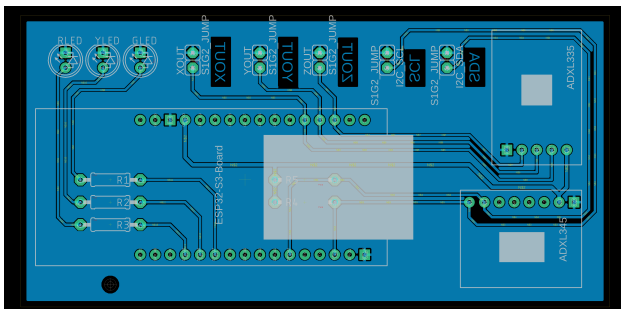
Knihovny pro komunikaci se senzory

Pro komunikaci se senzory jsem si vytvořil knihovny v C++. Chtěl jsem se vyhnout pouhému stažení knihovny z internetu a použití jejich funkcí.

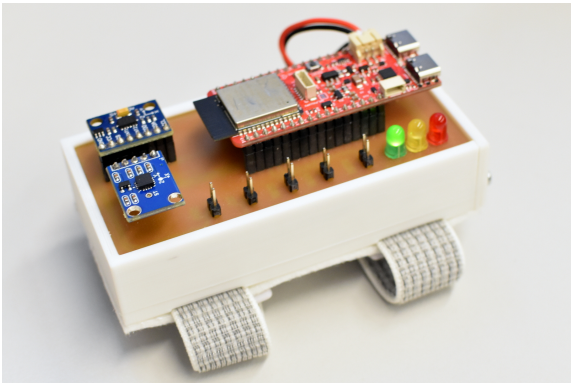
Nejprve jsem vytvořil knihovnu *SensorsLib.hpp* která vytvoří základní struktury společné pro oba senzory. Začal jsem vytvořením struktury ve které bude přenášen vektor zrychlení s časovým údajem.

```
typedef struct acc_t{
unsigned long time;
float X;
float Y;
float Z;

//Sum calculations
acc_t& operator+=(const acc_t& other) {
    this->time = other.time;
    this->X += other.X;
    this->Y += other.Y;
    this->Z += other.Z;
    return *this;
}
}acc_t;
```



Obrázek 2.6: Deska plošných spojů



Obrázek 2.7: Sestavené měřicí zařízení

Jako časový údaj využívám přímo hodnotu, která udává dobu od spuštění programu v milisekundách. Rozlišení to je dostatečné, používám vzorkovací frekvenci v řádu desítek Hz. Hodnoty zrychleni v jednotlivých osách již budou v této proměnné převedené z bitových reprezentací do zrychlení v jednotce g ($g = 10\ m/s$). Přidal jsem také funkci, jež sčítá dvě hodnoty této struktury. Je výhodný například po průměrování hodnot.

Pro komunikaci přes I2C jsem si také napsal vlastní kódy, které čtou z registrů senzoru. Jsou založené na knihovně *Wire.h*.

```
void writeReg(uint8_t addr, uint8_t reg, uint8_t val){
    Wire.beginTransmission(addr);
    Wire.write(reg);
    Wire.write(val);
    Wire.endTransmission(true);
    delay(50);
}

uint8_t readReg(uint8_t addr, uint8_t reg){
    Wire.beginTransmission(addr);
    Wire.write(reg);
    Wire.endTransmission(false);
    Wire.requestFrom(addr, (size_t)1);
    while(Wire.available() < 1);
```

```
15      return Wire.read();
16  }
```

Všechny vstupní hodnoty do této funkce jsou osmibitové. Proměnná *addr* udává adresu zařízení na které se zapisuje. *Reg* je registr do kterého zapisuji. *Val* je hodnota kterou chci do vybraného registru zapsat. Při čtení z registru je funkce téměř identická, jenom má návratovou hodnotu odpovídající požadovanému registru.

Dále knihovna obsahuje funkci pro tisk struktury *acc_t* na sériovou linku. Je vhodná pro debugování kódu, ale zde ji nebudu dále rozebírat.

Knihovna ADXL335

Výstupem tohoto akcelerometru je napětí, které je úměrné zrychlení působící na senzor. Vzorec 2.1 převede hodnotu z ADC na napětí. Pracovní napětí ESP32 je 3,3 V a rozlišení AD převodníku je 12 bitů. Rovnice pro přepočet je

$$U = val_{\text{bin}} * \frac{V_{\text{ref+}} - V_{\text{ref-}}}{2^n} = val_{\text{bin}} * \frac{3,3}{2^{12}}, \tag{2.1}$$

kde

val_{bin} – hodnota získaná z AD převodníku

$V_{\text{ref+}}$ – maximální referenční napětí, v mém případě 3,3 V

$V_{\text{ref-}}$ – minimální referenční napětí, v mém případě 0 V

n – rozlišení AD převodníku, ESP32 využívá 12 bitový převodník

Tímto získáme hodnotu napětí na vstupu ADC. Tu nyní přepočítám na zrychlení v g.

V datasheetu [10] jsou uvedeny hodnoty pro nulové zrychlení $V_s/2$. Velikost napětí pro zrychlení 1 g je proporcionální napájecímu napětí. Pro $V_s = 3,6$ V je citlivost 360 mV/g a pro $V_s = 2$ V je citlivost 195 mV/g. Předpokládám tedy lineární úměru a použiji hodnoty pro nulové zrychlení $k = 1.65$ V a pro citlivost $C = 330$ mV/g.

Pro velikost zrychlení platí

$$|a| = \sqrt{\left(\frac{U_x - k}{C}\right)^2 + \left(\frac{U_y - k}{C}\right)^2 + \left(\frac{U_z - k}{C}\right)^2} \tag{2.2}$$

kde

$U_{\text{x,y,z}}$ ... Výstupní napětí ze senzoru

k ... napětí při nulovém zrychlení (pro jednoduchost jsem dal pro všechny osy stejnou hodnotu, podle datasheetu se tolik neliší)

C ... konstanta pro přepočet napětí na zrychlení (opět uvedená pro všechny osy stejná)

Dále budu pracovat s hodnotami v klidovém stavu. Díky tomu bude velikost zrychlení rovno jedné. Pro jednodušší zpracování upravím tuto rovnici. Nejprve ji celou umocním na druhou abych se zbavil odmocniny.

$$\left(\frac{U_x - k}{C}\right)^2 + \left(\frac{U_y - k}{C}\right)^2 + \left(\frac{U_z - k}{C}\right)^2 = 1 \tag{2.3}$$

Mocniny ještě roznásobím do podoby elipsoidu

$$\frac{(U_x - k)^2}{C^2} + \frac{(U_y - k)^2}{C^2} + \frac{(U_z - k)^2}{C^2} = 1. \tag{2.4}$$

Tuto rovnici použiji pro optimalizaci parametrů k a C tak, aby výsledná hodnota byla co nejblíže jedné. Naměřil jsem si hodnoty napětí pro různé statické polohy (vektor gravitačního zrychlení musí směřovat do různých směrů). V matlabu jsem si napsal skript, který využívá funkce *lsqnonlin*. Ta optimalizuje parametry tak, aby pravá strana rovnice byla co nejblíže jedné. Výsledné hodnoty jsem ještě ověřil a lehce upravil na signálu v reálném čase. Konečné hodnoty parametrů mi vyšly:

$$k = 1.522 \text{ V}$$

$$C = 0.312 \text{ V/g}$$

Upravené hodnoty se velmi málo lišily od předpovězených pouhou aproximací. Používám je tedy pro přepočet napětí na zrychlení.

Knihovna obsahuje třídu ADXL335 v níž popisuji základní funkce a inicializuji hodnoty parametrů přepočtu.

```
class ADXL335
{
private:
    uint8_t Xin;
    uint8_t Yin;
    uint8_t Zin;
    //voltage in zero acceleration
    float ZEROGVOLT = 1.522;
    //convertion constant from V to g (9.81 m/s2)
    float VOLTDIV = 0.312;

    acc_t acceleration; //last measured data
public:
    //Inicialization of sensor
    void init(uint8_t Xpin, uint8_t Ypin,uint8_t Zpin);

    //Measuring acceleration on analog sensor ADXL335
    void measure(void);

    //Get acceleration value
    acc_t getAccel(void);
};
```

Inicializace senzoru probíhá nastavením pinů na kterých budu odečítat napětí. Těm se automaticky přiřadí pull down rezistor, aby v případě špatného zapojení negeneroval náhodné hodnoty ale pouze se vynuloval.

```
void ADXL335::init(uint8_t Xpin, uint8_t Ypin,uint8_t Zpin){
    Xin = Xpin;
    Yin = Ypin;
    Zin = Zpin;
    //default input value is zero
    pinMode(Xin, INPUT_PULLDOWN);
    pinMode(Yin, INPUT_PULLDOWN);
    pinMode(Zin, INPUT_PULLDOWN);
    }
```

Měření přečte hodnoty na výstupu ADC a ty výše uvedenými výpočty 2.1, 2.2 převede na hodnotu zrychlení v dané ose. Výsledná hodnota se uloží do privátní proměnné *acceleration* odkud ji můžu získat s pomocí funkce *getAccel()*.

```
void ADXL335::measure(void){
    acceleration.time = millis();
    //Max voltage 3.3V
    //ADC 12 bits -> 4095 levels
    acceleration.X =
    ((float)analogRead(Xin)*(3.3/4095.0) - ZEROGVOLT)/VOLTDIV;

    acceleration.Y =
    ((float)analogRead(Yin)*(3.3/4095.0) - ZEROGVOLT)/VOLTDIV;

    acceleration.Z =
    ((float)analogRead(Zin)*(3.3/4095.0) - ZEROGVOLT)/VOLTDIV;

}
acc_t ADXL335::getAccel(void){
    return acceleration;
}
```

Knihovna je připravená na odečítání dat z analogového senzoru.

Knihovna ADXL345

Komunikace s tímto senzorem je komplexnější než s analogovým z důvodu většího mnnožství parametrů, jež je potřeba nastavit. Probíhá přes digitální sběrnici I2C. V datasheetu [6] je uvedena adresa zařízení 0x53.

Dále nastavím registry, které na začátku inicializuji a ze kterých následně odečítám hodnoty zrychlení. Nejprve si rozeberu jaké registry jsou pro mě důležité a jaké parametry jimi lze navolit. Zde uvedu pouze výtažek z popisu těchto registrů, více informací lze nalézt v [10]. Všechno pak shrnuji v přehledové tabulce, kde jsou uvedeny adresy registrů a mnou nastavené

hodnoty. Registry procházím od nejnižšího tak, jak jsou uvedeny v oficiálním datasheetu.

BW_RATE – nastavení spotřeby a vzorkovací frekvence

- Upřednostnil jsem nižší šum nad vyšší spotřebou
- Vzorkovací frekvenci jsem nechal na původní hodnotě 100 Hz

POWER_CTRL – nastavení úsporného režimu v závislosti na aktivitě

- Measure bit pokud je nastaven na log. 0, senzor je ve stand by módu, používá se při nastavování registrů
- Sleep bit zastaví přenos dat do FIFO, přepne senzor na nižší vzorkovací frekvenci.

DATA_FORMAT – formát výstupních dat ze senzoru

- Full Res bit v log. 1 měří senzor v plném rozlišení
- Justify bit upravuje zarovnání MSB, pro log. 1 do leva, 0 po pracovat
- Range bit nastaví rozlišení senzoru

DATA – výstupní data ze senzoru

- Z tohoto registru lze pouze odečítat
- Obsahuje poslední naměřené hodnoty

Jednotlivé registry jsem nastavil následovně:

Adresa	Název	Hodnota	Popisek
0x2C	BW_RATE	0b00001010	Vzorkovací frekvence
0x2C	POWER_CTRL	0b00111010	Nastavení Sleep módu
0x31	DATA_FORMAT	0b00101001	Rozlišení senzoru
0x32-0x37	DATA	Read	Hodnoty zrychlení

Tabulka 2.3: Přehled nastavení registrů

Nyní nastavím zvolené registry ve své knihovně. Nejprve nastavuji adresy registrů do kterých zapisuji v hexadecimálním formátu. Poté nastavím hodnoty registrů do mnou zvolených hodnot. Pro lepší přehlednost je zapisuji v binárním formátu.

```
#define ADXL_BW_RATE 0x2C       //R/W
#define ADXL_POWER_CTRL 0x2D    //R/W
#define ADXL_DATA_FORMAT 0x31   //R/W
#define ADXL_DATA 0x32          //R
#define ADXL_FIFO_CTL 0x38      //R/W

//Settings from my bachelor project
#define ADXL_BW_RATE_VAL         0b00001010
#define ADXL_POWER_CTRL_VAL      0b00111010
```

```
#define ADXL_DATA_FORMAT_VAL      0B00101001
#define ADXL_FIFO_CTL_VAL         0b00000000
```

Vytvořím třídu s proměnnými a fukcemi se kterými budu během měření pracovat.

```
class ADXL345
{
private:
    float divider;
    acc_t acceleration; //last measured data
public:
    uint8_t ADXL345_addr; //ADXL345 address (usually 0x53)

    //Setting ADXL345_addr to 0x53
    void init(void);

    //Measuring acceleration on ADXL345
    void measure(void);

    //Get acceleration value
    acc_t getAccel(void);
};
```

Proměnná *divider* ukládá hodnotu kterou vydělím přijaté číslo z I2C sběrnice. Je určená podle nastaveného rozsahu. Pro rozsah ±4 g používám hodnotu 255.

Funkce *init()* nastaví všechny registry do požadovaných stavů. Zároveň inicializuje hodnotu zrychlení do nuly aby nedošlo k nějakým nechtěným chybám.

```
void ADXL345::init(void){
    ADXL345_addr = 0x53;

    //set acceleration to zero
    acceleration.time = 0;
    acceleration.X = 0;
    acceleration.Y = 0;
    acceleration.Z = 0;

    writeReg(ADXL345_addr,ADXL_BW_RATE, ADXL_BW_RATE_VAL);

    writeReg(ADXL345_addr,ADXL_POWER_CTRL,
        ADXL_POWER_CTRL_VAL);

    writeReg(ADXL345_addr,ADXL_DATA_FORMAT,
        ADXL_DATA_FORMAT_VAL);

    writeReg(ADXL345_addr, ADXL_FIFO_CTL, ADXL_FIFO_CTL_VAL);
```

```
19
20      divider = 255;
21  }
```

Měření provádí funkce *measure()*, jež nejprave načte hodnoty ze sběrnice I2C. Zde je potřeba prohodit byty, jelikož senzor nejprve odesílá méně důležitý byte který je nakonci a až poté ten důležitější. Nakonec se přijaté hodnoty vydělí konstantou a uloží se do privátní proměnné třídy. Odtud je mohu získat funkcí *getAccel()*, jejíž návratová hodnota je právě toto zrychlení.

```
1  void ADXL345::measure(void){
2      acceleration.time = millis();
3      Wire.beginTransmission(ADXL345_addr);
4      Wire.write(ADXL_DATA);
5      Wire.endTransmission(false);
6      Wire.requestFrom(ADXL345_addr, (size_t)6);
7      while(Wire.available() < 6);
8      int16_t accX = (Wire.read() | Wire.read()<<8);
9      int16_t accY = (Wire.read() | Wire.read()<<8);
10     int16_t accZ = (Wire.read() | Wire.read()<<8);
11     acceleration.X = (float)accX/divider;
12     acceleration.Y = (float)accY/divider;
13     acceleration.Z = (float)accZ/divider;
14 }
```

V dalším kódu již budu používat jen tyto knihovny.

Schéma propojení ESP32 a PC

Nadefinuji jakým způsobem je možné komunikovat mezi ESP32 a počítačem na němž se zpracovávají naměřená data. Na obrázku 2.8 jsou nakresleny tři možné způsoby. Barvy označují jednotlivé cesty.

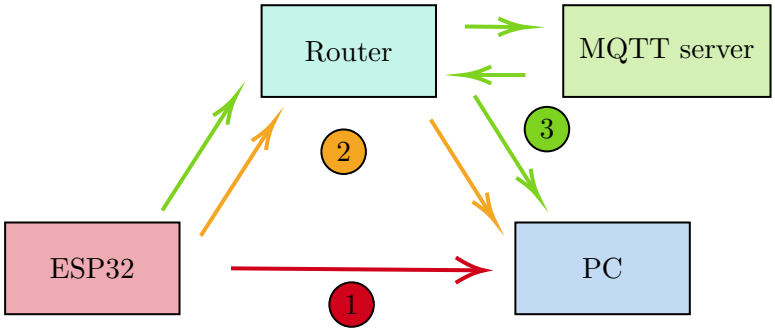
- 1 – Přímé spojení ESP32 - PC
- 2 – Spojeni přes wifi
- 3 – Komunikace přes MQTT server

Přímé propojení ESP – PC

ESP vytvoří wifi síť ke které se připojí počítač (stolní počítač musí mít k dispozici anténu). Přes tuto síť se pak obě dvě zařízení dorozumívají.

PC se pak aktivně ptá IP adresy která náleží ESP (adresa routeru) na nová data. Případně lze aby ESP posílalo broadcast na všechny adresy v síti. Počítač by pak jen tento broadcast zpracovával.

Výhodou tohoto zapojení je jeho jednoduchá realizace. Stačí nastavit ESP a PC se k němu už jen připojí. Nevýhodou je omezený dosah, protože pracuji s malou anténou, která má sloužit jako vysílač, se zařízením u kterého požaduji



Obrázek 2.8: Schéma komunikace

nižší spotřebu. S tím také souvisí otázka, jak by se choval počítač při ztrátě spojení. Musel by být nastaven tak, aby se připojoval výhradně k této síti aby v případě výpadku nedal přednost nějaké silnější síti. Nevýhodou také je, že vytvořením nové wifi sítě omezuji ostatní, které se již na místě nachází.

Připojení ESP na LAN

V tomto případě se snažím eliminovat vytváření nové sítě tím, že se připojím na již existující síť na které je již připojen počítač. Počítač se ptá IP adresy která náleží ESP na nová data. Výhodou je mnohem větší dosah jelikož využívám již navrženou infrastrukturu lokální sítě. Například v případě použití školní sítě bych měl pokrytí po celé škole. Nevýhodou však je, že potřebuji znát alespoň jednu IP adresu aby se mohli vzájemně dorozumívat. To lze vyřešit statickou IP adresou která je přiřazená k jednomu ze zařízení. V případě „veřejných“ sítí je však problém přiřadit statickou adresu, protože správce nechce zabírat jednu adresu pouze pro moje zařízení. Přepisování IP adresy na ESP nepřipadá v úvahu. Dalším řešením je tuto adresu zobrazit na ESP a následně ji do počítače na začátku měření nastavit, ale taková aplikace není moc praktická.

Využití MQTT serveru

V tomto případě využívám služeb MQTT serveru. MQTT (Message Queuing Telemetry Transport) je publish–subscribe komunikační protokol navržený pro zařízení u kterých je vyžadována nízká spotřeba a úzké přenosové pásmo. Metoda publish–subcribe znamená, že zařízení označené jako publisher odesílá data na server do zvoleného *topicu*. *Topic* slouží pro oddělení dat z více senzorů. Druhé zařízení, subscriber, sbírá data která přichází na tento server do *topicu*. ESP je v této konfiguraci publisher který odesílá data na server například do topicu */rawAccel*. Tento topic odebírá počítač, na kterém dále zpracovávám naměřená data.

Výhodou je, že nepotřebuji znát IP adresy ESP ani počítače, ale stačí použít adresu serveru a topic na který odesílám nebo z kterého přijímám data. Volně dostupné servery (například *io.adafruit.com*) umožňují komunikaci mezi publisherem a subscriberem i když se oba nacházejí na různých sítích.

Nevýhodou tohoto řešení je nízká propustnost dat. V neplacené verzi lze poslat 30 datových bodů za minutu, v placené 60. Já však potřebuji přenos v řádu desítek požadavků za sekundu. Musím tedy vytvořit svůj vlastní MQTT server.

Pro realizaci svého projektu jsem zvolil řešení s MQTT serverem.

MQTT server

Jako MQTT server používám Raspberry Pi 4, verze se 4 GB paměti. Jedná se o malý, jednodeskový počítač na kterém běží Raspbian založený na Linuxu. Má ARM procesor díky kterému má velice nízkou spotřebu energie (15 W) aniž by se snižoval výkon samotného zařízení. Součástí jsou také periferie jako USB A, Ethernet, Mini HDMI nebo IO piny.

Pro snadnější správu na Raspberry Pi zapnu wifi access point. Tím se ze samotného Rasberry Pi vytvářím router ke kterému se mohou připojovat další zařízení. Následující kód vytváří takovou síť s názvem *mySSID* a *myPASS*.

```
sudo nmcli device wifi hotspot ssid mySSID password myPASS
```

Abych tento kód nemusel opakovaně psát do terminálu při spuštění Raspberry, vytvořil jsem soubor rc.local s pomocí následujícího příkazu

```
sudo nano /etc/rc.local
```

Do něj zapíšu výše uvedený řádek kódu a zakončím jej příkazem *exit 0*

```
sudo nmcli device wifi hotspot ssid mySSID password myPASS
exit 0
```

Pro tento kód nastavím oprávnění, že se může spustit při startu Raspberry Pi

```
sudo chmod +x /etc/rc.local
```

Při spuštění Raspberry Pi se zapne zároveň wifi síť ke které se mohou připojovat ostatní zařízení.

Následně nastavím MQTT server. Využívám Mosquitto což je open source message broker který implementuje MQTT protokol. Vycházel jsem z návodu [11].

Do Raspberry Pi jsem přes terminnál stáhl Mosquitto server.

```
sudo apt install -y mosquitto-clients
```

Stejně jako u wifi, i zde požaduji aby MQTT server nastartoval při zapnutí Rasberry Pi.

```
sudo systemctl enable mosquitto.service
```

Následně zabezpečím Mosquitto Broker tak, aby se na tento server mohli přihlásit pouze vybraní uživatelé.

```
sudo mosquitto_paswd -c /etc/mosquitto/passwd username
```

Po zadání příkazu do terminálu jsem vybídnut zadat heslo, které potom budu používat pro připojení publisherů a subscriberů k tomuto serveru. Tím se vytvořil soubor */etc/mosquitto/passwd* ve kterém jsou uloženy uživatelské údaje. Tento soubor přiřadím do konfigurace serveru.

Přes terminál si otevřu konfigurační soubor.

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Na začátek tohoto souboru jsem přidal následující příkaz, který zařídí, že nastavení platí pouze pro jednoho posluchače (port).

```
per_listener_settings true
```

Na konec souboru jsem napsal příkazy, které zakáží anonymní přístup k serveru a přiřadí mu soubor s dříve zadaným uživatelským jménem a heslem. Také nastaví přístupový port k serveru na 1883, což je standardní port pro nešifrovanou MQTT komunikaci.

```
allow_anonymous false
listener 1883
password_file /etc/mosquitto/passwd
```

Nakonec restartuji Mosquitto server aby se projevili změny.

```
sudo systemctl restart mosquitto
```

Tímto mám nastavený a připravený MQTT server. Ten je nyní připraven ihned po startu Raspberry Pi a je schopen přenášet zprávy.

ESP32 s MQTT

ESP32 používám jako publishera. Ten odesílá data do zvoleného topicu na MQTT serveru. Aby byla komunikace funkční, používám knihovnu *PubSubClient.h*, která umožňuje jak publikovat tak i odebírat zvolené topicy. Vysvětlím kód který používám pro odesílání dat na server.

Nejprve nastavuji parametry MQTT a wifi sítě. Používám údaje které jsem používal již při nastavování wifi a mqtt serveru. Vytvořil jsem si vlastní topic *topic/rawAccel* do kterého ESP32 publikuje nově naměřené hodnoty.

```
1 const char* ssid = "mySSID";
2 const char* password = "myPASS";
3 const char* mqttServer = "10.42.0.1";
4 const int mqttPort = 1883;
5 const char* mqttUser = "username";
6 const char* mqttPassword = "password";
7 const char* mqttTopic = "topic/rawAccel";
```

IP adresu MQTT serveru jsem zjistil v terminálu Raspberry Pi po zadání příkazu

```
hostname -I
```

Pro správné načasování měření používám metodu, kdy zaznamenám čas při kterém se měřilo naposledy a ten porovnávám s aktuálním časem procesoru. Pokud je rozdíl vyšší než nějaká konstanta, lze měřit znovu. Pokud ne, program chvíli počká aby mohl znovu porovnat jestli neuplynul požadovaný čas. Používám dvě proměnné, kdy do proměnné *lastMsg* je ukládán čas posledního měření a do proměnné *MSG_INTERVAL* interval mezi měřeními. Do komentáře jsem přidal délky intervalu mezi měřeními pro různé vzorkovací frekvence.

```
unsigned long lastMsg = 0;
/**
 * @brief Pro 2 Hz -> t = 499 ms
 * @brief Pro 5 Hz -> t = 199 ms
 * @brief Pro 10 Hz -> t = 99 ms
 * @brief Pro 20 Hz -> t = 49 ms
 * @brief Pro 50 Hz -> t = 19 ms
 */
unsigned long MSG_INTERVAL = 49; //ms
```

Následně inicializuji MQTT klienta, wifi síť a oba senzory.

```
WiFiClient espClient;
PubSubClient client(espClient);
ADXL335 analogADXL;
ADXL345 digiADXL;
```

Nastavím piny pro I2C komunikaci a odečítání hodnot z analogového akcelerometru. Zároveň nastavím výstupní piny pro indikační LEDky. Ty budou sloužit pro určení v jakém stavu se zařízení nachází. Pokud není zařízení připojené k wifi síti bude svítit červená LED. V případě, že wifi vypadne a nelze se dále připojit na MQTT server se rozsvítí žlutá LED. Pokud vše funguje správně tak svítí zelená LED.

```
#define VCCPIN 20
#define I2C_SDA 42
#define I2C_SCL 2

#define RLED 35
#define YLED 36
#define GLED 37

#define Xin 6
#define Yin 5
#define Zin 4
```

Zavedu funkci pro připojenní k wifi síti. Nejprve je zapnutá červená LED, která indikuje, že zařízení není k wifi připojeno. Následuje připojování se k vybrané síti. Funkce zkouší dvakrát za sekundu připojení k wifi dokud se zařízení nepřipojí. Po připojení se rozsvítí zelená indikační LED. Na sériový výstup se výpíší parametry wifi sítě, ke které se zařízení připojilo.

```
void wifi_connect() {
    digitalWrite(RLED, HIGH);
    digitalWrite(GLED, LOW);
    digitalWrite(YLED, LOW);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    delay(10);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    digitalWrite(RLED, LOW);
    digitalWrite(GLED, HIGH);
    digitalWrite(YLED, LOW);
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
```

Funkce *reconnect()* zajišťuje stálé připojení k MQTT serveru. V případě, že dojde k odpojení ESP od serveru se jej tato funkce připojí zpět. Proces připojování se opakuje každých 5 sekund, kdy se vyzkouší jestli již lze navázat spojení. Pokud nelze, rozsvítí se žlutá LED a na sériový výstup se vypíše chybová hláška.

```
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if(client.connect("ESP32Client",mqttUser,mqttPassword
    )){
            Serial.println("connected");
        } else {
            digitalWrite(RLED, LOW);
            digitalWrite(GLED, LOW);
            digitalWrite(YLED, HIGH);
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
```

```
}
```

Funkce *setup()* provede inicializaci ESP. Nastaví módy jednotlivých pinů, které v kódu používám. *VCCPIN* nastavuji kvůli chybě na desce, která způsobuje, že pin 3.3V určený pro napájení nenapájí. Vytvořil jsem tuto alternativu napájení veškerých senzorů.

Proběhne inicializace všech sběrnic a senzorů, které používám. Nejprve je nastavena sběrnice I2C. Jsou jí přiřazeny piny 2 a 42. Rychlost přenosu je nastavena na 100 kHz. Dále probíhá inicializace sériové linky používané pro debug kódu. Ta je nastavena na rychlost 115200 baudů. Následně proběhne nastavení senzorů. Analogovému jsou přiřazeny piny, ze kterých lze odečítat analogový signál z jednotlivých os. Nakonec je ESP připojeno k wifi síti a k MQTT serveru.

```
void setup(){
    pinMode(VCCPIN, OUTPUT);  // power
    digitalWrite(VCCPIN, HIGH);
    pinMode(RLED, OUTPUT);
    pinMode(YLED, OUTPUT);
    pinMode(GLED, OUTPUT);
    pinMode(Xin, INPUT);
    pinMode(Yin, INPUT);
    pinMode(Zin, INPUT);

    Wire.begin(I2C_SDA, I2C_SCL, 100000);

    Serial.begin(115200);
    analogADXL.init(6, 5, 4);
    digiADXL.init();

    wifi_connect();
    client.setServer(mqttServer, mqttPort);
}
```

Samotné měření a odesílání zpráv probíhá ve funkci *loop()*. Ta probíhá neustále od spuštění ESP do jeho ukončení. Funkce *client.loop()* se snaží udržet aktivní připojení k MQTT serveru. Periodicky odesílá ping na server aby server nezrušil nečinné spojení. Probíhá odečtení aktuálního procesorového času, který se následně porovnává s hodnotou, kdy naposledy proběhlo měření. Pokud je rozdíl vyšší než zvolený interval může měřeení proběhnout. Testuje se, zda-li je ESP připojeno k MQTT serveru. Pokud ne, snaží se o opětovné připojení. Při měření je aktivní zelená indikační LED. Značí, že měření probíhá v pořádku. Jsou naměřeny hodnoty na senzorech, které se následně převedou do formátu JSON. Ten se serializuje a odešle se do zvoleného topicu.

```
void loop(){
    client.loop();

    unsigned long now = millis();
```

```
    if (now - lastMsg > MSG_INTERVAL) {
        if (!client.connected()) {
            reconnect();
        }
        digitalWrite(RLED, LOW);
        digitalWrite(GLED, HIGH);
        digitalWrite(YLED, LOW);
        lastMsg = now;
        analogADXL.measure();
        digiADXL.measure();
        acc_t analogValue = analogADXL.getAccel();
        acc_t digitalValue = digiADXL.getAccel();

        JsonDocument doc;
        doc["time"] = digitalValue.time;
        doc["ad_x"] = digitalValue.X;
        doc["ad_y"] = digitalValue.Y;
        doc["ad_z"] = digitalValue.Z;
        doc["aa_x"] = analogValue.X;
        doc["aa_y"] = analogValue.Y;
        doc["aa_z"] = analogValue.Z;

        // JSON to string
        char msg[150];
        serializeJson(doc, msg);

        client.publish(mqttTopic, msg);
    }
}
```

Příjem MQTT zpráv

Příjem MQTT zpráv na počítači je založeno na stejných principech jako odesílání zpráv z ESP. Kód je napsaný v pythonu, jelikož je snadno spustitelný na všech počítačích. Využívá knihovnu paho.mqtt která se běžně používá pro komunikaci s MQTT serverem. Používám verzi 1.6.1 protože na nejnovější verze knihovny s tímto kódem nefunguje.

Přijatá data ukládám do csv souboru a využívám struktury kterou naměřeným hodnotám dává JSON. Na tento soubor je napojená aplikace v matlabu. Před spuštěním tohoto kódu je potřeba se připojit na wifi síť z Raspberry Pi.

Na začátku souboru importuji knihovny které potřebuji pro práci s csv souborem a JSONem. Z knihovny *paho.mqtt* vkládám část pro práci s MQTT klientem. Dále nastavím parametry klíčové pro připojení k MQTT serveru. A nastavím název csv souboru do kterého se ukládají naměřená data.

```
import csv
import json
```

```
from paho.mqtt import client as mqtt_client
import os

broker = '10.42.0.1'
port = 1883
topic = "topic/rawAccel"
client_id = 'my_id'
username = 'username'
password = 'password'
csv_file = 'mqtt_data.csv'
```

Funkce *connect_mqtt()* zařizuje připojení k MQTT serveru. O úspěchu této akce informuje uživatele do terminálu. Nastavuje uživatelské jméno a heslo, kterým se program připojuje na daný server.

```
def connect_mqtt():
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code \%d\n", rc)

    client = mqtt_client.Client(client_id)
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

Funkce *subcribe()* odebírá zprávy z topicu a dekóduje je. Dekódovaná data pak pošle funkci *write_to_csv()*, která data uloží do csv souboru.

```
def subscribe(client: mqtt\textunderscore client):
    def on_message(client, userdata, msg):
        data = msg.payload.decode()
        write_to_csv(data)

    client.subscribe(topic)
    client.on_message = on_message
```

Po spuštění programu funkce *write_to_csv()* vytvoří csv soubor s hlavičku vhodnou pro uložení načítaných dat. Pokud takový soubor již existuje, funkce ho přepíše. Funkce přeloží data z JSONu a hodnoty přiřadí do správných sloupců.

```
def write_to_csv(data):
    with open(csv_file, mode='a', newline='') as file:
        writer = csv.writer(file)
        if os.stat(csv_file).st_size == 0:
            writer.writerow(['time', 'ad_x', 'ad_y', 'ad_z',
```

```
                'aa_x', 'aa_y','aa_z'])
        data_dict = json.loads(data)
        writer.writerow([data_dict['time'], data_dict['ad_x'],

            data_dict['ad_y'], data_dict['ad_z'], data_dict['
    aa_x'],
            data_dict['aa_z'], data_dict['aa_z']])
```

Funkce *run()* zařizuje, že program běží v neustálém cyklu dokud jej uživatel neukončí. Při startu programu funkce vymaže stávající csv soubor, aby nedocházelo k zbytečné kumulaci dat.

```
def run():
    if os.path.isfile(csv_file) and os.stat(csv_file).st_size
    !=0:
        os.remove(csv_file)
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()

if __name__ == '__main__':
    run()
```

2.4 Matlab aplikace

V této části ukážu jakým způsobem proběhlo zpracování dat. Zařízení bude používáno ve výuce, rozhraní musí být uživatelsky přívětivé. Využil jsem možnosti Matlabu pro navržení aplikace. Výhodou tohoto řešení je, že se nemusím zabývat návrhem designových prvků, Matlab mi je dává k dispozici, a mohu se plně soustředit na samotnou funkčnost aplikace.

Úvod do aplikace

První panel aplikace slouží jako shrnutí celého měření. V textu je popsáno jak bude měření probíhat a jaké jsou jeho cíle aby se na to studenti mohli připravit.

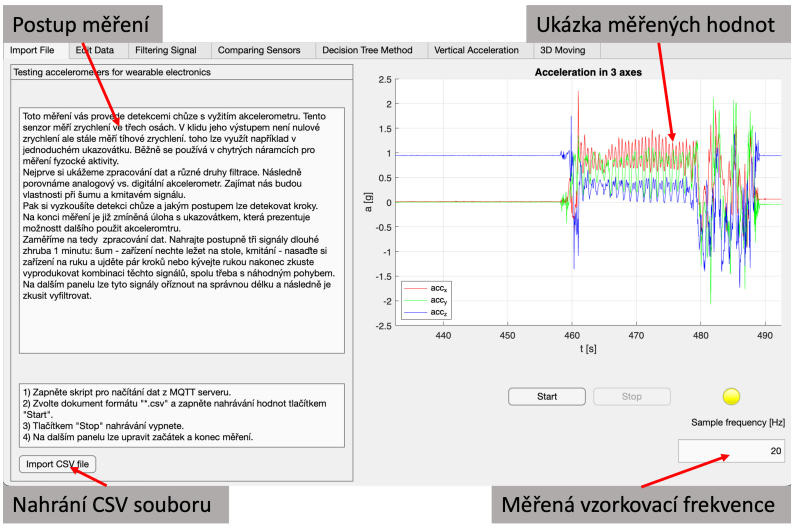
Pod úvodním textem je návod na vložení csv souboru s naměřenými daty. Nejprve je potřeba počítač k wifi síti vytvořené Raspberry Pi. Následně se spustí skript pro odečítání dat z MQTT serveru. Tlačítkem *Import CSV File* se do aplikace nahraje soubor s naměřenými daty. Na pravé straně se grafu zobrazí průběh záznamu posledních 60 sekund uložených dat. Zároveň se tlačítko start stane interaktivním. Jeho stisknutím se spustí nahrávání dat do paměti aplikace. Graf se jednou za desetinu sekundy aktualizuje a zobrazí aktuální průběh dat. Díky tomu je vidět průběh zrychlení v reálném čase. Data lze upravovat na následujicích panelech aplikace. Pokud je měření aktivní je rozsvícena zelená indikační kontrolka, pokud je pozastaveno, svítí

oranžově. V pravém dolním rohu je textové okno které zobrazuje vzorkovací frekvenci dat, tak je vypočtena

$$f_s = \frac{N}{\sum_{n=1}^{N-1} s\,[n] - s\,[n-1]}. \tag{2.5}$$

Jedná se o převrácenou hodnotu z rozdílu časů u nejbližších vzorků.

Na obrázku 2.9 je zobrazena úvodní stránka aplikace s popisky.



Obrázek 2.9: Úvodní panel aplikace

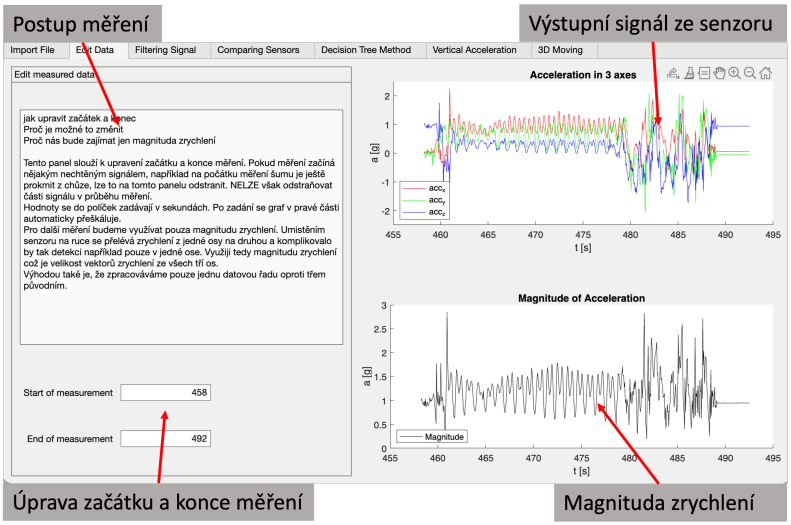
Úvod do filtrace

Cílem této části je seznámit studenty se základy filtrace. Vyzkouší si použití různých filtrů na různých signálech.

Na panelu 2.10je možnost upravit začátek a konec dat. Na pravé straně jsou grafy které zobrazují tato vybraná data aby uživatel názorně viděl na jaké hodnoty tato data má upravovat.

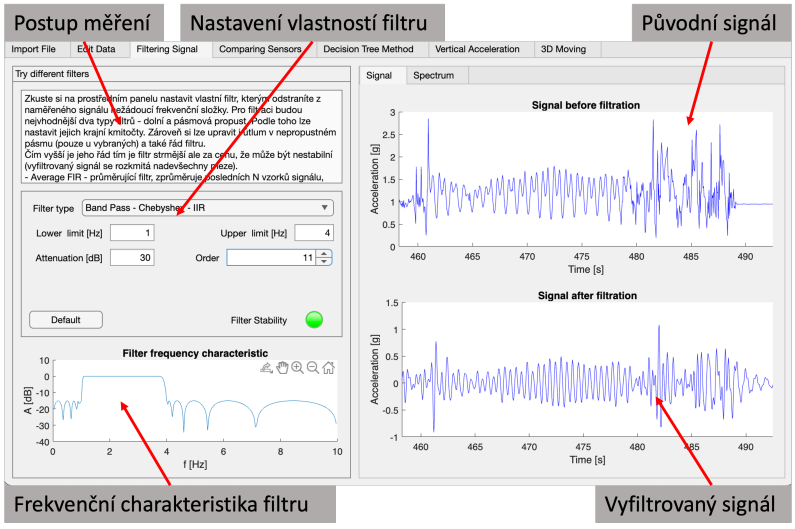
Panel *Filtering Data* 2.11 slouží k filtraci signálu a jeho porovnání k signálu původnímu. Na levé straně se nachází základní popis použitých filtrů, jejich vlastnosti a výhody nebo nevýhody. Na výběr je z pěti různých filtrů rozdělených do dvou hlavních skupin. FIR – s konečnou impulzovou odezvou a IIR – s nekonečnou impulzovou odezvou. U filtrů lze upravovat parametry jako útlum v nepropustném pásmu, řád filtru a horní a dolní hraniční frekvence. Ty parametry, které lze upravovat pro daný filtr se stanou interaktivními. V levém dolním rohu je zobrazena frekvenční charakteristika nastaveného filtru. U diskrétních signálů je tato charakteristika symetrická a proto ji zobrazuji pouze do poloviny vzorkovací frekvence.

Na pravé straně lze porovnat podobu vstupního a výstupního signálu. Uživatel si může zobrazit jak samotný signál tak i jeho spektrum. Délka okna výpočtu spektrogramu je nastavena na 64 vzorků. Toto je optimální nastavení



Obrázek 2.10: Panel pro editování měření

pro zobrazení spektrogramu s vhodným rozlišením jak ve frekvenční tak i časové oblasti.

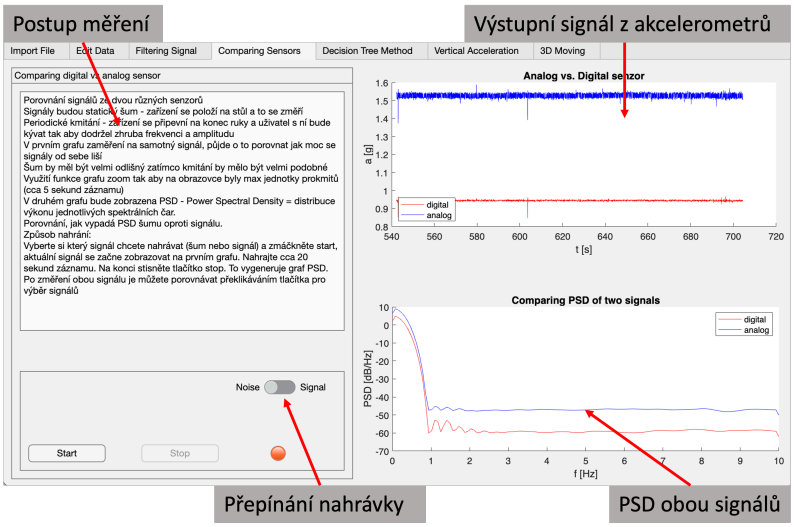


Obrázek 2.11: Filtrování signálu

Porovnání senzorů

Tento panel 2.12 slouží pro porovnání výstupních signálů z analogového a digitálního senzoru. Jsou zde dostupné dva grafy. První zobrazuje porovnání samotných signálů. Ukazuje, jak se liší surová data ze senzorů. Druhý zobrazuje PSD (Power Spectral Density = Spektrální výkonová hustota) neboli rozmístění výkonu napříč spektrem signálu.

Na levé straně lze spustit záznam signálu, který se zpracuje. Přepínacím



Obrázek 2.12: Porovnávání signálů z analogového a digitálního senzoru

tlačítkem lze nastavit jaký typ signálu nahraji. Nahrávání šumu znamená, měření pouze velikost tíhového zrychlení. V signálu se nesmí vyskytnout žádná periodická složka. Na grafu PSD se zobrazí síla šumu obou senzorů. Čím nižší tento šum bude, tím lépe senzor měří. Pro nahrávání signálu zvolím jakýkoliv periodický signál s frekvencí nižší než je polovina vzorkovací frekvence (jinak nesplňuji Shannon-Nyquistův vzorkovací teorém). Při měření je používána vzorkovací frekvence 20 Hz, maximální měřitelná frekvence je 10 Hz.

Po naměření obou signálů lze přepínat mezi těmito signály a porovnávat jejich charakteristiky.

1. Metoda detekce chůze

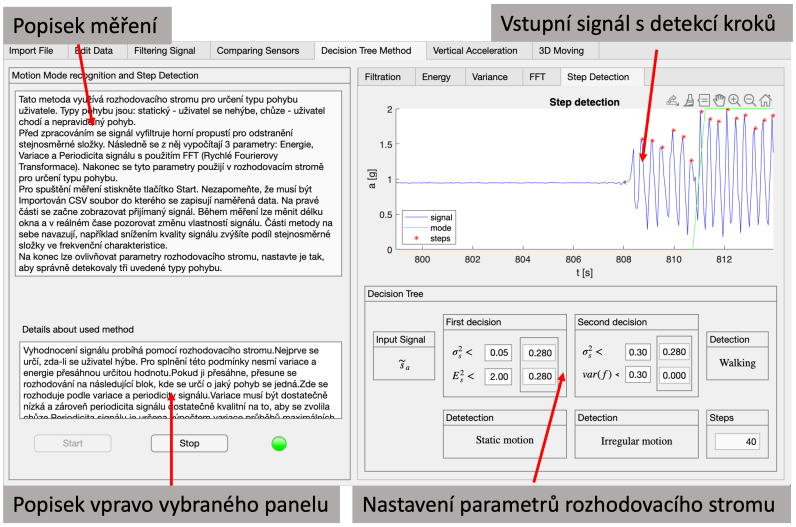
Detekce chůze je založena na rozhodovacím stromu, který určí o jaký typ signálu se jedná podle parametrů vypočítaných z naměřeného signálu. Detekce chůze probíhá v reálném čase veškeré parametry se tedy počítají okamžitě.

Na levé straně je text popisující jakým způsobem funguje tato matoda. Pod ním se nachází text, který se vyjadřuje přímo ke zvolenému panelu v pravé části. V něm je konkrétní popis výpočtu daného parametru. V levém spodním rohu se nachází tlačítka pro spuštení a ukončení měření. Při startu měření se resetuje proměnná počítající naměřené kroky. Všechny grafy jsou automaticky aktualizovány.

Na pravé straně se na chází panely s grafy. Mezi panely lze během měření přepínat a měnit délku okna. V grafech je porovnáván původní signál se signálem právě zvoleného parametru. Výjimkou je panel frekvenční analýzy kde nahoře je graf frekvenční analýzy signálu z okna 128 vzorků a dole je průběh tří maximálních frekvencí.

Výsledná detekce chůze je vidět na posledním panelu 2.13 kde je zobrazen vstupní signál a zároveň signál určující typ pohybu. Tento signál se přepíná mezi třemi úrovněmi:

- 0 – uživatel se nepohybuje
- 1 – chaotický pohyb
- 2 – periodický pohyb (chůze)



Obrázek 2.13: Ukázka matody detekce chůze

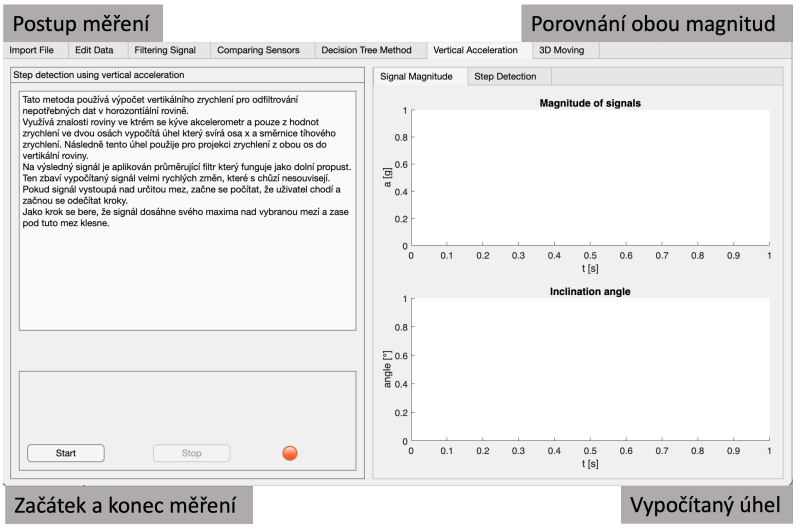
2. Metoda detekce chůze

Tato metoda je založena na vypočítání vertikálního zrychlení. Na hlavním panelu je porovnání dvou signálů. Magnitudy celého zrychlení a vypočítaného vertikálního zrychlení. V pravém spodním rohu je zobrazení vypočítaného úhlu, kde lze pozorovat natočení v ploše XY. Na druhém panelu je samotná detekce chůze. Je zde vykreslen vypočítaný signál a body kde byl detekován krok. Ve spodní části lze upravit hodnotu hranice nad kterou se počítá chůze. Zároveň je zde vidět počítadlo kroků které se restartuje po dalším spuštení měření.

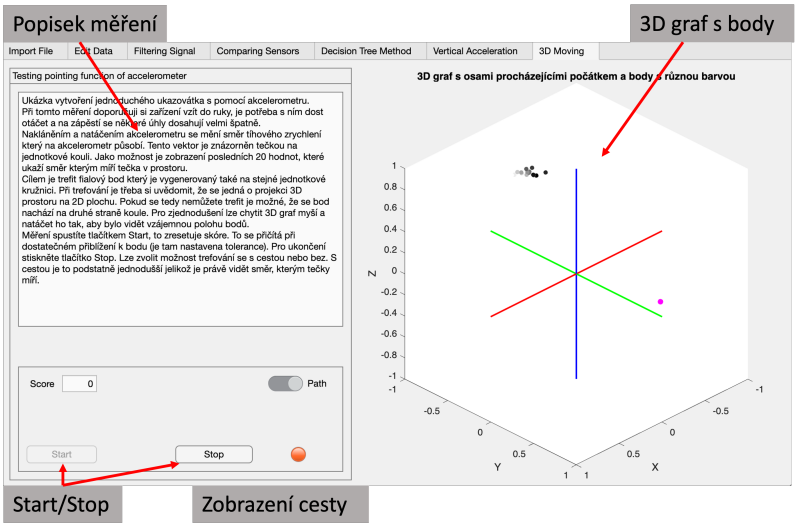
3D ukazovátko

Tato aplikace 2.15 demonstruje použití akcelerometru jako jednoduchého ukazovátka. Na pravé straně je 3D graf do kterého se zobrazuje tečka/tečky které představují vektor tíhového zrychlení které na akcelerometr působí. Tento vektor je normován tak, aby se tyto tečky pohybovaly po jednotkové sféře. Na sféře je vygenerován náhodný bod do kterého se uživatel nakláněním senzoru trefuje. Trefování zjednodušuje zobrazení záznamu 20 posledních hodnot, které pomáhá určovat směr jakým se tečky po kouli pohybují.

Tato úloha také prezentuje problém s projekcí 3D prostoru na plochu.



Obrázek 2.14: Ukázka matody detekce chůze



Obrázek 2.15: Jednoduché ukazovátko

Kapitola 3
Měření

- *Naměření vhodné vzorkovací frekvence*
- *Vyzkoušení metod na nějakém definovaném úseku*
- *Grafy s porovnáním jednotlivých metod*

Pro měření jsem využíval svoji aplikaci napsanou v Matlabu.

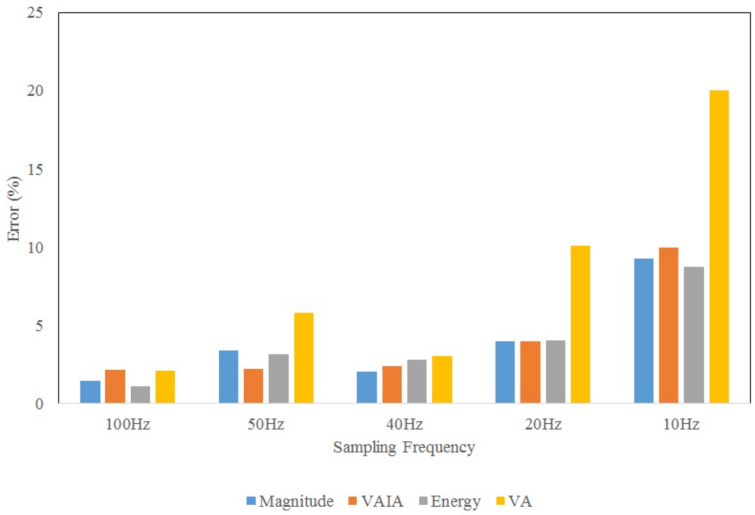
3.1 Parametry měření

Vzorkovací frekvence

Volba vhodné vzorkovací frekvence je velmi důležitou součástí jakéhokoli měření diskrétního signálu. Pro správné zobrazení signálu je vžy třeba volit vzorkovací frekvenci vyšší něž je dvojnásobek maximální frekvence. Pokud to nesplním, dojde k aliasingu. Ten vede ke ztrátě informace v a zkreslení vzorkovaného signálu. Chůze má hlavní frekvenci okolo 2 Hz, ta se mění v závislosti na rychlosti (čím rychleji jdeme, tím rychleji kmitá ruka). Zároveň výstupní signál obsahuje další frekvence typické pro chůzi, které jsou sice slabší než hlavní frekvence, ale také nám mohou poskytnnoutt důležité informace o pohybu.

U výběru vhodné vzorkovací frekvence jsem vycházel z práce [5]. V ní je porovnávána chybovost algoritmu detekce kroků při různých vzorkovacích frekvencích. Pracovali s frekvencemi 10 Hz, 20 Hz, 40 Hz, 50 Hz a 100 Hz. Na každé této vzorkovací frekvenci signálu aplikovali čtyři různé metody detekce kroků a porovnávali chybovost těchto detekcí. Výsledek je na grafu 3.1. Chybovost zvolených metod je na frekvencích od 20 do 100 Hz pod hodnotou 5 %. Výjimkou je metoda v níž jsou počítány kroky z vertikálního zrychlení, která má na 20 Hz chybovost 10 %. Nejhorší jsou výsledky pro vzorkovací frekvenci 10 Hz, kde nejlepší chybovost je 10 %

Zvolil jsem vzorkovací frekvenci 20 Hz, jelikož se jedná o nejlepší kompromis mezi co nejnižší frekvencí (nižší spotřebou měřicího zařízení) a chybovostí. S vyšší vzorkovací frekvencí již chybovost detekce tolik neklesá.



Obrázek 3.1: Porovnání chybovosti použitých metod v závoslisti na vzorkovací frekvenci

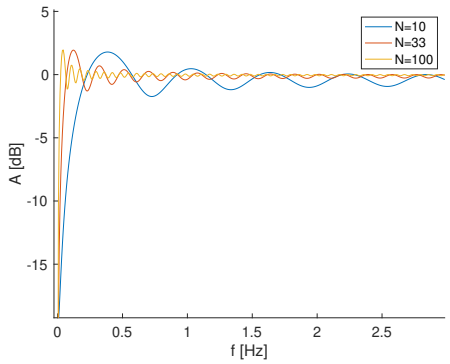
Filtr stejnosměrné složky

FIR filtr horní propusti uvedený vztahem

$$s_{\mathrm{rms0}}[n] = s_{\mathrm{rms}}[n] - \frac{1}{N}\sum_{l=0}^{N-1} s_{\mathrm{rms}}[n-l]\,. \tag{3.1}$$

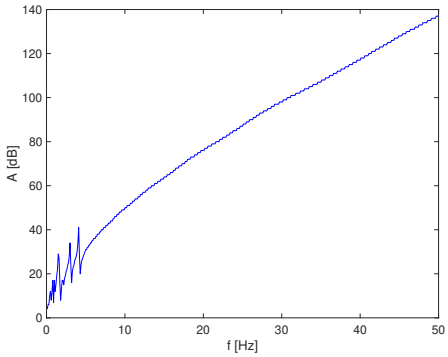
Tento filtr odstraňuje výhradně stejnosměrnou složku signálu.

Filtrace spočívá v tom, že filtr vypočítává průměr z posledních N vzorků, který odečítá od posledního načteného vzorku. Délka okna určuje jak vysoký je útlum stejnosměrné složky a zároveň, protože se jedná o FIR filtr, jak moc bude signál zpožděný. Na grafu frekvenční charakteristiku a impulzovou odezvu pro tři různě dlouhá okna. Použiji délky N = 10, 33 a 100; Z grafů lze



Obrázek 3.2: Frekvenčí charakteristika pro okna různé délky

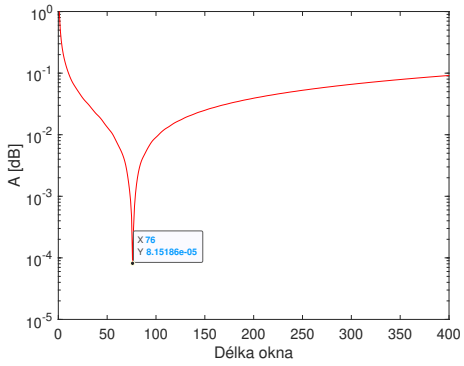
pozorovat, že se zvyšující délkou okna se zvyšuje útlum stejnosměrné složky. Zároveň se snižuje rozkmit signálu u ostatních frekvenšních složek. Pokud



Obrázek 3.3: Frekvenční charakteristika pro okna různé délky

si zobrazím závislost útlumu na délce okna, vyjde mi, že má logaritmický průběh $10 \log_{10}(N)$. Čím více prodlužuji délku okna tím menší bude úbytek útlumu. Tuto vlastnost jsem si ještě ověřil na simulovaném reálné signálu. Vytvořil jsem signál se stejnosměrnou složkou a třemi sinusovými průběhy na frekvencích 1,7 Hz, 3,1 Hz a 4,2 Hz. To simuluje chůzi a její harmonické složky. Následně jsem signál navzorkoval vzorkovacími frekvenci od 2 do 50 Hz.

Z takto navzorkovaných signálů jsem spočítal FFT a zanesl si poměr vyfiltrované stejnosměrné složky oproti nevyfiltrované. To jsem provedl pro různě dlouhá okna a snažil jsem se nalézt okno s nejnižším útlumem. Výsledný vztah vzorkovací frekvence a minimálního útlumu jsem zobrazil do grafu**??**. Pro vzorkovací frekvenci 20 Hz má toto okno délku 76 vzorků. V grafu 3.5 je uveden průběh útlumu signálu na délce okna. Charakteristika má tvar

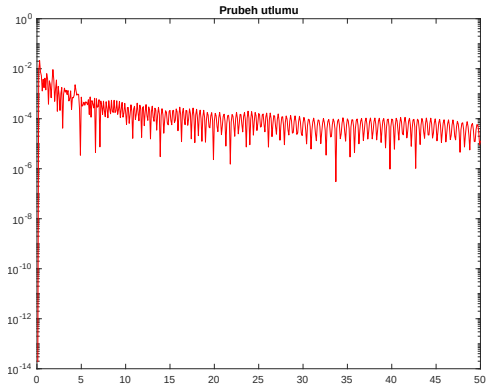


Obrázek 3.4: Průběh útlumu na 20 Hz pro různé délky okna

hokejky, kdy v nule a v nekonečnu jsou maxima a minimum je právě v hodnotě 76.

Frekvenční analýza signálu

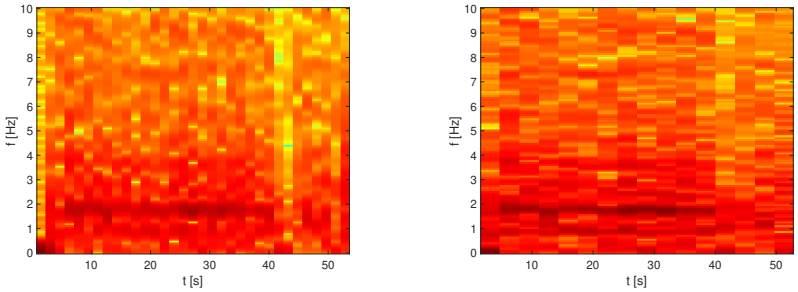
Pro frekvenční analýzu zvolím vhodné okno délky ze kterého se bude počítat. Délku okna signálu musím volit jako mocninu 2 pro využití rychlého výpočtu



Obrázek 3.5: Průběh útlumu stejnosměrné složky v závislosti na vzorkovací frekvenci

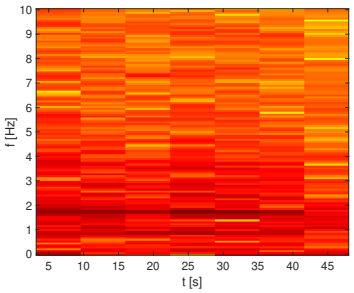
FFT. Zde opět nastává problém se zpožděním signálu. V případě signálu navzorkovaného frekvencí 20 Hz je zbytečné uvažovat o délce okna delší než 256 vzorků. Výstupní signál by tak byl zpožděn o cca 13 s. Zároveň velikost tohoto okna udává jak moc bude reagovat na změny. Kratší okno bude zaznamenávat krátké změny signálu a výstupní signál tak bude častěji rozkmitaný.

Délka okna také ovlivňuje kvalitu spektrogramů, které v aplikaci generuji. Pro dlouhá okna si mohu dovolit vyšší segmentaci signálu ze které se spektrogram počítá. Tím získám mnohem přesnější informaci o proměnlivosti dat v časové oblasti. Avšak přicházím o přesnost v časové oblasti. Na obrázku 3.7 jsou zobrazeny spektrogramy pro stejný signál chůze. Lze si povšimnout, že rozlišení je u každého spektrogramu různé.



(a) : N = 64 **(b) :** N = 128

Pro nejlepší rozlišení spektrogramu jsem zvolil délku segmentu 128 vzorků.



(a) : N = 256

Obrázek 3.7: Spektrogramy pro různé délky segmentů

Závěr

V této bakalářské práci jsem se zabýval detekcí chůze s pomocí dat z běžně dostupných akcelerometrů. Cílem bylo vytvořit aplikaci v Matlabu která bude pomáhat uživatelům pochopit, jakým způsobem jsou data z akcelerometru zpracovávána.

V první kapitole jsem probral co to vlastně je zrychlení a jakým způsobem se měří. Zaměřil jsem se také na metody výroby těchto akcelerometrů.

Vytvořil jsem zařízení schopné zpracovávat naměřené zrychlení a odesílat je na MQTT server. Odtud si je bere počítač, který data vkládá do aplikace v Matlabu kde jsou také zobrazována. V aplikaci si je možné osvojit a pochopit zpracování těchto dat. Ať už se jedná o filtraci signálu, kde uvádím různé přístupy. Nebo se jedná o statistické výpočty ze signálu jako je energie či variace. Nakonec jsem aplikoval metody detekce chůze sloužící pro určení počtu kroků zvýše uvedených parametrů.

Literatura

[1] Jacob Fraden. *Handbook of modern sensors. physics, designs, and applications*. Fifth edition. Cham: Springer, [2016]. ISBN: ISBN978-3-319-19303-8.

[2] Pavel Ripka a Alois Tipek. *Modern Sensors Handbook*. 1. vyd. Great Britain a the United States: ISTE Ltd, 2007. ISBN: 978-1-905209-66-8.

[3] *Invensense MPU6050 6-axis MEMS IMU*. 2013. URL: https://zeptobars.com/en/read/Invensense-MPU6050-6d-MEMS-IMU-gyroscope-accelerometer.

[4] Melania Susi, Valérie Renaudin a Gérard Lachapelle. „Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users“. In: *Sensors* 13.2 (2013), s. 1539–1562. ISSN: 1424-8220. DOI: 10.3390/s130201539. URL: http://www.mdpi.com/1424-8220/13/2/1539 (cit. 29. 11. 2023).

[5] Edith Pulido Herrera et al. „Sampling Frequency for Step Detection Based on Smartphone Accelerometry“. In: (2017), s. 409–413. DOI: 10.1109/ISPAN-FCST-ISCC.2017.89. URL: http://ieeexplore.ieee.org/document/8121805/ (cit. 20. 12. 2023).

[6] ANALOG DEVICES. *ADXL345 datasheet*. [Revize 5/2022]. [Online]. Dostupné také z: URL: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf.

[7] InvenSense Inc. *MPU6050 datasheet*. [Revize 8/2013]. [Online]. Dostupné také z: URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[8] InvenSense Inc. *MPU6050*. URL: https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/ (cit. 21. 05. 2024).

[9] LaskaKit. *LaskaKit ESP32-S3-DEVKit*. URL: https://www.laskakit.cz/laskakit-esp32-s3-devkit/ (cit. 21. 05. 2024).

[10] ANALOG DEVICES. *ADXL335 datasheet*. [Revize 1/2010]. [Online]. Dostupné také z: URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf.

[11] *How to install mosquitto broker on Raspberry Pi.* URL: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/ (cit. 21. 05. 2024).