

# Príloha A: Plán práce

## A.1 Zimný semester

Obdobie	Náplň práce
1. týždeň	Základný prehľad relevantnej literatúry.
2. týždeň	Štúdium literatúry ohľadom monitorovania vibrácií. Pokusný zber dát akcelerácie z MHD pomocou akcelerometra na smartfóne a ich prieskumná analýza.
3. týždeň	Štúdium článkov o frekvenčnej analýze a rešerš algoritmov na hľadanie špičiek. Implementácia objavených prístupov hľadania špičiek a aplikovanie na merania vibrácií z električiek a autobusov.
4. týždeň	Flashovanie firmvéru na vývojový kit iCOMOX od Shiratech.
5. týždeň	Osnova práce s referenciami na nájdenú literatúru.
6. týždeň	Firmvér pre dosku na platforme ESP32 pre záznam akceleračných dát na SD kartu cez OpenLog
7. týždeň	Merania vibrácií v MHD a analýza získaných záznamov v Jupyter notebooku. Doplnenie zdrojov pre časti osnovy s málo referenciami.
8. týždeň	Sekcia 2.1. práce o monitorovaní vibrácií a šoku.
9. týždeň	Doplnenie typov akcelerometrov a časti o numerickej kvadratúre k sekcií 2.1. Úvod do sekcie 2.2. o analýze v časovej doméne.
10. týždeň	Deskriptívne štatistiky a algoritmy na identifikáciu špičiek.
11. týždeň	Sekcia 2.2. o frekvenčnej a časovo-frekvenčnej analýze signálu.
12. týždeň	Sekcia 2.3 o architektúre senzorových sietí a ich obmedzeniach. Návrh riešenia a úvod k priebežnej správe BP1.
13. týždeň	Zapracované pripomienky k prezentovanému návrhu.

Rozvrhnutie pred začiatkom zimného semestra sa držalo dvoch oporných termínov a sice 6. týždňa a 12. týždňa. V 6. týždni sme chceli zavŕšiť rešerš podstatných zdrojov literatúry podľa predstavy o charaktere vibračných signálov nadobutnutých aj prieskumnými meraniami. V druhej polovici semestra sme tak vedeli zostaviť osnovu a každý týždeň sa venovať jednej sekcií analýzy až do 12. týždňa.

## A.2 Letný semester

Obdobie	Náplň práce
1. týždeň	Tvorba generátora syntetického signálu s mechanizmom vyhodnocovania metrik klasifikácie detektorov.
2. týždeň	Pripravenie vývojového prostredia s ESP-IDF SDK a výber vhodných knižníc pre DSP a Message Pack.
3. týždeň	Odladžovanie ovládania hardvérových periférií: akcelerometer, pripojenie na WiFi. Návrh krokov spracovania údajov.
4. týždeň	Zakomponovanie posielania vzoriek cez MQTT. Validácia na syntetických dátach rozdelených na trénovaciu a testovaciu sadu.
5. týždeň	Implementácia kostry spracovania na IoT zariadenie. Message Pack serializácia konfigurácie a jej publikovanie cez MQTT.
6. týždeň	Parser prijatej konfigurácie, uloženie a aplikácia nastavení na zariadení. Optimalizácia alokovania dostupnej pamäte.
7. týždeň	Návrh algoritmu na identifikáciu udalostí. Jednoduché jednotkové testy na validáciu funkčnosti systému a prvotné výkonnostné testy. Tvorba doxygen dokumentácie.
8. týždeň	Experimentálne merania pamäťovej a časovej efektivity. Vyhodnocovanie úspešnosti hľadania špičiek podľa hyperparametrov.
9. týždeň	Ilustrácie a diagramy zahrnuté do kapitoly návrhu.
10. týždeň	Písanie textu 3. kapitoly „Návrh riešenia“ a 4. kapitoly „Implementácia“.
11. týždeň	Písanie textu, vyhotovenie grafov a tabuľiek pre zvyšné kapitoly hlavnej časti práce.
12. týždeň	Doplnenie príloh práce, najmä technickej dokumentácie a používateľskej príručky.
13. týždeň	Prezentácia celkového vypracovania vedúcemu práce a zapracovanie pripomienok.

Pôvodný plán vychádzal z trojtýždenných cyklov, kde po každom by bola kompletnejšia časť systému, v skutočnosti sa prirodzene prelínali a dopĺňali. Do konca 3. týždňa sme plánovali odladenie modelov na monitorovanie vibrácií na základe analyzovaných algoritmov a meranie úspešnosti na synteticky generovaných dátach. Do 6. týždňa mal byť funkčný záznam udalostí na pamäťovú kartu vo firmvéri. Do 9. týždňa sa mala uskutočniť optimalizácia posielaných dát a vzdialená konfigurácia. Na posledný beh pripadali experimenty a ich využitie, počas ktorých bol už písaný text práce. Konzultácie raz za dva týždne tvorili kontrolné body, kedy sme konfrontovali plnenie plánu s postupom.

# Príloha B: Technická dokumentácia

## B.1 Doxygen dokumentácia

Nástroj Doxygen zhотовil podľa komentárov v zdrojom kóde prehľadnú technickú dokumentáciu, ktorá je po typografickej úprave súčasťou tejto prílohy.

### B.1.1 Moduly

- **Udalosti** (B.1.2) - Binárne klasifikátory na označenie význačných úrovní v posuvnom okne vzoriek. Zdrojový kód: `events.h`, `events.c`.
- **Akcelerometer** (B.1.3) - Adaptér pre SPI rozhranie senzora LSM9DS1 lineárnej 3D akcelerácie (IMU). Zdrojový kód: `inertial_unit.h`, `inertial_unit.c`
- **Hardvérové adaptéry** (B.1.4) - Rozhrania na komunikáciu s perifériami. Zdrojový kód: `peripheral.h`, `peripheral.c`
- **Spracovanie dát** (B.1.5) - Fázy spracovania zdrojového signálu. Zdrojový kód: `pipeline.h`
  - **Oknové funkcie** - `window.c`
  - **Správa pamäti systému** - `pipeline.c`
  - **Fázy spracovania oknovaného signálu** - `pipeline.c`
  - **Message Pack serializácia** - Serializácia a parsovanie nameraných dát a konfigurácie. `serialize.c`
- **Deskriptívna štatistiká** (B.1.6) - výpočet popisných štatistik. Zdrojový kód: `statistics.h`, `statistics.c`

## B.1.2 Udalosti

Modul s binárnymi klasifikátormi na označenie význačných úrovní v posuvnom okne vzoriek.

### Dátové štruktúry

**struct SpectrumEvent** - Stav udalosti frekvenčného vedierka.

- **SpectrumEventAction action:** Značka vymedzenia udalosti
- **uint32\_t start:** Časová pečiatka alebo poradie posuvného okna, kedy začala aktuálne aktívna udalosť
- **uint32\_t duration:** Trvanie aktívnej udalosti v počte posuvných okien
- **int32\_t last\_seen:** Počet posuvných okien do minulosti, kedy bol detegovaný posledný výskyt špičky vo frekvencii
- **float amplitude:** Priemerná amplitúda frekvenčného vedierka počas trvania udalosti

### Enumeračie

**enum SpectrumEventAction** - Časové vymedzenie udalosti frekvenčného spektra.

- **SPECTRUM\_EVENT\_NONE:** Udalosť prebieha alebo žiadna nie je aktívna
- **SPECTRUM\_EVENT\_START:** Značka začiatku udalosti
- **SPECTRUM\_EVENT\_FINISH:** Značka ukončenia udalosti

### Funkcie

```
void find_peaks_above_threshold (
    bool *peaks, const float *y, int n, float t
)
```

Hľadanie špičiek absolútnej prahovou úrovňou amplitúdy signálu.

#### Parametre:

- **peaks** (out): Nájdené špičky v signále. Dĺžka poľa musí byť rovnaká počtu vzoriek
- **y** (in): Vzorky signálu
- **n** (in): Počet vzoriek

- **t** (in) Prahová úroveň amplitúdy. Odporúčaná hodnota je z prípustných hodnôt rozsahu pre vzorky

---

```
void find_peaks_neighbours (
    bool *peaks, const float *y, int n, int k,
    float e, float h_rel, float h
)
```

Hľadanie špičiek s algoritmom význačnosti vrchola spomedzi susedov.

$$f[t-i] < f[t] > f[t+i], \quad \forall i \in 1, 2, \dots, k$$

#### Parametre:

- **peaks** (out): Nájdené špičky v signále. Dĺžka poľa musí byť rovnaká počtu vzoriek
- **y** (in): Vzorky signálu
- **n** (in): Počet vzoriek
- **k** (in): Počet najbližších uvažovaných susedov na každú zo strán od kandidátnej špičky  $[t-k; t+k]$ ; Rozsah:  $[1, n/2]$
- **e** (in): Relatívna tolerancia pre vyššiu úroveň v susedstve od vrchola
- **h\_rel** (in): Minimálna relatívna výška špičky v susedstve
- **h** (in): Absolútne prahová úroveň amplitúdy špičky

---

```
void find_peaks_zero_crossing (
    bool *peaks, const float *y, int n, int k, float slope
)
```

Hľadanie špičiek s algoritmom prechodu nulou do záporu.  $\Delta f[i] = 0$

#### Parametre:

- **peaks** (out): Nájdené špičky v signále. Dĺžka poľa musí byť rovnaká počtu vzoriek
- **y** (in): Vzorky signálu
- **n** (in): Počet vzoriek
- **k** (in): Dĺžka sečnice na každú stranu od kandidátnej špičky  $[t-k; t+k]$ ; Rozsah:  $[1, n/2]$
- **slope** (in): Prahová úroveň strmosti kopca, čiže rozdielu medzi hladiny medzi koncami sečnice. Rozsah:  $slope \geq 0$

```
void find_peaks_hill_walker (
    bool *peaks, const float *y, int n,
    float tolerance, int hole,
    float prominence, float isolation
)
```

Hľadanie špičiek s modifikovaným algoritmom horského turistu.

**Parametre:**

- **peaks** (out): Nájdené špičky v signále. Dĺžka poľa musí byť rovnaká počtu vzoriek
- **y** (in): Vzorky signálu
- **n** (in): Počet vzoriek
- **tolerance** (in): Prahová úroveň vo vertikálnej osi. Rozsah:  $[\min(y), \max(y)]$
- **hole** (in): Prahová úroveň v horizontálnej osi. Rozsah:  $[0, n]$
- **prominence** (in): Relatívna výška oproti predošej navštívenej doline. Rozsah:  $[0, \max(y) - \min(y)]$
- **isolation** (in): Vzdialenosť ku najbližšiemu predošlému vrcholu. Rozsah:  $[0, n]$

---

```
void event_init (SpectrumEvent *events, uint16_t bins)
```

Nastavenie počiatocného stavu online detektora udalostí vo frekvenciach.

**Parametre:**

- **events** (out): Pole udalostí frekvenčného spektra
- **bins** (in): Počet frekvenčných vedierok a zároveň dĺžka poľa udalostí

---

```
size_t event_detection (
    size_t t, SpectrumEvent *events, const bool *peaks,
    const float *spectrum, uint16_t bins,
    uint16_t min_duration, uint16_t time_proximity
)
```

Online detekcia zmien v časovom priebehu frekvenčných spektier.

**Parametre:**

- **t**: Poradové číslo posuvného okna. S každým ďalším volaním funkcie musí byť navýšené o 1
- **events**: Pole udalostí frekvenčného spektra s dĺžkou počtu vedierok
- **peaks** (in): Nájdené špičky vo aktuálnom frekvenčnom spektri jedným z klasifikátorov *find\_peak\_\**
- **spectrum** (in): Frekvenčné spektrum aktuálneho posuvného okna vzoriek na

zistenie priemernej amplitúdy udalostí

- **bins** (in): Počet frekvenčných vedierok
  - **min\_duration** (in): Minimálne trvanie po koľkých oknách je vyhlásená špička za udalosť. Udáva oneskorenie vyhlásenia začiatku udalosti.
  - **time\_proximity** (in): Najväčšia vzdialenosť súvislej udalosti v počte okien. Najväčšia dĺžka časovej medzery medzi nájdenými špičkami. Udáva oneskorenie vyhlásenia ukončenia udalosti.
  - **Návratová hodnota:** Počet detegovaných zmien, čiže začiatočných a koncov udalostí v danom spektre posuvného okna
- 

### B.1.3 Akcelerometer

Modul adaptéra pre SPI rozhranie senzora LSM9DS1 lineárnej 3D akcelerácie (IMU)

#### Dátové štruktúry

**struct InertialUnit** - Inerciálna meracia jednotka.

- `gpio_num_t clk`: GPIO pin SPI hodinového signálu
- `gpio_num_t miso`: GPIO pin SPI Master In Slave Out
- `gpio_num_t xgcs`: GPIO pin SPI Chip select akcelometra a gyroskopu
- `gpio_num_t mcs`: GPIO pin SPI Chip select magnetometra
- `gpio_num_t int1`: GPIO vstup prerušenia č.1
- `gpio_num_t int2`: GPIO vstup prerušenia č.2
- `gpio_num_t en_data`: GPIO vstup príznaku pripravených dát
- `gpio_num_t isr_int1`: Podprogram prerušenia pre INT č.1
- `gpio_num_t isr_int2`: Podprogram prerušenia pre INT č.2
- `spi_host_device_t spi`: SPI zbernice
- `spi_device_handle_t dev`: SPI zariadenie pre akcelerometer
- `AccelerationPrecision precision`: Citlivosť akcelometra v mg/LSB

#### Enumerácie

**enum AccelerationRange** - Dynamický rozsah akcelerometra v g.

- `IMU_2G`: Rozsah  $\pm 2 \text{ g} = \pm 19.6133 \text{ m/s}^2$
- `IMU_4G`: Rozsah  $\pm 4 \text{ g} = \pm 39.2266 \text{ m/s}^2$

- IMU\_8G: Rozsah  $\pm 8 \text{ g} = \pm 78.4532 \text{ m/s}^2$
- IMU\_16G: Rozsah  $\pm 16 \text{ g} = \pm 156.9064 \text{ m/s}^2$
- IMU\_RANGE\_COUNT: Počet možností nastavenia rozsahu na účely serializácie

**typedef float AccelerationPrecision** - Citlivosť akcelerometra podľa dynamického rozsahu v mg/LSB.

## Funkcie

**esp\_err\_t imu\_setup(InertialUnit \*imu)**

Inicializácia senzora lineárnej akcelerácie.

### Parametre:

- **imu**: Senzor
- **Návratová hodnota**: Úspešnosť inicializácie senzora

---

```
void imu_acceleration_range (
    InertialUnit *imu, AccelerationRange range
)
```

Nastavenie dynamického rozsahu lineárneho 3D akcelerometra v  $g$ .

### Parametre:

- **imu**: Senzor
- **range**: Dynamický rozsah akcelerometra

---

```
void imu_output_data_rate (InertialUnit *imu, uint16_t fs)
```

Nastavenie výstupného dátového toku (ODR) akcelerometra podľa vzorkovanej frekvencie.

### Parametre:

- **imu**: Senzor
- **fs**: Vzorkovacia frekvencia v Hz. Hardvér povolojuje max. ODR 956 Hz

---

```
void imu_acceleration (
    InertialUnit *imu, float *x, float *y, float *z
)
```

Meranie aktuálnej hodnoty 3D akcelerácie v  $m/s^2$ .

### Parametre:

- **imu**: Senzor
- **x (out)**: Zrýchlenie v osi x. Rozsah je podľa nastavenia dynamického rozsahu.

- **y** (out): Zrýchlenie v osi y
- **z** (out): Zrýchlenie v osi z

### B.1.4 Hardvérové adaptéry

#### Dátové štruktúry

**struct OpenLog** - SparkFun OpenLog - zaznenávač údajov na SD kartu cez sériovú linku.

- `gpio_num_t vcc`: GPIO pin na ovládanie napájania cez FET tranzistor
- `uint8_t uart`: Číslo UART rozhrania
- `gpio_num_t rx`: GPIO pin UART RX
- `gpio_num_t tx`: GPIO pin UART TX
- `int baudrate`: Symbolová rýchlosť komunikácie. Rovnaká rýchlosť musí byť nastavená v `config.txt` na SD karte
- `int buffer`: Dĺžka vyrovnávacej pamäte pre UART

**struct MqttAxisTopics** - MQTT témy pre os zrýchlenia.

- `char stats[TOPIC_LENGTH]`: Názov témy pre štatistické údaje
- `char spectra[TOPIC_LENGTH]`: Názov témy pre frekvenčné spektrum
- `char events[TOPIC_LENGTH]`: Názov témy pre udalosti zmeny spektra

#### Funkcie

**void axis\_mqtt\_topics (MqttAxisTopics \*topics, int axis)**

Poskladanie názvu MQTT tému pre odosianie dát o osi akcelerácie.

#### Parametre:

- **topics** (out): MQTT témy zložené s označením osi x, y, z
- **axis** (in): Index osi vektora akcelerácie: 0, 1, 2

---

**void clock\_reconfigure (uint16\_t frequency)**

Zmena frekvencie časovača.

#### Parametre:

- **frequency** (in): Vzorkovacia frekvencia v Hz

---

```
void clock_setup (uint16_t frequency, timer_isr_t action)
```

Spustenie časovača na vzorkovanie signálu.

**Parametre:**

- **frequency** (in): Vzorkovacia frekvencia v Hz
- **action** (in): Obsluha prerušenia časovača s predpisom:

```
bool IRAM_ATTR f(void *args)
```

---

```
void mqtt_event_handler (
    void *handler_args, esp_event_base_t base,
    int32_t event_id, void *event_data
)
```

Predbežná deklarácia spätného volania. Implementáciu musí poskytnúť hlavný program. Používa sa v mqtt\_setup().

---

```
esp_mqtt_client_handle_t mqtt_setup (
    const char *broker_url
)
```

Pripojenie sa k MQTT broker a zaregistrovanie spätného volania pre všetky udalosti.

**Parametre:**

- **broker\_url**: URL MQTT broker

---

```
esp_err_t nvs_load (
    Configuration *conf, Provisioning *login
)
```

Načítanie nastavení systému z nevolatilného úložiska.

**Parametre:**

- **conf** (out): Globálne nastavenia spracovania dát
- **login** (out): Nastavenie sietového pripojenia

---

```
esp_err_t nvs_save_config (const Configuration *conf)
```

Uloženie nastavení systému na nevolatilné úložisko.

Používa sa v mqtt\_event\_handler().

**Parametre:**

- **conf** (in): Globálne nastavenia spracovania dát

---

```
esp_err_t nvs_save_login (const Provisioning *login)
```

Uloženie nastavení sietového pripojenia na nevolatilné úložisko.

Používa sa v mqtt\_event\_handler().

**Parametre:**

- **login** (in): Nastavenie sietového pripojenia
- 

```
void openlog_setup (OpenLog *logger)
```

Nastavenie UART rozhrania pre zariadenie OpenLog.

---

```
void wifi_connect (wifi_config_t *wifi_config)
```

Pripojenie sa k Wifi AP blokujúce.

## B.1.5 Spracovanie dát

### Dátové štruktúry

**struct SamplingConfig** - Nastavenie vzorkovania signálu.

- **uint16\_t frequency**: Vzorkovacia frekvencia v Hz. Najviac MAX\_FREQUENCY
- **AccelerationRange range**: Dynamický rozsah akcelerometra
- **uint16\_t n**: Veľkosť posuvného okna. Musí byť mocninou dvojky a najviac MAX\_BUFFER\_SAMPLES
- **float overlap**: Pomer prekryvu posuvných okien. Rozsah: 0 až MAX\_OVERLAP
- **bool axis[AXIS\_COUNT]**: Osi akcelerácie povolené na spracovanie

**struct SmoothingConfig** - Nastavenie vyhľadzovania časovo premenného signálu alebo frekvenčného spektra.

- **bool enable**: Vyhľadzovanie signálu povolené
- **uint16\_t n**: Dĺžka konvolučnej masky
- **uint8\_t repeat**: Počet prechodov konvolučnej masky.

Najviac MAX\_SMOOTH\_REPEAT

**struct StatisticsConfig** - Nastavenie zberu štatistik posuvného okna.

- **bool min**: Výpočet minimálnej hodnoty povolený
- **bool max**: Výpočet maximálnej hodnoty povolený
- **bool rms**: Výpočet strednej kvadratickej odchýlky povolený

- `bool mean`: Výpočet aritmetického priemeru povolený
- `bool variance`: Výpočet rozptylu povolený
- `bool std`: Výpočet smerodajnej odchýlky povolený
- `bool skewness`: Výpočet šikmosti povolený
- `bool kurtosis`: Výpočet špicatosti povolený
- `bool median`: Výpočet mediánu povolený
- `bool mad`: Výpočet mediánovej absolútnej odchýlky povolený
- `bool correlation`: Výpočet korelácie medzi osami povolený

**struct FFTTransformConfig** - Nastavenie frekvenčnej transformácie.

- `WindowTypeConfig window`: Oknová funkcia
- `FrequencyTransform func`: Typ frekvenčnej transformácie
- `bool log`: Prevod magnitúdy frekvencie do dB

**struct SaveFormatConfig** - Nastavenia ukladania a posielania spracovaných dát.

- `bool local`: Záznam vzoriek na SD kartu povolený. OpenLog bude zapnutý po spustení
- `bool mqtt`: Posielanie cez MQTT povolené. Wifi a MQTT klient bude zapnutý po spustení. Pozor: po deaktivácii sa zariadenie nedá vzdialene rekonfigurovať. Na znova povolenie sa musí nahrať firmvér so touto možnosťou povolenou.
- `bool mqtt_stats`: Odosielanie štatistik cez MQTT na topic podľa MQTT\_TOPIC\_STATS
- `bool mqtt_events`: Odosielanie zmien spektra cez MQTT na topic podľa MQTT\_TOPIC\_EVENT
- `SendUnprocessed mqtt_samples`: Odosielanie nespracovaných vzoriek alebo frekvencií cez MQTT na topic podľa MQTT\_TOPIC\_STREAM, MQTT\_TOPIC\_SPECTRUM.
- `uint16_t subsampling`: Podvzorkovanie pre záznam vzoriek bez ďalšieho spracovania. Preskočí sa každých `subsample` vzoriek

**struct FFTTransformConfig** - Nastavenia algoritmov na detekciu udalostí a ich parametrov (popis sa nachádza pri funkciách z modulu „Udalosti“ B.1.2).

**struct Configuration** - Systémová konfigurácia spracovania vzoriek z akcelerometra.

**struct Provisioning** - Sieťové nastavenia pre pripojenie na Wifi AP s WPA2 a MQTT broker.

**struct Correlation** - Medzivýsledky korelácie zdieľanej všetkými osami spracovania s prístupom cez zahrnutú synchronizačnú bariéru.

**struct Statistics** - Výsledky všetkých dostupných štatistik.

**struct BufferPipelineKernel** - Vyrovnávacie pamäte spoločné pre všetky spracovateľské úlohy.

**struct BufferPipelineAxis** - Vyrovnávacie pamäte samostatné pre každú os akcelerácie.

**struct Sender** - Rad pre záznam vzoriek bez ďalšieho spracovania.

## Konštanty

V zátvorkách sú uvedené predvolené hodnoty

- **AXIS\_COUNT:** Celkový počet osí akcelerácie (3)
- **MAX\_MPACK\_FIELDS\_COUNT:** Maximálny počet dvojíc „klúč - hodnota” v Message Pack slovníku (20)
- **SAMPLES\_QUEUE\_SLOTS:** Násobok veľkosti posuvného okna ako počet vzoriek čakajúcich na spracovanie v rade (3)
- **MAX\_CREDENTIALS\_LENGTH:** Maximálna dĺžka prihlásovacieho údaju Wifi pripojenia (64)
- **MAX\_MQTT\_URL:** Dĺžka URL adresy na MQTT broker (256)
- **MAX\_BUFFER\_SAMPLES:** Najdlhšie povolené posuvné okno, vyššia mocnina 2 ako 1024 sa nezmestí do DRAM (1024)
- **MAX\_FREQUENCY:** Najvyššia vzorkovacia frekvencia daná fyzickým obmedzením akcelerometra (952)
- **MAX\_OVERLAP:** Maximálny prekryv posuvných okien (0.8)

- MAX\_SMOOTH\_REPEAT: Maximálny počet prechodu konvolučnej masky výhľadzovacieho filtra (8)
- LARGEST\_MESSAGE: Najväčšia veľkosť vyrovňávacej pamäte pre serializáciu vzoriek (14000)
- LARGEST\_CONFIG: Najväčšia veľkosť serializovanej konfigurácie (480)

## Enumerácie

**enum WindowTypeConfig** - Oknové funkcie.

- BOXCAR\_WINDOW: Obdĺžníkové okno
- BARTLETT\_WINDOW: Bartlettovo okno
- HANN\_WINDOW: Hannovo okno
- HAMMING\_WINDOW: Hammingovo okno
- BLACKMAN\_WINDOW: Blackmanovo okno
- WINDOW\_TYPE\_COUNT: Počet dostupných oknových funkcií. Potrebné pre serializáciu.

**enum PeakFindingStrategy** - Algoritmy na hľadanie špičiek.

- THRESHOLD: Špičky nad prahovou úrovňou.
- NEIGHBOURS: Špičky najvýznačnejšieho bodu spomedzi susedov.
- ZERO\_CROSSING: Špičky prechodou nulou do záporu.
- HILL\_WALKER: Špičky algoritmom horského turista.
- STRATEGY\_COUNT: Počet možností na účely serializácie

**enum FrequencyTransform** - Frekvenčné transformácie.

- DFT: Rýchla Fourierová transformácia radix-2
- DCT: Konsínusová transformácia DCT-II
- TRANSFORM\_COUNT: Počet dostupných frekvenčných transformácií. Potrebné pre serializáciu

**enum SendUnprocessed** - Doména odosielaných nespracovaných vzoriek.

- RAW\_NONE\_SEND: Žiadne nespracované vzorky
- RAW\_TIME\_SEND: Nespracované vzorky v časovej oblasti
- RAW\_FREQUENCY\_SEND: Nespracované vzorky vo frekvenčnej oblasti
- SEND\_UNPROCESSED\_COUNT: Počet možností na účely serializácie

## Oknové funkcie

Parametre všetkých oknových funkcií:

- **w** (out): Váhy oknovej funkcie
- **n** (in): Dĺžka okna

**void bartlett\_window (float \*w, int n)**

Bartlettovo okno.  $w(n) = \frac{2}{N-1} \left( \frac{N-1}{2} - \left| n - \frac{N-1}{2} \right| \right)$

**void blackman\_window (float \*w, int n)**

Blackmanovo okno.  $w(n) = 0.42 - 0.5 \cos(2\pi n/N) + 0.08 \cos(4\pi n/N)$

**void boxcar\_window (float \*w, int n)**

Obdĺžníkové okno.  $w(n) = 1$

**void hamming\_window (float \*w, int n)**

Hammingovo okno.  $w(n) = 0.54 - 0.46 \cos(2\pi n/N)$

**void hann\_window (float \*w, int n)**

Hannovo okno.  $w(n) = \sin^2(\pi n/N)$

**void mean\_kernel (float \*w, int n)**

Vyhľadzovací filter kľzavého priemeru.  $w(n) = \frac{1}{n}$ ,  $n = 0, 1, \dots, N-1$

**void window (WindowTypeConfig type, float \*w, int n)**

Oknová funkcia podľa voľby.

### Parametre

- **type** (out): Oknová funkcia
- **w** (out): Váhy oknovej funkcie
- **n** (in): Dĺžka okna

## Správa pamäti systému

```
void axis_allocate (
    BufferPipelineAxis *p, const Configuration *conf
)
```

Alokácia vyrovnávacích pamäťí pre jednu os akcelerácie.

### Parametre:

- **p** (out): Dynamické vyrovnávacie pamäte s dĺžkami podľa nastavení
  - **conf** (in): Konfigurácia systému
- 

**void axis\_release (BufferPipelineAxis \*p)**

Uvoľnenie vyrovnávacích pamäťí pre jednu os akcelerácie.

**Parametre:**

- **p** (out): Dynamické vyrovnávacie pamäte
- 

**void process\_allocate (**  
    **BufferPipelineKernel \*p, const Configuration \*conf**  
**)**

Alokácia a inicializácia spoločných vyrovnávacích pamäťí a synchronizačných primitív.

**Parametre:**

- **p** (out): Dynamické vyrovnávacie pamäte s dĺžkami podľa nastavení
  - **conf** (in): Konfigurácia systému
- 

**void process\_release (BufferPipelineKernel \*p)**

Uvoľnenie spoločných vyrovnávacích pamäťí.

**Parametre:**

- **p** (out): Dynamické vyrovnávacie pamäte
- 

**void axis\_release (BufferPipelineAxis \*p)**

Uvoľnenie vyrovnávacích pamäťí pre jednu os akcelerácie.

**Parametre:**

- **p** (out): Dynamické vyrovnávacie pamäte
- 

**void sender\_allocate (Sender \*sender, uint16\_t length)**

Vytvorenie radu na odosielanie nameraných vzoriek odlišnou systémovou úlohou.

**Parametre**

- **sender**: Vyhradený rad s požadovanou maximálnou kapacitou
  - **length**: Dĺžka radu
- 

**void sender\_release (Sender \*sender)**

Odstránenie radu na odosielanie nameraných vzoriek.

**Parametre**

- **sender**: Rad s vyhradenou kapacitou
-

## Fázy spracovania oknovaného signálu

---

```
void buffer_shift_left(
    float *buffer, uint16_t n, uint16_t k
)
```

Posun vzoriek vo vyrovňávacej pamäti doľava, čím sa dosahuje prekryv okien. Nadbytočné hodnoty od začiatku poľa budú nahradené vzorkami o  $k$  pozícii vpravo.

### Parametre:

- **buffer**: Vyrovňávacia pamäť, ktorej obsah bude posunutý
  - **n** (in): Dĺžka vyrovňávacej pamäte
  - **k** (in): Počet pozícii o koľko sa majú posunúť hodnoty.
- 

```
void process_correlation(
    uint8_t axis, const float *buffer,
    Statistics *stats, Correlation *corr,
    const SamplingConfig *c
)
```

Korelácia medzi osami akcelerácie: XY, XZ, YZ. Dochádza k bariérovej synchronizácii. Úloha pre každú os zrýchlenia si nezávisle prepočíta rozdiely vzoriek od priemeru a smerodajné odchýlky. Následne dochádza k bariérovej synchronizácii aktívnych osí. Každá úloha si dopočíta všetky korelácie samostatne.

### Parametre:

- **axis** (in): Os akcelerácie: 0, 1, 2
  - **buffer** (in): Posuvné okno vzoriek signálu
  - **stats** (out): Zistené medzi-osové korelácie
  - **corr** (out): Pomocné polia pre výmenu predspracovaných údajov medzi úlohami (osami)
  - **corr** (in): Nastavenia vzorkovania. Využíva sa dĺžka posuvného okna a povoľené osi.
- 

```
void process_smoothing(
    float *buffer, float *tmp, uint16_t n,
    const float *kernel, const SmoothingConfig *c
)
```

Vyhľadzovanie signálu.

### Parametre:

- **buffer**: Posuvné okno vzoriek signálu s dĺžkou  $n$ , ktoré bude vyhľadené

## PRÍLOHA B. TECHNICKÁ DOKUMENTÁCIA

---

- **tmp**: Pomocné pole o dĺžke  $n + c \cdot n - 1$
  - **n** (in): Dĺžka posuvného okna
  - **kernel** (in): Konvolučná maska vyhladzovania
  - **c** (in): Nastavenia vyhladzovanie
- 

```
int process_spectrum(
    float *spectrum, const float *buffer,
    const float *window, uint16_t n,
    const FFTTransformConfig *c
)
```

Frekvenčné spektrum (FFT, FCT) posuvného okna vzoriek vynásobené váhami oknovej funkcie.

Parametre:

- **spectrum** (out): Frekvenčné spektrum s dĺžkou  $n/2$
  - **buffer** (in): Posuvné okno vzoriek signálu s dĺžkou  $n$
  - **window** (in): Váhy oknovej funkcie s dĺžkou  $n$
  - **n** (in): Dĺžka posuvného okna
  - **c** (in): Nastavenia frekvenčnej transformácie
- 

```
void process_statistics(
    const float *buffer, uint16_t n,
    Statistics *stats, const StatisticsConfig *c
)
```

Požadované štatistiky podľa nastavení.

Parametre:

- **buffer** (in): Posuvné okno vzoriek signálu
  - **n** (in): Dĺžka posuvného okna
  - **stats** (out): Deskriptívne štatistiky zo vzoriek posuvného okna. Korektné hodnoty majú len tie povolené v nastaveniach ‘c’
  - **c** (in): Povolenia pre zber vybraných štatistik
- 

```
void process_peak_finding(
    bool *peaks, const float *spectrum,
    uint16_t bins, const EventDetectionConfig *c
)
```

Hľadanie špičiek vo frekvenčnom spektre podľa nastavení aktívneho algoritmu.

Parametre:

- **peaks** (out): Váhy oknovej funkcie s dĺžkou bins
  - **spectrum** (in): Frekvenčné spektrum s dĺžkou bins
  - **bins** (in): Počet frekvenčných vedierok
  - **c** (in): Nastavenia spracovania udalostí
- 

## Message Pack serializácia

```
size_t stream_serialize(
    char *msg, size_t size,
    const float *stream, size_t n
)
```

Serializácia prúdu vzoriek v posuvnom okne do formátu Message Pack.

### Parametre:

- **msg** (out): Serializované vzorky signálu
  - **size** (in): Vyhradená veľkosť pre správu do msg
  - **stream** (in): Vzorky signálu
  - **n** (in): Počet vzoriek signálu
  - **Návratová hodnota:** Dĺžka serializovanej správy
- 

```
size_t stats_serialize(
    size_t timestamp, char *msg, size_t size,
    const Statistics *stats, const StatisticsConfig *c
)
```

Serializácia štatistik signálu v posuvnom okne do formátu Message Pack.

### Parametre:

- **timestamp** (in): Poradové číslo posuvného okna
  - **msg** (out): Serializované štatistiky
  - **size** (in): Vyhradená veľkosť pre správu do msg
  - **stats** (in): Štatistiky signálu
  - **c** (in): Nastavenia zberu štatistik. Do serializovanej správy sa zahrnú len aktívne štatistiky.
  - **Návratová hodnota:** Dĺžka serializovanej správy
- 

```
size_t spectra_serialize(
    size_t timestamp, char *msg, size_t size,
    const float *spectrum, size_t n, uint16_t fs
)
```

## PRÍLOHA B. TECHNICKÁ DOKUMENTÁCIA

---

Serializácia frekvenčného spektra posuvného okna do formátu Message Pack.

### Parametre

- **timestamp** (in): Poradové číslo posuvného okna
  - **msg** (out): Serializované frekvenčné spektrum
  - **size** (in): Vyhradená veľkosť pre správu do msg
  - **spectrum** (in): Frekvenčné spektrum
  - **n** (in): Počet frekvenčných vedierok
  - **fs** (in): Vzorkovacia frekvencia v Hz
  - **Návratová hodnota:** Dĺžka serializovanej správy
- 

```
size_t events_serialize(  
    size_t timestamp, float bin_width,  
    char *msg, size_t size,  
    const SpectrumEvent *events, size_t n  
)
```

Serializácia udalostí zmien frekvenčného spektra do formátu Message Pack.

### Parametre:

- **timestamp** (in): Poradové číslo posuvného okna
  - **bin\_width** (in): Veľkosť frekvenčného vedierka v Hz:  $fs/n$
  - **msg** (in): Serializované udalosti
  - **size** (in): Vyhradená veľkosť pre správu do msg
  - **events** (in): Udalosti frekvenčného spektra s dĺžkou n. Do správy budú pridané iba začiatočné a ukončujúce udalosti.
  - **n** (in): Počet frekvenčných vedierok
  - **Návratová hodnota:** Dĺžka serializovanej správy
- 

```
size_t config_serialize(  
    char *msg, size_t size,  
    const Configuration *config  
)
```

Serializácia systémovej konfigurácie do formátu Message Pack.

### Parametre:

- **msg** (out): Serializovaná konfigurácia
- **size** (in): Vyhradená veľkosť pre správu do msg
- **config** (in): Systémová konfigurácia
- **Návratová hodnota:** Dĺžka serializovanej správy

```
bool config_parse(
    const char *msg, int size,
    Configuration *conf, bool *error
)
```

Parsovanie systémovej konfigurácie z formátu Message Pack.

**Parametre:**

- **msg** (in): Serializovaná konfigurácia
- **size** (in): Dĺžka konfigurácie v Message Pack
- **conf** (out): Systémová konfigurácia
- **error** (out): Chyba pri parsovaní
- **Návratová hodnota:** Zmena konfigurácie oproti pôvodnému obsahu conf

---

```
size_t login_serialize(
    char *msg, size_t size,
    const Provisioning *conf
)
```

Serializácia údajov o sietovom pripojení.

**Parametre:**

- **msg** (out): Serializované nastavenia pripojenia
- **size** (in): Vyhradená veľkosť pre správu do msg
- **config** (in): Nastavenia pripojenia
- **Návratová hodnota:** Dĺžka serializovanej správy

---

```
bool login_parse(
    const char *msg, size_t size,
    Provisioning *conf
)
```

Parsovanie nastavení sietového pripojenia z formátu Message Pack.

**Parametre:**

- **msg** (in): Serializované konfigurácia
- **size** (in): Vyhradená veľkosť pre správu do msg
- **conf** (out): Nastavenia pripojenia
- **Návratová hodnota:** Chyba pri parsovaní

---

## B.1.6 Deskriptívna štatistika

Rovnako označené parametre funkcií:

- **x** (in): Vzorky signálu
- **n** (in): Počet vzoriek signálu

## Funkcie

**float minimum(const float \*\*x, int n)**

Najnižšia hodnota.

### Parametre:

- **Návratová hodnota:** Minimum z hodnôt signálu

---

**float maximum(const float \*\*x, int n)**

Najvyššia hodnota.

### Parametre:

- **Návratová hodnota:** Maximum z hodnôt signálu

---

**float root\_mean\_square(const float \*\*x, int n)**

Stredná kvadratická odchýlka.

### Parametre:

- **Návratová hodnota:** RMS z hodnôt signálu

---

**float mean(const float \*\*x, int n)**

Aritmetický výberový priemer.

### Parametre:

- **Návratová hodnota:** Priemer hodnôt signálu

---

**float variance(const float \*\*x, int n, float mean)**

Rozptyl populácie (vychýlená štatistika).

### Parametre:

- **mean** (in): Aritmetický priemer signálu
- **Návratová hodnota:** Rozptyl hodnôt signálu

---

**float standard\_deviation(float variance)**

Smerodajná odchýlka.

### Parametre:

- **variance** (in): Rozptyl signálu
- **Návratová hodnota:** Smerodajná odchýlka hodnôt signálu

---

```
float moment(const float *x, int n, int m, float mean)
```

Centrálny moment rádu m.

**Parametre:**

- **m** (in): Rád centrálneho momentu. Kladné číslo väčšie ako 1.
  - **mean** (in): Aritmetický priemer signálu
  - **Návratová hodnota:** Centrálny moment
- 

```
float skewness(const float *x, int n, float mean)
```

Šikmost'.

**Parametre:**

- **mean** (in): Aritmetický priemer signálu
  - **Návratová hodnota:** Šikmost'
- 

```
float kurtosis(const float *x, int n, float mean)
```

Špicatosť.

**Parametre:**

- **mean** (in): Aritmetický priemer signálu
  - **Návratová hodnota:** Špicatosť
- 

```
float correlation(
    const float *x_diff, const float *y_diff, int n,
    float x_std, float y_std
)
```

Korelácia z medzivýsledkov.

**Parametre:**

- **x\_diff** (in): Predspracované vzorky prvého signálu odčítané od aritmetického priemeru:  $(x_i - \bar{x})$
  - **y\_diff** (in): Predspracované vzorky druhého signálu odčítané od aritmetického priemeru:  $(y_i - \bar{y})$
  - **n** (in): Počet vzoriek signálu. Dĺžky oboch polí musia byť rovnaké.
  - **x\_std** (in): Smerodajná odchýlka prvého signálu
  - **y\_std** (in): Smerodajná odchýlka druhého signálu
  - **Návratová hodnota:** Pearsonov korelačný koeficient
- 

```
float quickselect(const float *x, int n, int k)
```

Quickselect. Algoritmus na nájdenie k-teho najmenšieho prvku v nezoradenom poli. Aby nedochádzalo k modifikácii poradia pôvodného poľa kopíruje prvky do poľa z premenlivou dĺžkou (Variable-length array) podľa n, na zásobníku.

**Parametre:**

- **k** (in): Rád k-teho najmenšieho prvku
  - **Návratová hodnota:** k-ty najmenší prvk
- 

```
float median(const float *x, int n)
```

Medián cez Quickselect.

**Parametre:**

- **Návratová hodnota:** Medián
- 

```
float median_abs_deviation(
    const float *x, int n, float med
)
```

Mediánová absolútна odchýlka (MAD). Medzi-výsledky odchýlok na nájdenie mediánu ukladá do poľa z premenlivou dĺžkou (Variable-length array) podľa n, na zásobníku

**Parametre:**

- **med** (in): Medián signálu
- **Návratová hodnota:** MAD

## B.2 MQTT témy

Správy s výsledkami spracovania a konfigurácií posielané protokolom MQTT sú kódované v Message Pack. Obsah je tematicky odlišený do zvlášť MQTT topics, s ujednotenou štruktúrou formátu na tému.

Zdrojové zariadenie je navyše identifikovateľné *Device ID* v prefixe názvu témy podľa konštanty DEVICE\_MQTT\_TOPIC v hlavičkovom súbore peripheral.h.

Prefix pre témy z nasledujúceho zoznamu je tvaru imu/1/.

Operácie publikovania (*Publish*) a odberu (*Subscribe*) témy sú uvádzané z pohľadu externého klienta, ktorý má záujem vykonávať zber údajov zo senzorovej jednotky. Štatistiky, frekvenčné spektrum a udalosti sa produkujú pre viaceré priestorové rozmery vektora zrýchlenia, preto okrem osi x sú platné aj y, z.

## Zoznam tém

Príklady štruktúry správ sú z dôvodu čitateľnosti zapísané vo formáte JSON, v skutočnosti sú kódované vo formáte Message Pack.

- **syslog** (*Subscribe*) - Textový reťazec informujúci o stave zariadenia alebo úspechu vyžiadanej akcie:

- ★ „*imu started*”: Mikrokontrolér sa po reštarte pripojil na MQTT broker
- ★ „*config received*”: Konfigurácia bola úspešne prijatá na téme *config/set*
- ★ „*config applied*”: Konfigurácia obsahuje zmenené pravidlá oproti aktuálnym nastaveniam a nahradili sa v nevolatilnej pamäti. Zariadenie bude onedlho reštartované.
- ★ „*config malformed*”: Prijaté pravidlá konfigurácie budú nezodpovedajú požadovanej štruktúre alebo hodnoty sú neprípustné.
- ★ „*login saved*”: Prihlasovacie údaje do siete prijaté na téme *login/set* boli pozmenené uložené na zariadení. Reštart nebude vykonaný, pretože môže následne dôjsť k strate spojenia.
- ★ „*login malformed*”: Prijaté nastavenia sietového pripojenia budú nezodpovedajú požadovanej štruktúre alebo hodnoty sú neprípustné.

- **samples** (*Subscribe*) - Pole nespracovaných vzoriek z akcelerometra v  $m/s^2$ .

Priestorové zložky trojrozmerného vektora sú usporiadane sekvenčne za použitia jednoduchej presnosti float32. V správe sa naraz nachádza  $n/3 + 1$  vektorov. Napríklad pri  $f_s = 8$  sú v poli 3 vektory vzoriek:

```
[-0.078, -0.910, -9.964, -1.773, -16.439, -0.401, 1.499,
 5.202, 1.499]
```

- **stats/x** (*Subscribe*) - Sumárne aktívne štatistiky posuvného okna s poradovým číslom  $t$ . Názvy atribútov zodpovedajú príslušným pravidlám konfigurácie, ktoré ich povoľujú. Korelácia je vyjadrená zo všetkých spracúvaných párov dimenzií.

```
{
  "t": 1,
  "min": -9.964, "max": 11.846, "rms": 8.343,
  "avg": 4.126, "std": 7.251, "skew": -0.614,
  "kurt": -0.821, "med": 5.773, "mad": 5.370,
  "corrXY": 0.528, "corrXZ": 0.648, "corrYZ": 0.431
}
```

- **spectrum/x** (*Subscribe*) - Výstup frekvenčnej transformácie v posuvnom okne s poradím  $t$ . Počet komponentov  $bins$  je polovicou dĺžky transformovanej postupnosti. Vzorkovacia frekvencia  $f_s$  slúži na vyjadrenie šírky frekvenčného vedierka.

```
{  
    "t": 1,  
    "fs": 8,  
    "bins": [0.000, -0.026, -38.772, -38.195]  
}
```

- **events/x** (*Subscribe*) - Ohlásenie začiatkov  $A$  alebo koncov  $Z$  zmien spektrálneho obsahu signálu zistené prúdovým algoritmom na detekciu udalostí v posuvnom okne s poradím  $t$  a šírkou frekvenčného vedierka  $df$ . Udalosť významnej frekvencie sa vyznačuje pozíciou vedierka  $i$  (skutočná frekvencia je potom  $f = i \cdot df$ ), začiatkom v okne s poradím  $t$ , trvaním  $d$  a priemernou amplitúdou  $h$ .

```
{  
    "t": 310,  
    "df": 2.0,  
    "A": [{"i": 2, "t": 305, "d": 5, "h": -5.621}],  
    "Z": []  
}
```

- **config/set** (*Publish*) - Nastavenie systémových pravidiel spracovania. Dokáže aplikovať čiastkové úpravy celkovej konfigurácie detailne vypísanej v *config/response*. Napríklad, zmena stratégie hľadaniu špičiek:

```
{"peak": {"strategy": "zero_crossing"}}
```

- **config/request** (*Publish*) - Dopyt aktuálne prítomnej konfigurácie s prázdnym obsahom. Odpoveď sa očakáva na tému *config/response*.
- **config/response** (*Subscribe*) - Serializovaná konfigurácia zodpovedajúca vlastnostiam uložením v štruktúre Configuration za rovnakých obmedzení. Enumerácie sa pretvárajú na reťazce s významom priamo vychádzajúcim z pôvodných konštánt:

- ★ Dynamický rozsah senzora „range“: "2g", "4g", "8g", "16g"
- ★ Názvy oknových funkcií: "boxcar", "bartlett", "hann", "hamming", "blackman"
- ★ Transformácie do frekvenčnej domény: "dft", "dct"

- ★ Algoritmy hľadania špičiek „strategy“ sa nazývajú rovnako ako skupiny možností ich parametrov: "threshold", "neighbours", "zero\_crossing", "hill\_walker"

- ★ Odosielané nespracované údaje („logger“) „samples“ sú v časovej doméne "t", frekvenčnej doméne "f", alebo nie posielané "".

```
{
  "sensor": {
    "fs": 476, "range": "2g",
    "n": 256, "overlap": 0.5,
    "axis": [true, true, true]
  },
  "tsmooth": {"on": false, "n": 8, "repeat": 1},
  "stats": {
    "min": true, "max": true, "rms": true,
    "avg": true, "var": true, "std": true,
    "skew": true, "kurt": true, "med": true,
    "mad": true, "corr": false
  },
  "transform": {"w": "hann", "f": "dft", "log": true},
  "fsmooth": {"on": false, "n": 8, "repeat": 1},
  "peak": {
    "tmin": 4, "tprox": 5,
    "strategy": "threshold",
    "threshold": {
      "t": -15.0
    },
    "neighbours": {
      "k": 9, "e": 0.0, "h": -100.0, "h_rel": 10.0
    },
    "zero_crossing": {
      "k": 4, "slope": 3.0
    },
    "hill_walker": {
      "t": 0.0, "h": 0, "p": 10.0, "i": 3.0
    }
  },
  "logger": {
    "local": false, "mqtt": true,
    "samples": "t", "subsamp": 1,
    "stats": true, "events": true
  }
}
```

- **login/set (Publish)** - Nastavenie sieťového pripojenia pred premiestnením zariadenia. Dokáže aplikovať čiastkové úpravy z atribútov *login/response*.

- **login/request** (*Publish*) - Dopyt aktuálnych informácií o sietovom pripojení s prázdnym obsahom. Odpoveď sa očakáva na tému *login/response*.
- **login/response** (*Subscribe*) - Aktuálne sietové pripojenie: WiFi SSID príspovedného bodu, heslo a URL adresa na MQTT broker. Neodporúča sa ponechávať heslo zahrnuté v tomto zobrazení, ale napomáha sa tým testovaniu.

```
{  
    "ssid": "SSID AP",  
    "pass": "Heslo",  
    "url": "mqtt://broker.url"  
}
```

### B.3 Datasetsy z premávky

Datasetsy z autobusov a električiek boli zaznamenané v dátumoch 1.11. a 3.11.2021 so vzorkovacou frekvenciou 500 Hz a rozlíšením  $\pm 2$  g so zariadením opísaním v hlavnej časti. Vozidlá boli súčasťou bežnej výpravy mestskej hromadnej dopravy prepravcu Dopravný podnik Bratislava, a.s. Mierne diskrepancie na úrovni vzoriek a nezmyselné presahy v nahrávaní boli odstránené. Asfaltové povrchy cest boli pomerne nové a suché.

#### L3 \_ StnVinohrady \_ Riazanska.csv

- **Linka:** 3
- **Trvanie:** 120033 vzoriek (240,07 s)
- **Zastávky:** Stn. Vinohrady (v pokoji pred semafórom), Nám. Biely Kríž, Mladá Garda, Riazanská
- **Vozidlo:** električka Škoda 30 T
- **Umiestnenie:** pravá časť nápravy, vyvýšené sedenie v štvorke

#### L3 \_ Pionierska \_ RacianskeMyto.csv

- **Linka:** 3
- **Trvanie:** 70437 vzoriek (140,87 s)
- **Zastávky:** Pionierska, Ursínyho, Račianske mýto
- **Vozidlo:** električka Škoda 30 T
- **Umiestnenie:** pravá časť nápravy, vyvýšené sedenie v štvorke

**L4\_7954\_ZaluhyKrizovatka\_KutikyObratisko.csv**

- **Linka:** 4
- **Trvanie:** 97362 vzoriek (194,72 s)
- **Zastávky:** Záluhy (semafór, pokoj), Horné Krčace, Dolné Krčace, Kútiky obratisko za druhou výhybkou (pohyb)
- **Vozidlo:** električka ČKD Tatra T6A5 #7954
- **Umiestnenie:** zadný vozeň v okolí nad ľavou časťou prednej nápravy

**L9\_Postova\_KralovskeUdolie.csv**

- **Linka:** 9
- **Trvanie:** 149150 vzoriek (298,3 s)
- **Zastávky:** Poštová, Kapucínska, (Tunel), Kráľovské údolie
- **Vozidlo:** električka Škoda 29 T
- **Umiestnenie:** ľavá časť v zníženom sedení medzi harmonikou a vyvýšenou zadnou plošinou

**L9\_Lanfranconi\_Riviera.csv**

- **Linka:** 9
- **Trvanie:** 79010 vzoriek (158,02 s)
- **Zastávky:** Lanfranconi, Botanická záhrada, Riviéra
- **Vozidlo:** električka Škoda 29 T
- **Umiestnenie:** ľavá časť v zníženom sedení medzi harmonikou a vyvýšenou zadnou plošinou

**L20\_3014\_Zaluhy\_Drobneho.csv**

- **Linka:** 20
- **Trvanie:** 113001 vzoriek (226 s)
- **Zastávky:** Záluhy (jazda, pred križovatkou smer Dúbravka od Lamača), Záluhy (zastávka), Švantnerova, Alexyho, Drobného, Podvornice (na polceste, jazda, križovatka)
- **Vozidlo:** elektrobus SOR NS 12 Electric #3014
- **Umiestnenie:** ľavá zadná náprava, predposledné zadné sedenie, pod pravým sedadlom v dvojke

**L35\_1915\_Most\_Kutiky\_neutral.csv**

- **Linka:** 35
- **Trvanie:** 8315 vzoriek (16,63 s)
- **Zastávky:** žiadne - zastavený na moste Kútiky kvôli prekážke na ceste
- **Vozidlo:** midibus Solaris Urbino 8,6 #1915
- **Umiestnenie:** pravá zadná náprava, predposledné zadné sedenie proti smeru jazdy

**L35\_1915\_Borska\_Zaluhy.csv**

- **Linka:** 35
- **Trvanie:** 109502 vzoriek (219 s)
- **Zastávky:** koniec Púpavovej ulice (jazda), Borská (zastávka), Záluhy (jazda, hneď za pravotočivou zákrutou križovatky na smer Lamač)
- **Vozidlo:** midibus Solaris Urbino 8,6 #1915
- **Umiestnenie:** pravá zadná náprava, predposledné zadné sedenie proti smeru jazdy

**L83\_4940\_PriKrizi\_Alexyho.csv**

- **Linka:** 83
- **Trvanie:** 208089 vzoriek (416,18 s)
- **Zastávky:** Pri Kríži (tesne po rozbehnutí), Homolova, Štepná, Žatevná, Pekníková, Drobného, Alexyho (križovatka, zastavenie na semafóre)
- **Vozidlo:** klíbový autobus Mercedes-Benz O 530 GL CapaCity #4940
- **Umiestnenie:** nad motorom vpravo vzadu, posledné zadné priečne sedadlo pred plošinou na batožinu

**L83\_4940\_Alexyho\_Svantnerova.csv**

- **Linka:** 83
- **Trvanie:** 44182 vzoriek (88,36 s)
- **Zastávky:** Alexyho (zastávka), Švantnerova (zastávka, na zvažujúcim kopci)
- **Vozidlo:** klíbový autobus Mercedes-Benz O 530 GL CapaCity #4940
- **Umiestnenie:** nad motorom vpravo vzadu, posledné zadné priečne sedadlo pred plošinou na batožinu

# Príloha C: Používateľská príručka

## C.1 Inštalačný manuál

Vývojovou platformou bola Linux distribúcia *Manjaro 21.2.6*. KDE Plasma s jadrom verzie 5.10. Uvedenie senzorovej jednotky do prevádzky popisuje ďalej uvedený postup:

1. Najprv je potrebné nainštalovať systémové závislosti pre ESP-IDF SDK a MQTT broker:

```
$ sudo pacman -S --needed mosquitto gcc git make flex bison  
gperf python-pip cmake ninja ccache dfu-util libusb
```

2. Následne stiahneme knižnice v požadovaných verziach. Pokiaľ použijeme knižnice už pribalené na digitálnom médiu v priečinku firmvér, **môžeme vyniechať tento krok** a nie je už nutné pridávať knižnice do CMake zostavenia a vykonať úpravu DCT v esp-dsp. ESP-IDF používame verzie 4.4.1, ESP-DSP je verzie 1.2, a MPack je verzie 1.1:

```
$ git clone -b v4.4.1 --recursive https://github.com/  
espressif/esp-idf.git  
$ git clone -b v1.2.0 https://github.com/espressif/esp-dsp.  
git  
$ wget https://github.com/ludocode/mpack/releases/download/  
v1.1/mpack-amalgamation-1.1.tar.gz && tar -xvf mpack-  
amalgamation-1.1.tar.gz
```

3. Nainštalujeme nástroje používané ESP-IDF na kompliaciu programu, spustením príkazu z priečinku `firmware/esp-idf`:

```
$ ./install.sh esp32
```

4. Na prehliadanie Jupyter notebookov prieskumných analýz doinštalujeme balíčky pre *Python 3.10*, odporúčane vo virtuálnom prostredí, z priečinku

measurements. Nástroj príkazového riadku na vzdialenú konfiguráciu ESP32 má závislosti v osobitom súbore:

```
$ pip install -r requirements.txt  
$ pip install -r test-requirements.txt
```

5. MQTT broker *Eclipse Mosquitto* v2.0.14 potrebuje na povolenie pripájania klientov v lokálnej sieti aplikovať konfiguráciu zo súboru mosquitto.conf a následne musí byť služba reštartovaná. Príkazy spúšťame z koreňového adresára.

```
$ mv firmware/vibration-analyzer/config/mosquitto.conf  
      /etc/mosquitto/mosquitto.conf  
$ sudo systemctl restart mosquitto  
$ sudo systemctl status mosquitto
```

6. Záznam na SD kartu umožníme umiestnením súboru nastavení OpenLog config.txt z priečinku firmware/vibration-analyzer/conf na pamäťovú kartu.

## C.2 Nahratie firmvéru

1. Prinesením IoT zariadenia do novej senzorovej siete v neznámom stave sa očakáva určenie prihlásovacích údajov WiFi prístupového bodu a URL lokácie pre MQTT broker v štruktúre login v súbore main.c a priečinku firmware/vibration-analyzer/main/src. Odporúča sa, aby URL adresa pozostávala z doménového mena, ale prípustná je aj IP adresa servera brokera (Zistená napr. cez ip addr). Požadovanú úvodnú systémovú konfiguráciu je možné ovplyvniť tam isto, v štruktúre conf. Príklad sieťového pripojenia:

```
static Provisioning login = {  
    .wifi_ssid="ap",  
    .wifi_pass="12345",  
    .mqtt_url="mqtt://192.168.1.10:1883"  
};
```

2. V súbore main.c musí na nahratie pravidel konfigurácie, ako atribútu do asociatívnej časti nevolatilnej pamäte, dočasne **odkomentovaná** direktíva FACTORY\_RESET.

3. ESP32 pripojíme cez Micro USB na počítač. Pomocný napaľovač nastavení umiestníme do flash pamäte mikrokontroléra spustením nasledovných príkazov z priečinku `firmware/vibration-analyzer`. Sériový port určíme podľa aktuálne priradeného názvu.

```
$ . ./.esp-idf/export.sh
$ idf.py build
$ idf.py -p /dev/ttyUSB0 flash
```

4. Samotný firmvér nahráme na ESP32 po opäťovnom **zakomentovaní**

`FACTORY_RESET:`

```
$ idf.py build
$ idf.py -p /dev/ttyUSB0 flash
```

5. Schopnosť prihlásiť sa na WiFi prístupový bod overíme:

```
idf.py -p /dev/ttyUSB0 monitor
```

Posledný riadok výpisu pre úspešné prihlásenie do siete má vyzeráť podobne tomuto:

```
W (858) wifi:<ba-add>idx:0 (ifx:0, 98:da:c4:79:6a:fa), tid
:0, ssn:3, winSize:64
```

6. Senzorovú jednotku môžeme odpojiť od počítača a pripojiť na samostatný zdroj napájania. Pred zapnutím ESP32 a po prihlásení klienta na MQTT topic „syslog“ by sme mali obdržať reťazec „imu started“. Následne sa začne zber a vyhodnocovanie zrýchlenia zo snímača.

```
$ mosquitto_sub -h localhost -t imu/1/syslog
imu started
```

## C.3 Konfiguračný klient

V priečinku `firmware/vibration-analyzer/tests` sa nachádza nástroj na vzdialenú interaktívnu konfiguráciu senzorovej jednotky:

```
$ python config_tool.py
```

Nástroj podporuje uvedenú sadu príkazov. Predvolené dodatočne dopytovaných hodnôt sú v hranatých zátvorkách a na ich potvrdenie stačí stlačiť riadkovač.

- **end** - Ukončenie programu konfiguračného klienta
- **connect** - Pripojenie k serveru so službou MQTT broker. Ponúkané možnosti:

- „Device ID [1]” - ID senzorovej jednotky na filtrovanie komunikácie. Naraz je umožnené spravovanie len jedného kozového uzla
  - „Broker IP [192.168.1.103]” - IP adresa alebo doménové meno MQTT brokera
  - „Broker Port [1883]” - Číslo TCP portu MQTT brokera
- **disconnect** - Odpojenie sa od nastavovania konkrétneho zariadenia
  - **^C** - (Ctrl+C) Zrušenie priebehu aktuálneho príkazu
  - **set** - Zmena konfigurácie zadaná vo formáte JSON na výzvu: `config>`
  - **config** - Dopyt konfigurácie prítomnej na senzorovej jednotke
  - **login** - Zmena sietových nastavení zadaných vo formáte JSON na výzvu: `login>`
  - **credentials** - Dopyt nastavených údajov o sietovom pripojení
  - **topic** - Odoberanie MQTT témy po stanovení časový interval ohľadom zariadením produkovaných údajov. Téma sa zadáva bez prefixu na výzvu: `topic>`

## C.4 Replikácia experimentov

Výsledky experimentov na implementovanom firmvéri sa riadili podľa presne stanoveného poradia uplatňovania pravidiel konfigurácie za stabilných podmienok. Najdôležitejšie prepínače kompilátora, vložené do `sdkconfig` nástrojom `idf.py menuconfig`, ktoré sa použili plošne sú:

- Optimalizácia na veľkosť: `-Os`  
 $(CONFIG_COMPILER_OPTIMIZATION_SIZE=y)$
- Taktovacia frekvencia: 160 MHz  
 $(CONFIG_ESP32_DEFAULT_CPU_FREQ_MHZ=160)$
- Interval plánovania operačného systému: 100 Hz  
 $(CONFIG_FREERTOS_HZ=100)$

Po odkomentovaní príslušnej direktívy na podmienenú kompliaciu v `main.c` podľa typu experimentu bol takto pozmenený firmvér nahratý na zariadenie. Predvolená konfigurácia bola cez `config_tool.py` obnovená na začiatku pokusu vždy rovnaká, zachytená v technickej dokumentácii k MQTT témam. Následne sa príkazom `set` konfigurácia pozmeňovala a odčítavali sa výpisu na konzolu aj do súboru. Realizovalo

sa zakaždým 10 meraní, s ktorých bol vypočítaný aritmetický priemer.

```
$ idf.py monitor | tee experiment.txt
```

Prieskumná analýza datasetov a vyhodnotenie úspešnosti hľadania špičiek na syntetickom signále sa nachádza v Jupyter notebookoch, ktoré sa otvárajú z priečinku `measurements` príkazom:

```
$ jupyter notebook
```

## Experiment: Pamäťová efektivita

Direktíva podmienenej kompliacie: MEMORY\_MEASUREMENT

Konfigurácie:

```
1 {"sensor": {"n": 8}, "tsmooth": {"n": 8}, "fsmooth": {"n": 8}}
2 {"sensor": {"n": 16}, "tsmooth": {"n": 16}, "fsmooth": {"n":
16}}
3 {"sensor": {"n": 32}, "tsmooth": {"n": 32}, "fsmooth": {"n":
32}}
4 {"sensor": {"n": 64}, "tsmooth": {"n": 64}, "fsmooth": {"n":
64}}
5 {"sensor": {"n": 128}, "tsmooth": {"n": 128}, "fsmooth": {"n":
128}}
6 {"sensor": {"n": 256}, "tsmooth": {"n": 256}, "fsmooth": {"n":
256}}
7 {"sensor": {"n": 512}, "tsmooth": {"n": 512}, "fsmooth": {"n":
512}}
8 {"sensor": {"n": 1024}, "tsmooth": {"n": 1024}, "fsmooth": {"n":
1024}}
```

Filtrovanie relevantných riadkov:

```
$ sed -rn 's/^.*\($MEM.*\)$/\2/p' experiment.txt
> memory_usage.csv
```

## Experiment: Časová efektivita algoritmov

Direktíva podmienenej kompliacie: EXECUTION\_TIME\_ALGORITHMS.

Zmena konfigurácie oproti predvolenej:

```
1 {"sensor": {"axis": [false, false, true]}}}
```

Odskúšané algoritmy frekvenčnej transformácie a detekcie špičiek:

```
1 {"sensor": {"n": 32, "fs": 16}, "transform": {"f": "dft"}, "
peak": {"strategy": "neighbours"}}
```

```
2 {"sensor": {"n": 32, "fs": 16}, "transform": {"f": "dft"}, "  
    peak": {"strategy": "zero_crossing"}}  
3 {"sensor": {"n": 32, "fs": 16}, "transform": {"f": "dft"}, "  
    peak": {"strategy": "hill_walker"}}  
4 {"sensor": {"n": 32, "fs": 16}, "transform": {"f": "dct"}, "  
    peak": {"strategy": "neighbours"}}
```

Pre každú predošlú stratégiu sa odmerali postupne rozličné dĺžky okien:

```
1 {"sensor": {"n": 64, "fs": 32}}  
2 {"sensor": {"n": 128, "fs": 64}}  
3 {"sensor": {"n": 256, "fs": 128}}  
4 {"sensor": {"n": 512, "fs": 256}}  
5 {"sensor": {"n": 1024, "fs": 512}}
```

## Experiment: Časová efektivita vyhľadzovacieho filtra

Direktíva podmienenej kompliacie: EXECUTION\_TIME\_SMOOTHING.

Zmena konfigurácie oproti predvolenej:

```
1 {"sensor": {"fs": 256, "n": 512, "axis": [false, false, true]  
        ], "tsmooth": {"on": true}}
```

Konfigurácie:

```
1 {"tsmooth": {"n": 4, "repeat": 1}}  
2 {"tsmooth": {"n": 16, "repeat": 1}}  
3 {"tsmooth": {"n": 64, "repeat": 1}}  
4 {"tsmooth": {"n": 4, "repeat": 4}}  
5 {"tsmooth": {"n": 16, "repeat": 4}}  
6 {"tsmooth": {"n": 64, "repeat": 4}}  
7 {"tsmooth": {"n": 4, "repeat": 8}}  
8 {"tsmooth": {"n": 16, "repeat": 8}}  
9 {"tsmooth": {"n": 64, "repeat": 8}}
```

## Experiment: Časová efektivita spracovania údajov

Direktíva podmienenej kompliacie: EXECUTION\_TIME\_PIPELINE.

Zmena konfigurácie oproti predvolenej:

```
{  
  "sensor": {"fs": 16, "n": 32, "axis": [false, false, true]},  
  "stats": {"min": false, "max": false, "rms": false,  
            "avg": false, "var": false, "std": false,  
            "skew": false, "kurt": false, "med": false,  
            "mad": false, "corr": false},  
  "transform": {"log": true}  
}
```

Nastavenie na začiatku série meraní. **Tabuľka A:**

```

1 {"sensor": {"axis": [false, false, true]}, "transform": {"f": "dft"}}
2 {"sensor": {"axis": [false, false, true]}, "transform": {"f": "dct"}}

```

**Tabuľka B:**

```

1 {"sensor": {"axis": [true, true, true]}, "transform": {"f": "dft"}}
2 {"sensor": {"axis": [true, true, true]}, "transform": {"f": "dct"}}

```

**Tabuľka C:**

```

1 {"sensor": {"axis": [false, false, true]}, "stats": {"min": true, "max": true, "rms": true, "avg": true, "var": true, "std": true, "skew": true, "kurt": true, "med": true, "mad": true, "corr": true}, "logger": {"samples": "f", "stats": true}, "transform": {"f": "dft"}}

```

**Tabuľka D:**

```

1 {"sensor": {"axis": [true, true, true]}, "transform": {"f": "dft"}}

```

Po každom riadku z predošlých úvodných nastavení nasleduje 9 obmien, kedy sa postupne upravuje veľkosť okna a algoritmus detekcie špičiek.

```

1 {"sensor": {"fs": 16, "n": 32}, "peak": {"strategy": "neighbours"}}
2 {"peak": {"strategy": "zero_crossing"}}
3 {"peak": {"strategy": "hill_walker"}}
4 {"sensor": {"fs": 128, "n": 256}, "peak": {"strategy": "neighbours"}}
5 {"peak": {"strategy": "zero_crossing"}}
6 {"peak": {"strategy": "hill_walker"}}
7 {"sensor": {"fs": 512, "n": 1024}, "peak": {"strategy": "neighbours"}}
8 {"peak": {"strategy": "zero_crossing"}}
9 {"peak": {"strategy": "hill_walker"}}

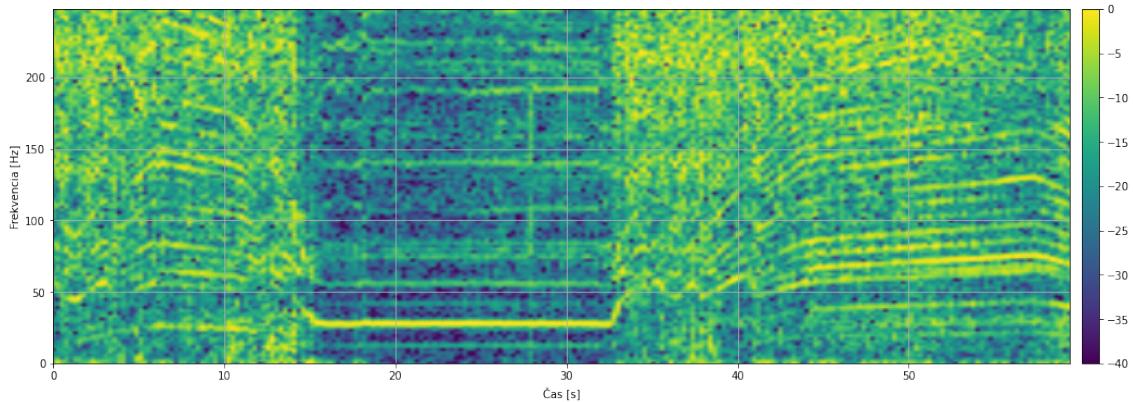
```

Výpisy zo separátnych experimentov A, B, C, D sú prehľadané na výskyt riadkov s časmi a odtiaľ manuálne upravené do finálnej podoby

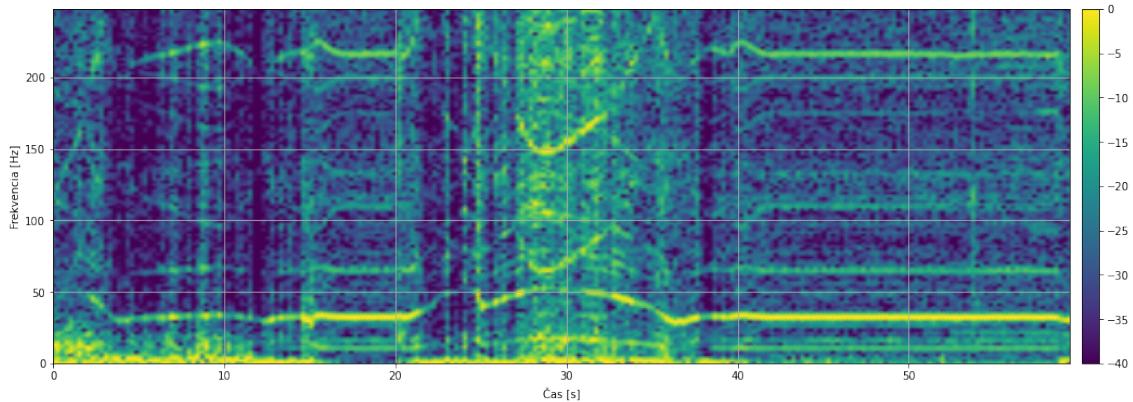
```
sed -rn '/^(.*) (main:.*$ /p' _pipeline_time-X.txt >_pipeline_time-X_filter.csv
```



## Príloha D: Spektrogramy

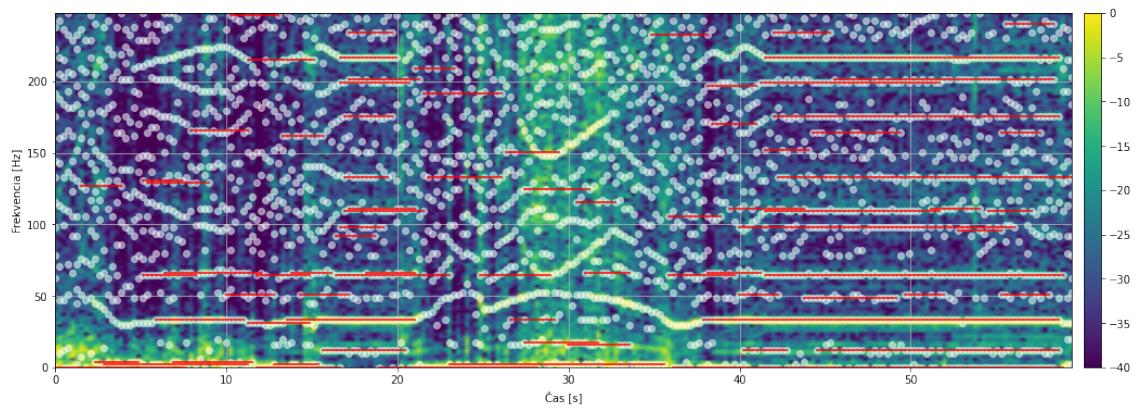


Obr. D.1: Spektrogram záznamu z vozidla *L83\_4940\_Alexyho\_Svantnerova.csv*

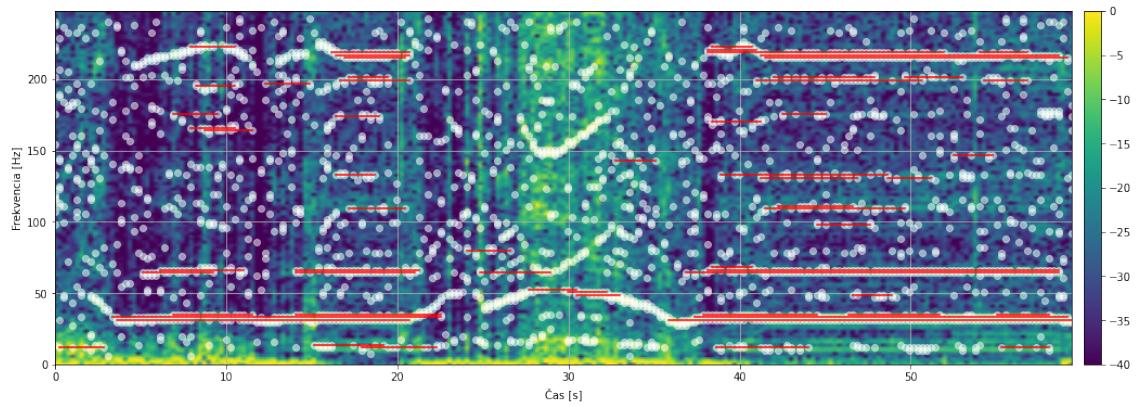


Obr. D.2: Spektrogram záznamu z vozidla *L35\_1915\_Borska\_Zaluhy.csv*

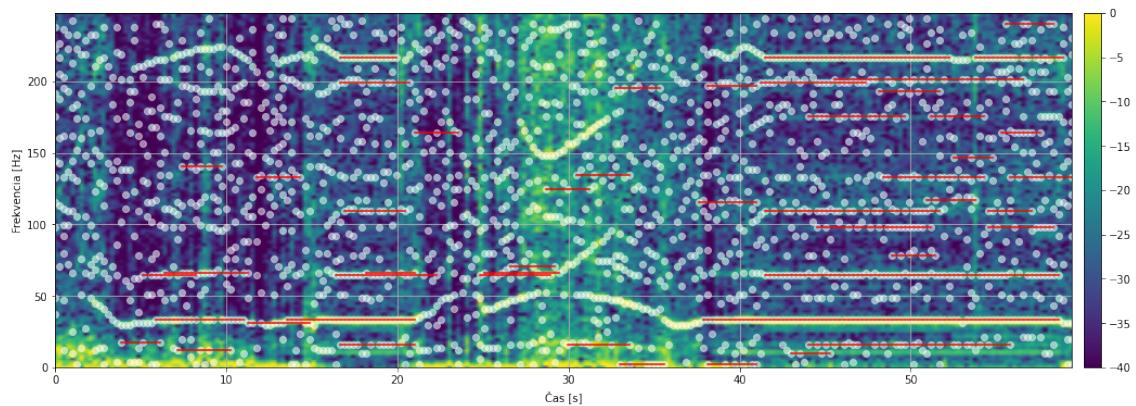
Detekcia udalostí pri vzorkovacej frekvencii 500 Hz, dĺžkou FFT 256 bodov,  $t_{min} = 10$  a  $t_{\Delta} = 4$  na datasete *L35\_1915\_Borska\_Zaluhy.csv*:



Obr. D.3: Algoritmus č.1:  $k = 5$ ,  $\epsilon = 0$ ,  $h_{rel} = 8$



Obr. D.4: Algoritmus č.2:  $k = 3$ ,  $s = 7$



Obr. D.5: Algoritmus č.3:  $t = 8$ ,  $h = 1$ ,  $p = 5$ ,  $i = 0$

# Príloha E: Obsah digitálneho média

Evidenčné číslo práce v informačnom systéme: FIIT-5212-102927

Obsah digitálnej časti práce (archív ZIP):

Názov odovzdaného archívu: BP\_MiroslavHajek.zip

- > **firmware** - Zdrojový kód firmvéru senzorovej jednotky
  - > **esp-dsp** - ESP DSP Library - knižnica na optimalizáciu spracovania signálov na ESP32.
  - > **esp-idf** - Espressif IoT Development Framework - SDK pre hardvér ESP32.
  - > **mpack** - MPack knižnica enkódera a dekódera MessagePack serializačného formátu.
  - > **vibration-analyzer** - Samotná implementácia aplikačnej logiky.
    - > **conf** - Konfiguračné súbory pre OpenLog (*config.txt*), MQTT broker (*mosquitto.conf*) a na zostavenie dokumentácie cez Doxygen (*doxygen.conf*).
    - > **docs** - Doxygen dokumentácia. Úvodná stránka je *index.html*.
  - > **main**
    - > **include** - Hlavičkové súbory *.h* aplikácie.
    - > **src** - Zdrojové súbory *.c* aplikácie.
  - > **tests** - Jednotkové testy na kontrolu najdôležitejšej funkcionality a validujúce konzistenciu medzi Python a C implementáciou algoritmov: *test\_events.c* (test prúdového algoritmu nájdenia zmeny frekvencií), *test\_peaks.c* (test klasifikátorov špičiek), *test\_config.py* (test vzdialenej konfigurácie zariadenia). Nástroj na interaktívny vzdialený prístup k IoT jednotke *config\_tool.py*.

- > **measurements** - Analýza dátových sád a generovanie syntetických signálov v notebookoch *.ipynb*.
  - > **datasets** - Vibračné záznamy z vozidiel verejnej dopravy.
  - > **experiments** - Pôvodné monitorovacie výpisy z experimentov slúžiace ako poklad na overenie riešenia.
    - > **accuracies** - Úspešnosti klasifikácie na syntetickom spektrálnom profile. Súbor *results.txt* vychádza z analýzy v *VibrationProcessingAlgorithms.ipynb*, do tabuľkovej podoby sa dostal v *results-table.csv*.
    - > **execution-algorithms** - Časy vykonávania jednotlivých algoritmov.
    - > **execution-pipeline** - Časy vykonávania obmien spracovania údajov.
    - > **memory-usage** - Spotreba flash pamäte.
    - > **network** - Odchytaná sieťová komunikácie z MQTT broker v *.pcap* súboroch.
  - > **signals** - Užitočné funkcie ku prieskumnej analýze, vizualizácii a tvorbe modelov v Jupyter notebookoch.