

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

FIIT-5212-102927

SPRACOVANIE DÁT GENEROVANÝCH
SENZOROVOU IOT SIEŤOU

BAKALÁRSKA PRÁCA

2022

MIROSLAV HÁJEK

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-102927

Miroslav Hájek

Spracovanie dát generovaných senzorovou IoT sietou

Bakalárská práca

Študijný program: Informatika

Študijný odbor: Informatika

Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky

Vedúci práce: Ing. Marcel Baláž, PhD.

Pedagogický vedúci: Ing. Jakub Findura

Máj 2022



ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Miroslav Hájek**

ID študenta: 102927

Študijný program: informatika

Študijný odbor: informatika

Vedúci práce: Ing. Marcel Baláž, PhD.

Vedúci pracoviska: Ing. Katarína Jelemenská, PhD.

Pedagogický vedúci práce: Ing. Jakub Findura

Názov práce: **Spracovanie dát generovaných senzorovou IoT sieťou**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Senzorové IoT siete sa stali bežnou súčasťou rôznych priemyselných procesov. Ich primárnu úlohou je zbieranie rôznorodých dát z prostredia, ich ukladanie a vyhodnocovanie v reálnom čase. Analyzovanie a vyhodnocovanie dát pri nepredržitom monitorovaní už len z malého množstva senzorov predstavuje veľkú vyzvu. Senzory produkujú veľké množstvo dát a anomália nemusia byť na prvý pohľad detegovateľné. Cieľom projektu je analyzovať dátá zachytené senzorovou sieťou. Analyzovať algoritmy na ich ukladanie a spracovanie. Analyzujte jednotlivé úrovne senzorovej siete a identifikujte miesta, kde by sa dali dátá čiastočne spracovať. Na základe analýzy navrhnite spôsob ukladania a spracovania dát, prípadne optimalizáciu toku dát pre existujúcu senzorovú sieť. Vaše riešenie implementujte a otestujte jeho funkčnosť.

Rozsah práce: 40

Termín odovzdania bakalárskej práce: 16. 05. 2022

Dátum schválenia zadania bakalárskej práce: 23. 11. 2021

Zadanie bakalárskej práce schválil: doc. Ing. Valentino Vranić, PhD. – garant študijného programu

Čestné prehlásenie

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 16.5.2022

.....

Miroslav Hájek

Pod'akovanie

Chcel by som sa pod'akovať vedúcemu práce Ing. Marcelovi Balážovi, PhD. za ústretovosť, mnohé cenné pripomienky a podnety k vylepšeniam, usmernenia pri vytýčení zamerania a povzbudenie ku tvorivému preskúmaniu problematiky.

Za poskytnutie senzorovej jednotky a za postrehy ku formálnej stránke vďačím Ing. Lukášovi Doubravskému.

Tiež d'akujem svojmu kolegovi Ing. Michalovi Juranyimu, ktorý ma za roky spolupráce mnohému priučil o vývoji softvéru. Veľmi si cením morálnu podporu popri štúdiu od rodičov a od najbližšieho okruhu spolužiakov – kamarátov.

Anotácia

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Študijný program: Informatika

Autor: Miroslav Hájek

Bakalárska práca: Spracovanie dát generovaných senzorovou IoT sieťou

Vedúci bakalárskej práce: Ing. Marcel Baláž, PhD.

Pedagogický vedúci: Ing. Jakub Findura

Máj 2022

V bakalárskej práci sa zameriavame na spôsoby spracovania signálov z vibrácií pri preprave, zachytených senzorom akcelerácie mikromechanickej konštrukcie. Zámerom je extrakcia črt záujmu z prúdu vzoriek do udalostí, čím sa redukuje objem posielaných dát v senzorovej sieti.

V časovej doméne nahliadame na sledovaný dej ako stochastický proces opísateľný metrikami deskriptívnej štatistiky. Špecifické okolnosti umožňujú odvodiť zo zrýchlenia ostatné kinematické veličiny numerickou integráciou. Vibrácie obsahujú frekvenčné zložky separovateľné Fourierovou a kosínusovou transformáciou realizovaných algoritmom FFT vo viacerých obmenách za aplikovania oknových funkcií. Významné okamihy sa prejavujú prudkostou zmeny alebo výraznou úrovňou lokálneho extrému, ktoré sú odlíšené binárnou klasifikáciou. Úlohu pri úprave zdrojového signálu zohrávajú tiež filtre s konečnou impulznou odozvou.

Prihliadame na obmedzenia vyplývajúce z nasadenia riešenia na zariadenia Internetu vecí v kontexte Edge computing architektúry. Konfigurovateľný postup spracovania harmonických zložiek trojosovej akcelerácie sa uplatní vo firmvéri na bezdrôtovo komunikujúcom mikrokontroléri vzorom Publish-Subscribe. Úspešnosť detekcie frekvencií sa overuje na základe syntetických sínusových časových radov a vlastných záznamov z vozidiel verejnej dopravy.

Annotation

Slovak University of Technology Bratislava

Faculty of Informatics and Information Technologies

Degree course: Informatics

Author: Miroslav Hájek

Bachelor's Thesis: Data Processing for Sensor IoT Network

Supervisor: Dr. Marcel Baláž

Departmental advisor: Jakub Findura

2022, May

In the bachelor's thesis we focus on signal processing of vibrations during transport, captured using microelectromechanical acceleration sensor. The intention is to extract features of interest from the stream of samples into events, thereby reducing the amount of data sent in the sensor network.

In the time domain, we look at the observed phenomenon as a stochastic process expressed by various descriptive statistics. Specific circumstances make it possible to derive other kinematic quantities from the acceleration by numerical integration. The vibrations contain frequency components separable by Fourier and cosine transform with FFT algorithm in several variants alongside application of window functions. Significant moments are manifested therein by the change intensity or the significant level of the local extremes, which are distinguished by binary classification. Finite impulse response filters also play a role in adjusting the source signal.

We take into account the limitations of deployment on the Internet of Things devices in the context of the Edge computing architecture. The configurable harmonic component processing procedure for three-axis acceleration is applied in the firmware of a microcontroller with wireless communication capability by the Publish-Subscribe pattern. The success of frequency detection is verified on the basis of synthetic sinusoidal time series and recordings from public transport vehicles.

Obsah

1	Úvod	1
2	Analýza	3
2.1	Monitorovanie vibrácií a šoku	3
2.1.1	Meranie fyzikálnej veličiny akcelerácie	3
2.1.2	MEMS kapacitný akcelerometer	4
2.1.3	Analógovo-digitálny prevodník	6
2.1.4	Vlastnosti bežných akcelerometrov	7
2.1.5	Odvodzovanie rýchlosťi a dráhy zo zrýchlenia	8
2.1.6	Numerická kvadratúra	9
2.2	Metódy analýzy signálu v časovej doméne	10
2.2.1	Prúdové algoritmy	11
2.2.2	Posuvné a rozširujúce sa okná	11
2.2.3	Číselné charakteristiky štatistického rozdelenia	12
2.3	Algoritmy na rozpoznávanie špičiek	15
2.3.1	Detekcia špičiek prahovou úrovňou	15
2.3.2	Význačnosť vrchola spomedzi susedov	16
2.3.3	Algoritmus prechodu nulou do záporu	17
2.3.4	Algoritmus horského turista	18
2.3.5	Metriky pre binárny klasifikátor	19
2.4	Frekvenčná a časovo-frekvenčná analýza signálu	21
2.4.1	Diskrétna Fourierová a kosínusová transformácia	21
2.4.2	Algoritmus FFT	23
2.4.3	Oknové funkcie	25
2.4.4	Filtre s konečnou impulznou odozvou	26
2.5	Senzorová siet'	28

3 Návrh riešenia	31
3.1 Špecifikácia požiadaviek	31
3.2 Hardvér senzorovej jednotky	32
3.3 Architektúra systému	33
3.4 Fázy dátovej pipeline	35
3.4.1 Nastaviteľné vlastnosti	36
3.4.2 Preskúmané obmeny pipeline	38
3.4.3 Prúdový algoritmus detekcie zmien frekvencií	39
3.5 Datasetsy	41
3.5.1 Syntéza časovo-premenného spektrálneho profilu	41
3.5.2 Zber vibrácií z premávky	42
4 Implementácia	43
4.1 Senzorová sieť	43
4.2 Komunikácia medzi úlohami	45
4.3 Udalosti vo frekvenčnom spektre	47
4.4 Systémová konfigurácia	48
5 Overenie riešenia	49
5.1 Pamäťová efektivita	49
5.2 Časová efektivita	51
5.3 Úspešnosť detekcie špičiek	53
6 Zhodnotenie	59
Literatúra	61
A Plán práce	
B Technická dokumentácia	
C Používateľská príručka	
D Spektrogramy	
E Obsah digitálneho média	

Zoznam obrázkov

2.1	Model oscilujúceho systému s pružinou a tlmičom	3
2.2	Mikroštruktúra 3DOF MEMS kapacitného akcelerometra [3]	5
2.3	Digitalizácia signálu v analógovo-digitálnom prevodníku [5]	7
2.4	Porovnanie pravidiel numerickej integrácie	10
2.5	Dopad šiknosti a špicatosti na histogram distribúcie	14
2.6	Topografia priebehu signálu	18
2.7	Radix-2 FFT na štyroch bodoch [28]	24
2.8	Motýlikové diagramy algoritmu FFT	24
2.9	Tvar oknových funkcií s dĺžkou $N = 31$	26
2.10	Bloková schéma FIR filtra rádu k	28
2.11	Prvky architektúry Edge computing [37]	30
3.1	Schéma zapojenia hardvéru	33
3.2	Komponenty navrhovaného systému	34
3.3	Sekvenčný diagram vzorkovania signálu a spolupráce úloh	35
3.4	Postup spracovania zaznamenávaných vibrácií	37
3.5	Príjem nových pravidiel a dopytovanie systémovej konfigurácie	38
3.6	Parametre algoritmu na detekciu udalostí	39
3.7	Základný tón v syntetickom signále	41
4.1	Univerzálny plošný spoj v krabičke osadený modulmi	44
5.1	Profilovanie dynamickej pamäte z haldy v DRAM	50
5.2	Spektrogramy detegovaných špičiek a udalostí pri $f_s = 476$ Hz a $N = 256$	56
5.3	Prierez spektrogramu okna 256 vzoriek s vrcholmi označenými algoritmom č.1 v 20. sekunde záznamu <i>L83_4940_Alexyho_Svantnerova.csv</i>	56

5.4 Detekcia udalostí v datasete *L83_4940_Alexyho_Svantnerova.csv* s
 $f_s = 500$ Hz, trvaním 60 s, dĺžkou okna 256, pri $t_{min} = 10$ a $t_\Delta = 4$. . . 57

Zoznam rovníc

2.0	Fyzikálny model oscilujúceho systému s pružinou a tlmičom	4
2.1	Newtonov zákon sily	4
2.2	Magnitúda vektora akcelerácie	4
2.3	Nyquist-Shannonova veta o vzorkovaní	6
2.4	Konverzia merania na akceleráciu podľa rozlíšenia A/D prevodníka	7
2.5	Prevod A/D prevodníkom u akcelerometra s rozsahom v mg/LSB	7
2.6	Kinematické rovnice pre polohu, rýchlosť, zrýchlenie a rychlosť	8
2.7	Rýchlosť a poloha cez integrál akcelerácie	9
2.9	Pravidlá numerickej kvadratúry: obdĺžníkové, lichobežníkové, Simpsonovo	9
2.12	Amplitúda špička-špička	12
2.13	Efektívna amplitúda RMS	12
2.14	Výberový priemer alebo amplitúda jednosmernej zložky	12
2.15	Výberové miery rozptylenosti: rozptyl, smerodajná odchýlka, MAD, IQR	13
2.19	Welfordov algoritmus na výpočet výberového rozptylu	14
2.22	Závislosť dvoch veličín cez kovarianciu a koreláciu	14
2.24	Lokálne extrémy funkcie	15
2.26	Detekcia špičiek prahovou úrovňou	16
2.27	Význačnosť vrchola spomedzi susedov	16
2.28	Metriky klasifikátora: senzitivita a špecifickosť	20
2.30	Metriky klasifikátora: správnosť, presnosť, chybovosť	20
2.33	Rozlíšenie vo frekvenčnej domény podľa vzorkovania	21
2.34	Diskrétna Fourierová transformácia	21
2.36	Exponenciálny faktor pre DFT v goniometrickom tvare	22
2.37	Magnitúdové spektrum absolútne a relatívne v decibeloch	22
2.39	Kosínusové transformácie: DCT-II, DCT-III, DCT-IV, MDCT	22

2.39	Oknové funkcie: obdlžník, Bartlett, Hann, Hamming, Blackman	25
2.44	Výpočet FIR filtra cez konvolúciu	27
2.45	Koeficienty FIR filtra pre dolnú, hornú a pásmovú prieplust'	27
3.0	Rovnica periodického kmitania	41

Zoznam skratiek a pojmov

f_s Vzorkovacia frekvencia

g Tiažové zrýchlenie ($1\ g = 9,80665\ m/s^2$)

A/D Analógový na digitálny

Chybovosť False positive rate (pravdepodobnosť výskytu falošného poplachu)

DCT Discrete Cosine Transform (diskrétna kosínusová transformácia)

DFT Discrete Fourier Transform (diskrétna Fourierová transformácia)

DOF Degree of Freedom (stupeň voľnosti mechanického systému)

FET Field-effect transistor (tranzistor riadený poľom)

FFT Fast Fourier Transform (algoritmus rýchlej Fourierovej transformácie)

I²C Inter-Integrated Circuit (dvojvodičová synchrónna sériová zberznica)

IoT Internet of Things (internet vecí)

ISM páisma Voľné páisma pre rádiové vysielanie v priemyselnom, vedeckom a zdrotovníckom sektore

LSB Least significant bit (najmenej významový bit)

MCU Microcontroller unit (mikrokontrolér na jednom integrovanom obvode)

MEMS Micro-Electro-Mechanical Systems (mikromechanický systém)

MQTT téma MQTT topic (logické zoskupenie publikovaných správ so spoločným zameraním)

MQTT Message Queuing Telemetry Transport (aplikačný protokol na prenos telemetrických údajov cez fronty správ)

MTU Maximum transmission unit (najdlhší poslaný paket bez fragmentácie)

ODR Output data rate (výstupný dátový tok)

Pipeline Súbor prvkov spracovania údajov zapojených do série, kde výstup jedného prvku je vstupom ďalšieho prvku

Precíznosť Precision (tesnosť zhody medzi výsledkami meraní navzájom)

Presnosť Accuracy (blízkosť nameraných hodnôt ku pravdivej hodnote)

SDK Software development kit (nástroje na vývoj softvéru pre špecifickú platformu)

Senzitivita Sensitivity, True Positive Rate (TPR), Recall. (pravdepodobnosť pozitívneho testu byť skutočne pozitívnym)

SPI Serial Peripheral Interface (synchrónne sériové periférne rozhranie)

TCP Transmission Control Protocol (transportný sieťový protokol riadenia prenosu)

UART Universal asynchronous receiver-transmitter (zbernica asynchronného sériového prenosu)

Špecifickosť Specificity, True Negative Rate (TNR). (pravdepodobnosť negatívneho testu byť skutočne negatívnym)

1 Úvod

Inteligentné senzorové systémy zariadení internetu vecí zaznamenávajú obrovskú kvantitu údajov z prostredia, kde pôsobia. Prúdy vzoriek meraných veličín majú samy osebe nízku informačnú hodnotu. Zbytočne zaťažujú prenosové pásmo komunikačných kanálov a kapacitu úložísk. Monitorovanie širokého rozsahu kladie požiadavky na nízke výrobné náklady senzorových jednotiek a dlhodobú výdrž pri napájaní z batérií za minimálnej údržby. Existuje preto potreba získané dátu spracovať do istej miery už v blízkosti ich zdroja, aby došlo k efektívному využitiu dostupných prostriedkov.

Význam a dôležitosť sledovania vibrácií spočíva v ich výskytte u každého mechanického zariadenia pohybom jednotlivých súčiastok a trením v ložiskách. Ich nadmerná prítomnosť býva spôsobená opotrebením dielov stroja alebo dôsledkom technických defektov. Ďalšou oblastou hojnej prítomnosti vibrácií je preprava osôb a tovaru. Tam sú zapríčinené nerovnosťami povrchu vozovky alebo koľaje v bode styku s kolesami, či aparátom ovplyvňujúcim pohyb vozidla. Menovite ich vyvoláva točivý moment spaľovacieho alebo elektrického motora a činnosť brzdového systému.

Detekciou nežiaducich vibrácií v preprave sa dokáže zabezpečiť bezpečnosť pasažierov včasnomu výmenou súčiastky, ktorá by ovplyvnila prevádzkyschopnosť v kritických momentoch. Ich odhalením predchádzame nenávratnému poškodeniu krehkých materiálov, znehodnoteniu reaktívnych substancií, či ich aktivácií v prípade výbušní a pyrotechniky. Vibrácie sú súčasťou nebezpečných prírodných úkazov a správna identifikácia má za následok varovania na evakuáciu obyvateľstva v oblasti postihnutej zemetrasením, či erupciou sopky, vedúcimi k ohrozeniu zdravia osôb a poškodenia majetku.

Vibračný signál je merateľný v digitálnej podobe snímačom pohybového zrýchlenia mikromechanickej konštrukcie, o čom pojednávame v kapitole 2. Na postupnosť

pozorovaní sa nazerá ako vlnový priebeh, ktorý sa sprehľadňuje agregačnými, korelačnými a testovacími štatistikami na odhalenie náhlych zmien. Významne úrovne sa odlišujú od nevýznamných algoritmami na detekciu špičiek. Metódami transformácie do frekvenčnej oblasti sa objavujú periodicky prítomné zložky. Modely spracovania majú byť nasadené do adekvátnej vrstvy senzorovej siete. V kapitole 3 popíšeme hardvér, pre ktorý navrhнемe firmvér uskutočňujúci sústavu krokov na extrakciu udalostí z vektora zrýchlenia a predstavíme dátové sady na validáciu funkčnosti. Ďalej v kapitole 4 je prezentovaná implementácia najdôležitejších štruktúr a komponentov. Nakoniec riešenie overíme v kapitole 5 a dosiahnuté výsledky okomentujeme v kapitole 6.

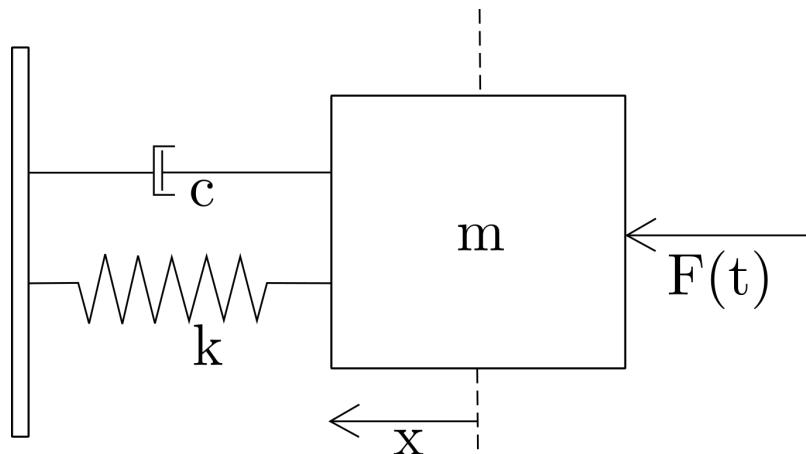
2 Analýza

2.1 Monitorovanie vibrácií a šoku

Vibrácie sú periodickým kmitaním hmoty okolo rovnovážnej polohy vznikajúce excitáciou látky, ktorej je dodaná potenciálna energia, a zo zákona zachovania energie je následne premieňaná na kinetickú energiu. V realite dochádza pôsobením trenia k útlmu voľného oscilačného pohybu s časom a pohybová energia sa uvoľňuje v podobe tepelnej alebo akustickej emisie do okolitého prostredia. Častejšie ako presné harmonické kmity sú pozorované náhodné vibrácie, ktorých vývoj nevieme dopredu predvídať. Naproti tomu šok, alebo aj prechodový jav, je náhle uvoľnenie kinetickej energie krátkeho trvania oproti prirodzenej oscilácii systému.

2.1.1 Meranie fyzikálnej veličiny akcelerácie

Pohyb mechanického systému vystaveného vonkajším silám sa nazýva odozva, ktorej správanie opisuje zjednodušený model s jedným stupňom voľnosti (1DOF) kmitajúceho telesa s pružinou a tlmičom na obr. 2.1 [1].



Obr. 2.1: Model oscilujúceho systému s pružinou a tlmičom

Pri pôsobení vonkajšej sily F na hmotu upevnenú na pružine vznikajú nútene vibrácie, ktoré ju vychyľujú z rovnovážnej polohy. Uvedená sila je charakterizovaná druhým Newtonovým zákonom v tvare $F = ma$, kde m je hmotnosť telesa, a predstavuje zrýchlenie. V protismere pôsobí sila vyvolaná pružinou $F_s = -kx$ a tlmiacim členom $F_d = -cv$, kde k je tuhost pružiny ovplyvnená jej konštrukciou, c je tlmiaci koeficient, x je vychýlenie z rovnovážneho stavu, v je rýchlosť vychýlenia.

Obmedzením telesa viazaním na pevnú podložku dochádza pri zanedbaní deformácie k takmer zaručenému návratu do rovnovážnej polohy, a to nám umožňuje merat intenzitu vibrácií cez zrýchlenie ťažidla. Výslednú silu v jednom smere získame sčítaním síl podieľajúcich sa na dynamike telesa:

$$F(t) = ma - cv - kx \quad (2.1)$$

U trojosového akcelerometra, kedy sa snímajú tri priestorové súradnice časovo-premennej akcelerácie dostávame nasledujúcu rovnicu vo vektorovom tvare:

$$\vec{a}(t) = \frac{\vec{F}(t)}{m} \quad (2.2)$$

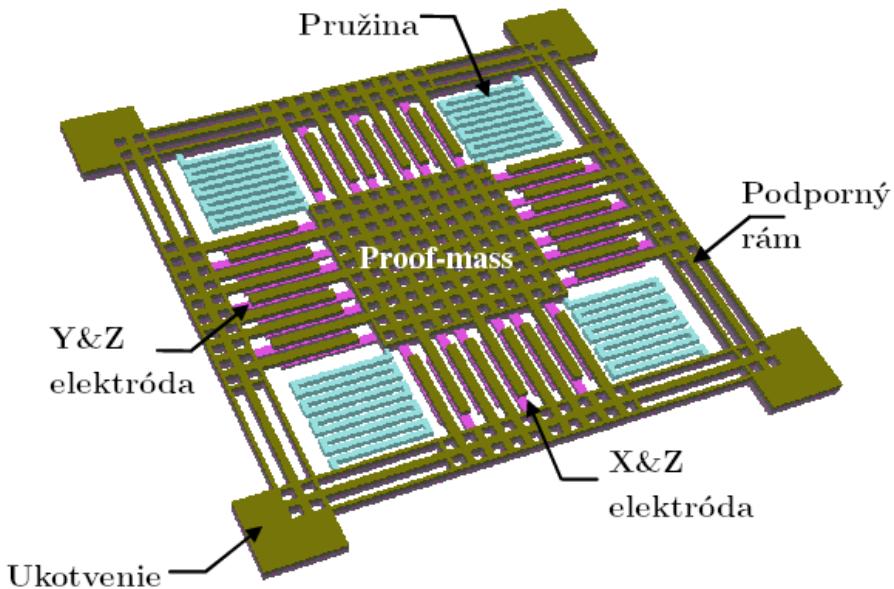
Magnitúda 3D vektora akcelerácie $\vec{a} = (a_x, a_y, a_z)$ je L_2 normou vektora:

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (2.3)$$

2.1.2 MEMS kapacitný akcelerometer

Bežné inerciálne senzory na meranie zrýchlenia priamočiareho, ale aj rotačného pohybu (gyroskop), sa vyrábajú technológiou *MEMS* (mikromechanický systém), kedy je celé zariadenie vrátane všetkých mechanických súčastí umiestnené na kremík procesom mikrovýroby vo viacerých vrstvách. Sila spôsobujúca zrýchlenie je potom meraná vychýlením vstavanej odpruženej hmoty vzhľadom na pevné elektródy, ktoré môžu byť usporiadane jednostranne alebo ako diferenčný párs [2].

Pri diferenčnom páre spôsobí pohyb doštičky ťažidla medzi elektródami zmenu kapacít, ich rozdielom je možné zistiť aplikovanú silu a cez uvedený vzťah sily zrýchlenie. Na zvýšenie celkovej kapacity sa používa viacero párov elektród zapojených paralelne. Pred prevodom na číslicový signál musí napäťová úroveň zo senzora



Obr. 2.2: Mikroštruktúra 3DOF MEMS kapacitného akcelerometra [3]

prejst' úpravou zahŕňajúcou nábojovocitlivý predzosilňovač, osovú demoduláciu a antialiasingové filtrovanie.

Viacosové akcelerometre vyžadujú viaceré opísané štruktúry orientované kolmo na seba, podľa obr. 2.2, s ohľadom na počet vyžadovaných stupňov voľnosti, pričom v skutočných senzoroch vždy existuje aspoň minimálna závislosť medzi osami, rádovo najviac v jednotkách percent. Teplota ovplyvňuje citlosť MEMS akcelerometrov len nepatrne v stotinách percenta na stupeň Celzia.

Akcelerometre sa odlišujú v niekoľkých dôležitých vlastnostiach, ktoré zvyknú byť nastaviteľné vo výrobcom stanovenom rozsahu prípustných hodnôt s príslušnými toleranciami [4].

Citlivosť stanovuje najmenšiu rozlíšiteľnú zmenu v odčítanom napäti ku zmene externého pohybu respektíve zrýchlenia. Uvádzajú sa v jednotkách mV/g (milivolt na tiažové zrýchlenie) pri analógovom výstupe, alebo mg/LSB (mili-g na najmenej významový bit) pri senzoroch so vstavaným analógovo-digitálnym prevodníkom. Jednotka mg/LSB vyjadruje o koľko sa zmení zrýchlenie, keď zvýšime alebo ponížime binárne číslo na výstupe o jedna. Niekoľko sa namiesto citlivosti uvádzajú mierka pre presnosť ako prevrátená hodnota citlivosti v LSB/g.

Dynamický rozsah sa uvádzajú v tiažovom zrýchlení g . Hovorí o najmenšej a najväčšej rozlíšiteľnej hodnote zrýchlenia, nad úrovňou ktorej už dochádza ku skresleniu signálu orezaním špičiek. Nevyhnutnými drobnými nepresnosťami výroby mik-

romechaniky vzniká *zero-g napätie* popisujúce odchýlku skutočného od ideálneho výstupu, keď na sústavu nepôsobí žiadne zrýchlenie. Za ideálnych okolností bez pohybu na vodorovnom povrchu namerajú osi \vec{x} a \vec{y} zrýchlenie 0 g , zatiaľčo na \vec{z} pôsobí 1 g . Očakávaním je nulová hodnota výstupného napäťa.

Šírka pásma senzora v Hz predurčuje rozsah frekvencie vibrácií, ktoré je možné zachytiť. Podmienená je zvolenou početnosťou čítania akcelerácie za sekundu. Stavuje sa nastaviteľným parameterom *ODR* (Output Data Rate), výstupným dátovým tokom, pričom šírka pásma je spravidla polovicou ODR. *Frekvenčná odozva* senzora potom hovorí o koľko sa v rámci tolerancie odlišuje skutočná citlivosť od referenčnej pre zodpovedajúcu frekvenciu vibrácií.

Na meranie zrýchlenia má vplyv šum zapríčinený Brownovým pohybom a nedokonalosťou skutočných materiálov v štruktúre akcelerometra. Intenzita šumu rastie inverznou odmocninou so šírkou pásma, čiže s častejším meraním získavame menšiu presnosť. Hardvér akcelerometra umožňuje vzorkovať amplitúdy až nad stanovený prah generovaním prerušenia, čím sa efektívne odstránia nevýznamné fluktuácie, za predpokladu dostatočného odstupu signálu od šumu.

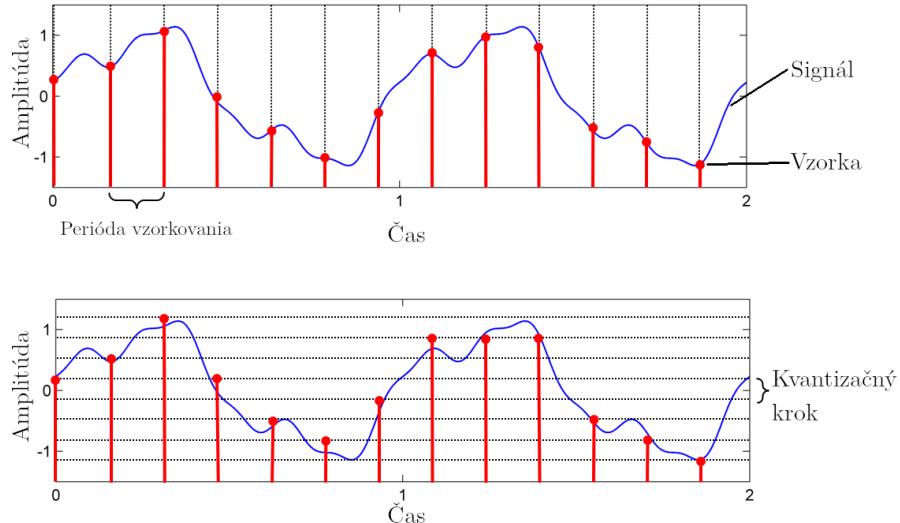
2.1.3 Analógovo-digitálny prevodník

Spojité napäťovú úroveň transformuje analógovo-digitálny (A/D) prevodník pre spracovanie digitálnym systémom do množiny diskrétnych hodnôt. Vstupný signál najprv prechádza fázou vzorkovania, kedy sa vzorky zaznamenávajú v pravidelných intervaloch. Počet vzoriek odčítaných za sekundu je vyjadrený vzorkovacou frekvenciou f_s v Hz. Časový rozdiel medzi vzorkami, nazývaný períoda vzorkovania, je prevrátenou hodnotou vzorkovacej frekvencie $T_s = 1/f_s$. Pre presnú rekonštrukciu pásmovo obmedzeného signálu v hraniciach $[-f_{max}; f_{max}]$ je nevyhnuté podľa *Nyquist-Shannonovej vety* o vzorkovaní, aby vzorkovacia frekvencia bola najmenej dvojnásobkom maximálnej frekvencie snímaného signálu:

$$f_s \geq 2 \cdot f_{max} \quad (2.4)$$

V procese kvantovania je každej vzorke je následne priradená diskrétna hodnota s konečným počtom bitov, ktorá je najbližšia možná ku skutočnej hladine analógového

vstupu. Dochádza pritom k istému zaokrúhľovaniu z dôvodu nepresnosti vyjadrenia spojitej domény amplitúd diskrétnym číslom, čo zapríčiňuje kvantizačný šum. Ten predstavuje najviac polovicu z maximálnej rozlíšiteľnej zmeny signálu.



Obr. 2.3: Digitalizácia signálu v analógovo-digitálnom prevodníku [5]

Pri n bitoch je k dispozícii 2^n rozličných čísel. Kódovaním v dvojkovom doplnku na zachytenie záporných hodnôt sa uvažuje s intervalom $[-2^{\frac{n}{2}}; 2^{\frac{n}{2}} - 1]$. Digitálna hodnota v dvojkovom doplnku získaná konverziou \hat{x} je prepočítaná na štandardné fyzikálne jednotky pre zrýchlenie, na m/s^2 . R prestavuje nastavený dynamický rozsah v jednotkách g a n je počet bitov A/D prevodníka:

$$a = \hat{x} \cdot ((R \cdot g) / 2^{n/2}) \quad (2.5)$$

Presnejší prevod dosiahneme zužitkovaním deklarovanej citlivosti senzora pri danom dynamickom rozsahu S_R udávaného v mg/LSB:

$$a = \hat{x} \cdot (S_R \cdot g) / 1000 \quad (2.6)$$

2.1.4 Vlastnosti bežných akcelerometrov

Na ilustráciu uvádzame parametre zvolených najrozšírenejších typov akcelerometrov. Akcelerometer LSM9DS1 [6] umožňuje cez zbernicu SPI alebo I²C zvoliť zo štyroch dynamických rozsahov, pričom každé rozpätie sa vyznačuje svojou citlivosťou. Zvolením menšieho dynamického rozsahu zvýšime citlivosť. LSM9DS1 funguje

pri rozsahoch ± 2 g, ± 4 g a ± 8 g a ± 16 g, postupne s citlivosťami 0.061 mg/LSB, 0.122 mg/LSB, 0.244 mg/LSB, 0.732 mg/LSB. Výstupný dátový tok je možné nastaviť na 10Hz, 50Hz, 119 Hz, 238 Hz, 476 Hz a najvyššie na 952 Hz. Navzorkované hodnoty sú ukladané do 16-bitového výstupného registra v dvojkovom doplnku.

Nízkoenergetický 3DOF MEMS akcelerometer ADXL362 [7] so spotrebou $2 \mu\text{A}$ pri 100 Hz disponuje rozsahmi ± 2 g, ± 4 g a ± 8 g s citlivosťami 1, 2 a 4 mg/LSB. Dostupné vzorkovacie frekvencie 12-bitového A/D prevodníka sú 12,5 - 400 Hz v 8 krokoch vždy po násobkoch predošlého kroku. Pre rýchlejšie čítanie pri nižšom rozlíšení dokáže senzor zakódovať dátá do 8-bitového registra.

Vyrábajú sa tiež akcelerometre s väčšími dynamickými rozsahmi a nízkym šumom. Ide napríklad o ADXL356 a ADXL357 [8] so škálami ± 10 g, ± 20 g a ± 40 g s citlivosťou 0,019 - 0,078 mg/LSB a rozlíšením A/D prevodníka 20 bitov pri ODR 4 - 4000 Hz. ADXL357 ponúka priamo analógové výstupy s citlivosťou 20 - 80 mV/g pri napájaní 3,3 volta.

2.1.5 Odvodzovanie rýchlosťi a dráhy zo zrýchlenia

Meranie akcelerácie umožňuje zároveň nepriamo získať ďalšie údaje o celkovom pohybe v priestore ako aj spôsobenom vibráciami. Zrýchlenie \vec{a} je definované ako časová zmena rýchlosťi \vec{v} , zatiaľ čo rýchlosť je časovou zmenou polohy \vec{r} . Na pozorovanie prechodových javov alebo na vyjadrenie miery plynulosti pohybu slúži ryv \vec{j} , ktorý je časovou zmenou akcelerácie. Pokiaľ nie sú známe počiatočné podmienky v okamihu začiatku snímania akcelerácie, budú hodnoty veličín relatívne vzhľadom na štart záznamu. Kinematika v diskrétnom čase je potom opísaná nasledujúcimi rovnicami, kde Δ je operátor diferencie $\Delta t = t(i) - t(i - 1)$:

$$\vec{v} = \frac{\Delta \vec{r}}{\Delta t}; \quad \vec{a} = \frac{\Delta \vec{v}}{\Delta t}; \quad \vec{j} = \frac{\Delta \vec{a}}{\Delta t} \quad (2.7)$$

Vyjadrenie neznámych premenných vzhľadom na akceleráciu spočíva v prenásobení rovníc členom Δt , čím sa získajú vzťahy pre okamžitú dráhu a okamžitú rýchlosť. Spočítaním čiastkových okamžitých rýchlosťí na intervale dostaneme celkovú rýchlosť a rovnaký úsudok platí pre polohu. V spojitom čase, keď by vzorkovacia períoda bola nekonečne krátka, dochádza naproti tomu k integrovaniu funkcie

akcelerácie. Dostávame, že rýchlosť je integrálom zrýchlenia a poloha je dvojným integrálom zrýchlenia:

$$\vec{v}(t) = \vec{a}_0 + \int \vec{a}(t) dt \quad (2.8)$$

$$\vec{r}(t) = \vec{r}_0 + \vec{v}_0 t + \int \int \vec{a}(t) dt \quad (2.9)$$

2.1.6 Numerická kvadratúra

Približný výpočet určitého integrálu funkcie akcelerácie je založený na geometrickej interpretácii integrálu ako plochy pod krivkou. Vtedy sa jedná o problém numerickej kvadratúry, kde sa pôvodný integrand nahradí interpolačným polynómom [9]. Rád polynómu n implicitne stanoví priebeh funkcie medzi ekvidistantnými vzorkami a má dopad na presnosť aproximácie. Najčastejšie sa používajú konštantný, lineárny alebo kvadratický polynóm, podľa toho rozlišujeme obdlžnikové pravidlo (vzorec 2.10), lichobežníkové pravidlo (vzorec 2.11) a Simpsonovo pravidlo (vzorec 2.12).

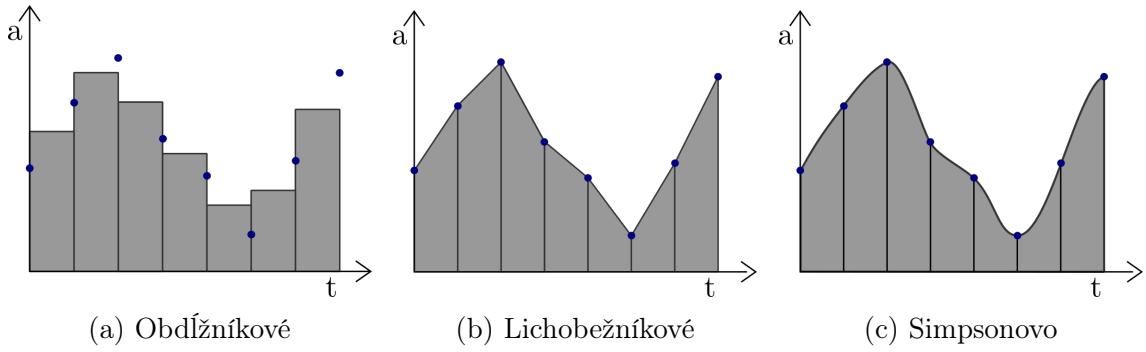
$$v(t_i) = T_s \cdot a \left(\frac{t_i + t_{i-1}}{2} \right) \quad (2.10)$$

$$v(t_i) = \frac{T_s}{2} \cdot [a(t_i) + a(t_{i-1})] \quad (2.11)$$

$$v(t_i) = \frac{T_s}{3} \cdot [a(t_{2i}) + 4a(t_{2i-1}) + a(t_{2i-2})] \quad (2.12)$$

Pri obdlžníkovom pravidle (obr. 2.4a) nepripúšťame zmenu hodnoty zrýchlenia medzi vzorkami. Okamžitú rýchlosť, čiže plochu, odhadneme ako dĺžku intervalu vzorkovania vynásobenú priemerom výšok dvoch následných pozorovaní. Interpoláčný polynóm je konštantná funkcia. Lichobežníkové pravidlo (obr. 2.4b) uvažuje s lineárhou zmenou veličiny medzi meraniami, preto interpoluje priamkou. Simpsonovo pravidlo (obr. 2.4c) sa snaží o ešte tesnejší odhad s využitím kvadratickej funkcie. Každé kvadratúrne pravidlo sa síce vyznačuje presne vyčísliteľnou chybosťou, ale k tomu je nevyhnutné poznáť analytické vyjadrenie vibrácií, čo dáva realistický odhad len pri čisto periodických kmitoch.

Priama integrácia zašumeného signálu zrýchlenia vedie k neskutočnému driftu, ktorý je ešte zvýraznený dvojitou integráciou pri odvodzovaní relatívneho posunutia. Dochádza k zosilneniu nízkych a potlačeniu vyšších frekvencií, čím začne dominovať



Obr. 2.4: Porovnanie pravidiel numerickej integrácie

neexistujúci trend vo výstupných dátach. Očakávané oscilujúce správanie vychýlenia u vibrácií so zväčšujúcim sa počtom sčítancov pri rekurentnom výpočte zaniká. Na zlepšenie stability integrátora sa uplatňuje korekcia cez obálky [10].

Najprv je na vstupnom signále vykonaná zvoleným pravidlom numerická kvadratúra, ktorá môže byť realizovaná na krátkych úsekoch funkcie, aby sa predišlo pretečeniu pri výraznej akumulácii odklonu. Prichádza sa k identifikácii lokálnych extrémov. Ich interpoláciou s kubickou B-spline sa sformuje horná $e_u(t)$, respektívne dolná obálka signálu $e_d(t)$. Obálky sú spriemerované $\bar{e}(t)$ za vzniku odhadu trendovej krivky, ktorá je od už integrovaného signálu odčítaná: $g(t) = f(t) - \bar{e}(t)$. V prípade výpočtu polohy je možné aplikovať uvedený postup kaskádovo, čiže rovnako ako akcelerácia je signál rýchlosťi opäť integrovaný a korigovaný obálkami.

2.2 Metódy analýzy signálu v časovej doméne

Pozorovania veličiny predstavujú udalosti merané sekvenčne v čase, kde sa s každou obdržanou hodnotou x_i viaže unikátna časová značka t_i . Postupnosť jednotlivých čítaní je jednorozmerný časový rad znázorniteľný ako usporiadaná množina dvojíc pečiatky rastúcej v čase a nasnímanej úrovne: $T = \{(t_1, x_1), (t_2, x_2), \dots, (t_n, x_n)\}$. Vzorkovaním v pravidelných intervaloch stačí uvažovať namiesto časových značiek o celočíselných indexoch, ktoré určujú pozíciu prvkov vo vektore pozorovanií: $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$.

2.2.1 Prúdové algoritmy

Pri veľkom objeme prichádzajúcich vzoriek produkovaných senzormi nie je uskutočniteľné ich úplné uchovanie ani spracovanie celkého dátového toku naraz. Častokrát by stratégia neuváženého odkladania viedla k plytvaniu zdrojov a zbytočnému archivovaniu údajov s nízkou informačnou hodnotou. Vhodnejšie je agregovanie toku údajov podľa preddefinovaného zmysluplného kritéria, ktoré zachytáva významné rysy a umožňuje okamžite zodpovedať na vyžadované dopyty.

Priamočiarou realizáciou agregácie je nahliadať na prvky časového radu postupne ako prichádzajú. Prúdové algoritmy pôsobiace v reálnom čase, a teda neschopné vidieť finálny vektor vzoriek vstupu sa vyznačujú vlastnosťou, že vyprodukujú parciálny výsledok platný pre dosiaľ sa vyskytnutú podmnožinu len na základe čiasťkového vstupu.

Za ideálnych okolností by sa mal online algoritmus učiť kontinuálne bez ukladania predošlých bodov a detekcií. V rozhodnutiach algoritmu budú v takom prípade zahrnuté informácie o všetkých predošlých bodoch do terajšieho rozhodnutia. Mal by mať schopnosť adaptovať sa dynamickému prostrediu, v ktorom pôsobí, bez nutnosti manuálnych úprav parametrov modelu. Zároveň je žiaduce minimalizovať falošné pozitíva a negatíva pri detekcii udalostí [11].

2.2.2 Posuvné a rozširujúce sa okná

Časový rad $(x_i)_{i=0}^n$ s dĺžkou n môže byť pre účely výpočtu sumárnych štatistik rozdelený oknovou funkciou $\mathcal{W}_{l,d}$ na podpostupnosti nazývané okná.

Posuvné okná („rolling window“) majú spravidla konštatnú dĺžku l menšiu ako celkovú veľkosť radu a sú aplikované s krokom odstupu d pozorovaní. Rad pozorovaní pozostáva z $(n - (l - 1)) / d$ okien [12]. Prirodzene sa posuvné okná objavujú pri manipulácii s vyrovnavacou pamäťou, ktoré sa využívajú pri blokovom prenose z adaptéra senzora do hlavnej pamäte. Vtedy sa veľkosť bloku sa rovná posunu $l = d$.

Rozširujúce sa okná („expanding window“) nachádzajú uplatnenie v menej prípadoch, spravidla sa jedná o inkrementálny odhad globálnej štatistiky, ktorá má zmysel prevažne pri sledovaní stabilného javu. [13]. Okno začína na stanovenej minimálnej veľkosti a s pribúdajúcim počtom bodov ich zahŕňa, čím sa zväčšuje.

2.2.3 Číselné charakteristiky štatistického rozdelenia

Náhodné vibrácie vyskytujúce sa pri skutočných materiáloch sú stochastický proces, ktorý tvorí sekvencia časovo indexovaných náhodných premenných. Časový rad predstavuje realizáciu tohto stochastického procesu $\mathbf{Y} = (X_1, X_2, \dots, X_n)^T$, kde X_t je náhodná premenná so svojím rozdelením pravdepodobnosti. Všeobecne sa pri ideálnych stacionárnych otrásach predpokladá, že premenné pochádzajú z unimodálnej Gaussovej distribúcie: $X_t \sim N(\mu, \sigma^2)$ [1].

Sumárna deskripcia nameraného dejia pre extrakciu typických črt konkrétnych pozorovaných situácií sa odvíja od viacerých štatistik $h(X_1, X_2, \dots, X_n)$ zostručňujúcimi opis funkcie hustoty rozdelenia. Na rozmiestnenie hodnôt meraní v priebehu časového úseku sa nazerá z pohľadu polohy, rozptylenosti a tvaru. Rozsah oboru hodnôt je amplitúda špička-špička („peak-to-peak”), ktorá je rozdielom maximálnej a minimálnej úrovne, tiež nazývaná variačné rozpätie [14]:

$$x_{pp} = \max_{t \in \mathcal{W}} \{x_t\} - \min_{t \in \mathcal{W}} \{x_t\} \quad (2.13)$$

Priemernú energiu obsiahnutú v signále predstavuje štvorec efektívnej amplitúdy RMS a určí sa ako kvadratický priemer pozorovaní:

$$x_{rms} = \sqrt{\frac{1}{n} \sum_{t=1}^n x_t^2} \quad (2.14)$$

Mierami polohy rozdelenia pozorovaní sú stredná hodnota, informujúca o centre hodnôt veličiny, a kvantily rozkladajúce usporiadany vektor pozorovaní na určený počet rovnakých skupín. Nevychýleným bodovým odhadom strednej hodnoty je *výberový priemer*, ktorý je zároveň amplitúdou jednosmernej zložky signálu:

$$\bar{x} = \frac{1}{n} \sum_{t=1}^n x_t \quad (2.15)$$

Najvýznamnejšími kvantilmi sú kvartily Q_q vytvárajúce štyri rovnako veľké časti z pôvodnej množiny, konkrétnie dolný kvartil Q_1 oddelí 25% najmenších údajov, medián Q_2 predelí zoradené údaje na polovicu a horný kvartil Q_3 zahrnie 75% nižších hodnôt. Hľadaný kvartil je k -ty najmenší prvok v utriedenom zozname meraní,

pričom podľa želaného kvartílu q a počtu pozorovaní je $k = \lceil n \cdot (1 / q) \rceil$.

Zistenie k -teho najmenšieho prvku s časovou zložitosťou $\mathcal{O}(n \log n)$ umožňuje ľubovoľný lepší triediaci algoritmus napríklad triedenie zlučovaním (merge sort). Algoritmus Quickselect dokáže taký prvok objaviť so zložitosťou $\mathcal{O}(n)$. V každom kroku vyberie náhodný deliaci bod (pivot) a preskupí k sebe hodnoty menšie ako pivot naľavo a väčšie ako pivot napravo. Najmenší prvok následne hľadá v časti, kde zostało viac ako k prvkov. Pokiaľ došlo k deleniu zoznamu, že pivot zaujme presne k -tu pozíciu, prehľadávanie je ukončené a pivot prehlásený za riešenie. Nesprávnym výberom pivota môže v najhoršom prípade dôjsť až k zložitosti $\mathcal{O}(n^2)$, čomu sa predchádza výberom pivota stratégou mediánov.

Sústredovanie realizácie veličiny, respektíve jej rozptylenosť okolo strednej hodnoty vieme opísť viacerými štatistikami ako sú výberový rozptyl (2.16), smerodajná odchýlka, priemerná absolútна odchýlka (2.17), mediánová absolútna odchýlka (2.18) a medzikvartilové rozpätie (2.19) [14]. Priemerná absolútna odchýlka je upraviteľná o mieru centrálnej tendencie, ktorou okrem priemeru môže byť aj medián alebo modus. Vyvarovanie sa príliš extrémnym a vychýlením hodnotám docielime zapojením práve mediánu do štatistik absolútnej odchýlky, rovnako tak to dosiahneme medzikvartilovým rozpätím obmedzením sa na 50% centrálnych dát.

$$s^2 = \frac{1}{n-1} \sum_{t=1}^n (x_t - \bar{x})^2 \quad (2.16)$$

$$d = \frac{1}{n} \sum_{t=1}^n (|x_t - \bar{x}|) \quad (2.17)$$

$$\text{MAD} = \text{med}(|x_t - \text{med}(\mathbf{x})|) \quad (2.18)$$

$$IQR = Q_3 - Q_1 \quad (2.19)$$

Numericky stabilné bežiace štatistiky priemeru a smerodajnej odchýlky sa udržiavajú cez rekurentné rovnice *Welfordovho algoritmu* [15]. M_1 je aktuálna priemerná hodnota údajov v toku a S_1 je počítadlo pre rozptyl, z ktorého je v ktoromkoľvek

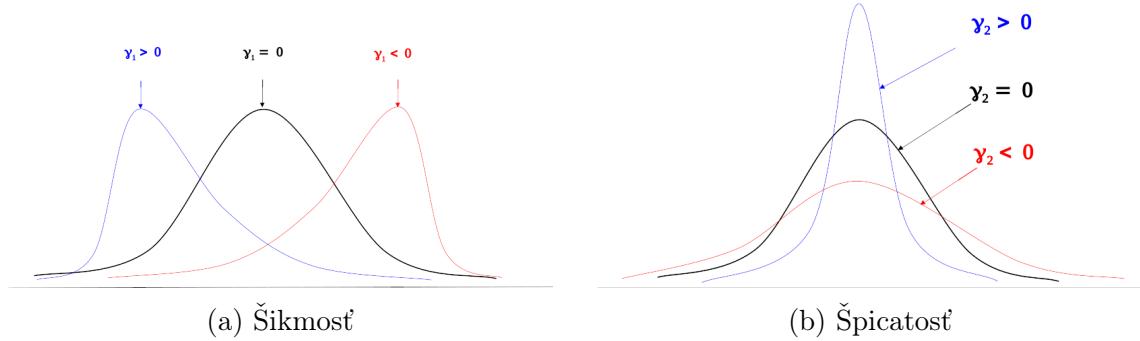
okamihu získateľná smerodajná odchýlka súboru σ :

$$M_1 = x_1; \quad M_k = M_{k-1} + \frac{(x_n - M_{k-1})}{k} \quad (2.20)$$

$$S_1 = 0; \quad S_k = S_{k-1} + (x_k + M_{k-1})(x_k + M_k) \quad (2.21)$$

$$\sigma = \sqrt{S_n/(n-1)} \quad (2.22)$$

Tvar distribúcie náhodnej premennej opisujú centrálné momenty šikmosť a špicatosť. Šikmosť udáva skosenie rozdelenia, pričom platí že záporná šikmosť značí dlhší ľavý chvost, zatiaľ čo u kladnej je to naopak (obr. 2.5a). Špicatosť (obr. 2.5b) porovnáva rozdelenie pozorovaní so strmosťou krivky normálneho rozdelenia. Kladná špicatosť signalizuje strmejšiu a záporná sploštenejšiu distribúciu.



Obr. 2.5: Dopad šikmosti a špicatosti na histogram distribúcie

Závislosť dvojíc veličín sa vyjadruje kovariancia $cov(\mathbf{x}, \mathbf{y})$ a korelácia $\rho(\mathbf{x}, \mathbf{y})$. U vektora akcelerácie nás bude napríklad zaujímať vzájomná korelácia medzi osami pohybu: $\rho(\vec{x}, \vec{y})$, $\rho(\vec{x}, \vec{z})$, $\rho(\vec{y}, \vec{z})$ upozorňujúca na diagonálny pohyb alebo podobné budenie v oboch korelovaných smeroch a tým umožňujúce redukciu údajov z dôvodu redundancie. Kovariancia je daná strednou hodnotou súčinu odchýlky od priemeru zodpovedajúcej premennej (2.23). Normovaním kovariancie smerodajnými odchýlkami veličín získame Pearsonov korelačný koeficient (2.24), ktorý je z intervalu $[-1; 1]$. Hodnota koeficientu -1 značí nepriamu lineárnu závislosť a $+1$ priamu závislosť.

$$cov(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{t=1}^n (x_t - \bar{x})(y_t - \bar{y}) \quad (2.23)$$

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{cov(\mathbf{x}, \mathbf{y})}{\sigma_x \sigma_y} \quad (2.24)$$

2.3 Algoritmy na rozpoznávanie špičiek

Detekcia udalostí a významných zmien signálového priebehu sa spolieha na hodnovernú identifikáciu špičiek amplitúdy. Dôležitými indikátormi pre celkový opis javu slúži časová pozícia špičky v rámci prúdu, výška prejavujúca sa nadobudnutou úrovňou, šírka obsahujúca údaj o trvaní, a plocha stvárňujúca energiu.

Ekvivalentne sa špičky z matematického hľadiska stotožňujú s lokálnymi extrémami funkcie, čo sú maximá (vrcholy) a minimá (údolia). Podľa definície je lokálne maximum t_0 bodom majúcim vyššiu funkčnú hodnotu ako všetky ostatné body na intervale $t_0 \in I$ (2.25), lokálne minimum má na intervale najmenšiu hodnotu (2.26) [16].

$$f_{t_0} \geq f_t, \forall t \in I \quad (2.25)$$

$$f_{t_0} \leq f_t, \forall t \in I \quad (2.26)$$

Kľúčové pre spoľahlivé určenie extrémov je práve interpretácia intervalu I v algoritnoch, ktoré zastupujú rozličné potreby korektného vyhodnotenia. Jediné minimum a maximum sa dosiahne zvolením celej dĺžky záznamu za interval, čím sa stratia dočasné disturbancie. Na druhej stane prílišným skrátením intervalu sa skoro všetky vzorky budú javiť ako náhle zmeny.

Skutočné signály sa potýkajú so šumom, ktorý sťaže odlišenie pravej tendencie od krátkodobých výkyvov. Pred samotným procesom hľadania špičiek býva preto aplikovaný vyhľadzovací filter, v prípade potreby aj opakovane na už vyhľadený signál. Najčastejšie sa jedná o filter kĺzavého priemeru, Savitzky–Golay alebo Gaussov filter [17]. Filtrovanie sa realizuje diskrétnou jednorozmernou konvolúciou vstupného signálu a masky filtra, ktorá býva hardvérovo akcelerovaná inštrukciami „vynásob a sčítaj“ (multiply-accumulate) so zvýšením presnosti výsledku.

2.3.1 Detekcia špičiek prahovou úrovňou

Za predpokladu, že priebeh meranej veličiny sa vyznačuje krátkymi impulzmi s viac-menej pravidelnou amplitúdou je priamočiarou metódou na odlišenie špičiek od hladín nízkej aktivity určenie prahu θ , ktorý zaregistruje väčšie hodnoty.

Lokálne extrémy sú potom vzorky signálu splňajúce podmienku:

$$|f_t| \geq \theta \quad (2.27)$$

Určenie takejto hraničnej hladiny prebieha zväčša empiricky alebo na základe heuristik, ktoré so sebou nesú domnenku o vlastnostiach priebehu pozorovaní. Uspokojivými odhadom za určitých okolností môžu byť prahy θ : viac ako priemer s toleranciou, horné $3/4$ celkového nedávneho rozsahu hodnôt, či dokonca viac ako k smerodajných odchýlok. Odlišné nazeranie na prahovú hodnotu spočíva v jej nastavení pre rozpoznanie vzájomnej korelácie signálu a masky zodpovedajúcej tvaru impulzu. Táto úvaha sa opiera o to, že impulz musí byť dostatočne pravidelný, aby bol nezameniteľne odlišiteľný.

2.3.2 Význačnosť vrchola spomedzi susedov

Doplnkom ku rozpoznávaniu špičiek podľa absolútnej prahovej úrovne je porovnanie bodov na obe strany od preskúmaného vrchola, čím zistíme relatívnu významnosť extrému pre najbližšie susedstvo. Aby bola hodnota na danej pozícii t označená za špičku v okolí pozostávajúcom z k príahlých bodov, musí byť v porovnaní so všetkými väčšia. Pre okrajové dátové body f_0 a f_{n-1} dochádza k porovnaniu iba z jednej strany [16]:

$$f_{t-i} < f_t > f_{t+i}, \quad \forall i \in 1, 2, \dots, k \quad (2.28)$$

Algoritmus č.1 „najvyšší spomedzi susedov“ [18] prechádza postupne pozorovania veličiny zo zoznamu Y a ku kandidátnej špičke na indexe i preveruje najbližších k hodnôt na obe strany, ak existujú.

Ked' po preskúmaní zostáva Y_i najväčšou hodnotou spomedzi susedov v rozmedzí $[a; b]$, za tolerancie bodu s vyššou amplitúdou ε v susedstve, a súčasne je relatívna výška vrcholu väčšia než parameter h_{rel} potom je kandidátny bod prehlásený za skutočnú špičku a pridaný do zoznamu *peaks*.

Súčasťou algoritmu je tiež preskočenie hodnôt, ktoré nespĺňajú základný predpoklad pre absolútну amplitúdu h . Časová zložitosť pre rozhodnutie o jednej špičke je lineárna v závislosti od veľkosti posuvného okna uvažovaného susedstva $\mathcal{O}(2k)$.

Algoritmus 1 Hľadanie špičiek najvyšších spomedzi susedov

Vstupy: $y, k, \varepsilon, h_{rel}, h$

Výstup: $peaks$

```

1: Zoznam  $peaks$  pre špičky v  $Y$  nastav na prázdný
2: for all  $i \in [0..length(Y)]$  do
3:    $a \leftarrow \max\{i - k, 0\}$ 
4:    $b \leftarrow \min\{i + k, length(Y)\}$ 
5:    $valley \leftarrow \min_{j \in [a..b]} \{Y_j\}$ 
6:   if  $Y_i \geq h$  and  $Y_i - valley \geq h_{rel}$  then            $\triangleright$  Preskoč nízke amplitúdy
7:     if  $Y_i \geq Y_j + \varepsilon; \forall j \in [a..b] \wedge i \neq j$  then       $\triangleright$  Bod je najvyšší v susedstve
8:       Pridaj kandidátny bod  $Y_i$  do zoznamu špičiek  $peaks$ 
9:     end if
10:   end if
11: end for

```

2.3.3 Algoritmus prechodu nulou do záporu

Pomyselné vrcholy a údolia v zosnímaných hodnotách sú miestom, kde sa mení smer úrovní amplitúdy zo stúpania na klesanie alebo z klesania na stúpanie. Na pomedzí týchto opozitných trendov vzniká stacionárny bod, kde je prvá differencia nulová: $\Delta f_i = 0$. V lokálnom maxime dochádza súčasne k zmene znamienka prvej diferencie z kladného na záporné. Prudkost kopca vyplýva z absolútnej hodnoty diferencie.

Viacnásobné vyhľadenie signálu predom je nesmierne dôležité, pretože algoritmus č.2 „prechodu nulou do záporu“ (Negative Zero-Crossing) je nesmierne citlivý na zákmity a nesprávne by ich považoval za špičky. Zvýšenie odolnosti proti takýmto tendenciám sa dosahuje dlhšou sečnicou spájajúcou bod i s k -tou vzorkou vedľa, ktorá sa použije namiesto diferencie s jednotkovým krokom.

Algoritmus 2 Hľadanie špičiek prechodom prvej derivácie nulou do záporu

Vstupy: $Y, k, slope$

Výstup: $peaks$

```

1: Zoznam  $peaks$  pre špičky v  $Y$  nastav na prázdný
2: for  $i \leftarrow k$  to  $length(Y) - k$  do
3:   if  $Y_{i+k} - Y_i < 0$  and  $Y_i - Y_{i-k} > 0$  then
4:     if  $|Y_{i+k} - Y_i| \geq slope$  and  $|Y_i - Y_{i-k}| \geq slope$  then
5:       Pridaj kandidátny bod  $Y_i$  do zoznamu špičiek  $peaks$ 
6:     end if
7:   end if
8: end for

```

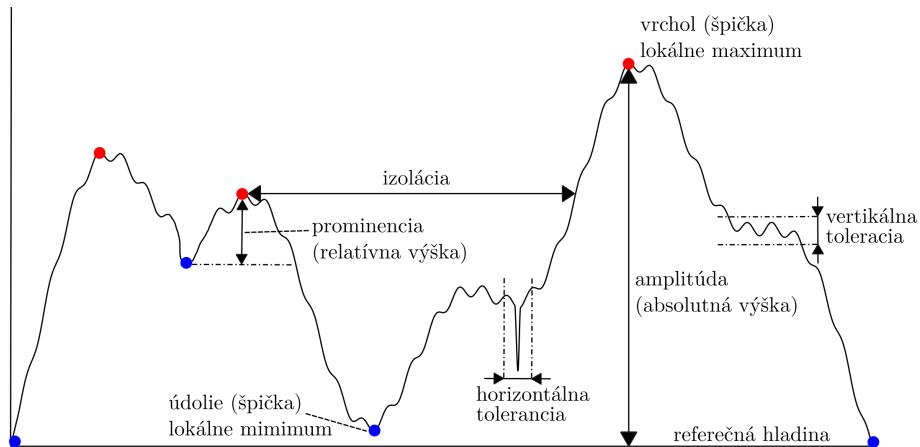
Označenie kandidátneho bodu za špičku v zozname hodnôt y stojí na teda dvoch kritériach. Sečnice na oboch stranách od uvažovaného vrchola sa musia lísiť zna-

mienkom a ich dĺžka má prekračovať prahovú strmosť kopca *slope*. Časová zložitosť pre jednu špičku je $\mathcal{O}(1)$.

2.3.4 Algoritmus horského turistu

Zanesením do grafu pripomína priebeh funkcie kmitajúceho deja členité pohorie. Na problém rozhodovania sa o tom, či danú lokalitu považovať za vrchol možno nahliať z pohľadu chodca cestujúceho po krivke z lineárne interpolovaných vzoriek. V princípe ide myšlienkovou o jednoduchý stavový automat sledujúci aktuálny stav terénu a konajúci rozhodnutia na základe predošej skúsenosti v intencích rozhodovacích pravidiel.

Algoritmus č.3 horského turista na začiatku púte z počiatočných bodov zistí, ktorým z dvoch vertikálnych smerov sa krivka ubera. V prípade, že po druhom kroku dôjde k zmene smeru zapíše sa indikácia možného spádu kopca. Výchylka môže byť v dôsledku neprekročenia prahových úrovni v horizontálnej (*hole*) a vertikálnej (*tolerance*) osi ignorovaná, lebo ani na lesnom chodníku sa nepovažuje každá jama za dolinu alebo vydutie za horu.



Obr. 2.6: Topografia priebehu signálu

Domnely vrchol je označený za lokálne maximum, keď spĺňa parametre pre topografické vlastnosti minimálnej akceptovateľnej prominencie a izolácie (obr. 2.6). Prominencia znamená relatívnu výšku oproti predošej navštívenej doline. Izolácia vycísluje vzdialenosť k najbližšiemu skoršiemu vrcholu. Podobný algoritmus už existuje v literatúre [19], avšak prezentovaný pseudokód je oproti nemu zjednodušený a doplnený o požadované tolerancie.

Algoritmus 3 Hľadanie špičiek metódou horského turista

Vstupy: Y , tolerance, hole, prominence, isolation

Výstup: peaks

```

1: Zoznam peaks pre špičky v  $Y$  nastav na prázdnny
2:  $change \leftarrow 0$ ,  $valley \leftarrow 0$ ,  $candidate \leftarrow false$ ,  $uphill \leftarrow Y_1 - Y_0 \geq 0$ 
3: for  $i \leftarrow 1$  to length( $Y$ ) do
4:    $step \leftarrow Y_i - Y_{i-1}$ 
5:    $slope \leftarrow step \geq 0$ 
6:   if  $\neg candidate$  and  $uphill \neq slope$  then
7:      $candidate \leftarrow \neg candidate$             $\triangleright$  Označenie potenciálneho extrému
8:      $change \leftarrow i - 1$ 
9:   else if  $candidate$  and  $uphill = slope$  then
10:     $candidate \leftarrow \neg candidate$            $\triangleright$  Potenciálny extrém bol zachvením
11:   end if
12:   if ( $candidate$  and  $uphill \neq slope$  and
13:      $|i - change| > hole$  and  $|Y_i - Y_{change}| > tolerance$ ) then
14:      $candidate \leftarrow \neg candidate$ 
15:      $prev\_uphill \leftarrow uphill$             $\triangleright$  Významný lokálny extrém potvrdený
16:      $uphill \leftarrow slope$ 
17:     if  $\neg prev\_uphill$  and  $uphill = true$  then
18:        $valley \leftarrow change$                  $\triangleright$  Nájdené údolie
19:     else if ( $prev\_uphill$  and  $\neg uphill$  and
20:        $|Y_{i-hole} - valley| > prominence$  and
21:        $|Y_{i-hole} - Y_{last(peaks)}| > isolation$ ) then
22:         Pridaj kandidátny bod  $Y_{change}$  do zoznamu špičiek peaks
23:     end if
24:   end if
25: end for
```

2.3.5 Metriky pre binárny klasifikátor

Uviedli sme tri rozdielne rovnocenné prístupy odhalenia špičiek. Rozobrali sme algoritmus porovnávajúci susedov na obe strany, algoritmus využívajúci sklon sečníc vychádzajúc s vlastnosťí prvej derivácie a napokon stavový automat odvolávajúci sa na sekvenčne preskúmanú topografiu krivky grafu. Spoločným rysom zmienených techník je binárne zaradenie vzorky, či sa nachádza alebo nenachádza na aktuálnej pozícii vrchol.

Rozhodnutie môže viest' k správnemu (P) alebo nesprávnemu (N) riešeniu vzhľadom na objektívnu pravdu sprostredkovanú anotovanými dátami. Keď sa kategórizácia zhoduje s realitou dostávame skupiny skutočne pozitívnych TP a skutočne negatívnych TN . V prípade, že sa klasifikátor pomýli, vyjde buď chyba prvého rádu FP , kedy registrujeme neexistujúcu špičku, alebo chyba druhého rádu FN , kedy ju

prehliadneme. Umiestnením počtov charakteru rozhodnutí do tabuľky vzniká matica zámen [20].

Úspešnosť klasifikačných algoritmov, pre ich vzájomné porovnanie, kvantifikujú viaceré metriky. Na odladenie parametrov vplývajúcich na náhľenosť preferovať kladné alebo záporné výsledky sa vzťahuje *prevalencia* výskytu očakávaného javu: $P / (P + N)$ v samotných dátach. Snahou rozhodovania je maximalizovať senzitivitu a špecifickosť výsledkov algoritmu. *Senzitivita* (2.29) udáva koľko bodov, ktoré sú prehlásené za špičky je naozaj špičkami. *Špecifickosť* (2.30) sa zameriava na potvrdenie, aké množstvo pozorovaní nepovažovaných za špičky, nie sú nimi aj skutočne.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \quad (2.29)$$

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} \quad (2.30)$$

Pravdivosť určenia lokálneho extrému sa skladá z precíznosti (2.31) a presnosti (2.32), ktoré je rovnako žiaduce dosahovať čo najbližšie sto percentám, pri nízkej chybovosti (2.33), čiže nízkeho počtu falošných poplachov.

$$PPV = \frac{TP}{TP + FP} \quad (2.31)$$

$$ACC = \frac{TP + TN}{P + N} \quad (2.32)$$

$$FPR = \frac{FP}{FP + TN} \quad (2.33)$$

Štandardným nástrojom na vyjadrenie kvality binárneho klasifikátora je *ROC krivka* zakresľujúca senzitivitu (*TPR*) vo zvislom smere voči vodorovnej chybovosti (*FPR*). ROC vytvoríme postupným posúvaním prahu pre klasifikáciu prostredníctvom parametrov algoritmu. Použiteľný algoritmus sa vyznačuje vypuklou krivkou smerom k ľavému hornému rohu nad diagonálou, ktorá by sprevádzala počinanie náhodného rozhodovania. Dokonalá metóda pri dosahovala stopercentnú senzitivitu za nulovej chyby. Vyjadrením plochy pod ROC krivkou je miera AUC, ktorá umožňuje približné číselné porovnanie rôznych získaných kriviek [21].

2.4 Frekvenčná a časovo-frekvenčná analýza signálu

Cyklicky sa opakujúce deje sú extrahované zo sekvencie vzoriek v časovej doméne transformovaným do domény frekvenčnej. Premenou dochádza k odhadu sumárnej intenzity jednotlivých rozsahov zložiek spektra, pričom rozlíšenie je podmienené vzorkovacou frekvenciou f_s a celkovým počtom meraní N (2.34) [22].

$$\Delta f = \frac{f_s}{N} \quad (2.34)$$

Kompromis potrebný učiniť pri spektrálnej analýze tkvie vo vyvážení dĺžky úseku pre časovú lokalizáciu frekvenčného obrazu, a jeho výslednej detailnosti na strane druhej. Prechod medzi časovou a frekvenčnou doménou postihuje princíp neurčitosti znamenajúci, že pri raste rozlíšenia v čase strácamo rozlíšenie vo frekvenciach a naopak [23]. O požadovanom množstve pozorovaní pre konkrétnu rozlíšiteľnosť spektra taktiež hovorí vzťah (2.34). Rozpätie frekvencií spadajúcich do diskrétneho frekvenčného vedierka k sú počínajúc $k\Delta f$ po $(k+1)\Delta f$.

2.4.1 Diskrétna Fourierová a kosínusová transformácia

Diskrétna Fourierová transformácia (DFT) slúži na učenie harmonického zloženia signálu (2.35) rozkladom na súčet sínusov a kosínusov radu frekvencií. Zobrazuje vektor komplexných čísel y dĺžky N do vektora N frekvenčných komponentov. Na vypočítanie k -teho frekvenčného vedierka sú prvky sekvencie pozorovaní prenásobené zodpovedajúcim exponenciálnym členom tzv. *twiddle factor* (2.36). Inverzná transformácia sa lísi iba opačným znamienkom exponenta a získané hodnoty sa zvyknú normovať podelením N .

$$Y[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk}; \quad k = 0, \dots, N-1 \quad (2.35)$$

$$W_N^{nk} = \exp(-i2\pi nk / N) \quad (2.36)$$

Alternatívne sa exponenciálny faktor vyjadruje v goniometrickom tvare rozkla-

dom na kosínusovú reálnu časť a sínusovú imaginárnu časť:

$$W_N^{nk} = \cos(2\pi nk/N) - i \cdot \sin(2\pi nk/N) \quad (2.37)$$

Spektrálne komponenty opísané vektorom komplexných Fourierových koeficientov majú magnitúdy dané veľkosťami komplexných čísel (2.38). Pre vstupy $x \in \mathbb{R}$ je výstup z DFT zrkadlovo symetrický, čiže druhá polovica výstupu je komplexne združená k prvej $x[m] = x^*[N-m]$. Symetrickosť zapríčinuje nadbytočnosť výsledkov nad pozíciou $N/2$. Rovnako naznačuje reformulácia vety o vzorkovaní, že za vzorkovacej frekvencie f_s sú zapríčinením aliasingu, v signále prítomné frekvencie do maximálne polovice f_s . Energia vo frekvenčnom vedierku je druhou mocninou magnitúdy, ale častejšie sa objavuje reprezentácia relatívneho energetického spektra v decibeloch (2.39) [22].

$$|Y[k]| = \sqrt{\Re\{Y[k]\}^2 + \Im\{Y[k]\}^2} \quad (2.38)$$

$$Y_{dB}[k] = 20 \cdot \log_{10} \left(\frac{|Y[k]|}{\max_{0 \leq k < N/2} \{|Y[k]|}\right) \quad (2.39)$$

Sekvencia pozorovaní vyjadrená cez súčet kosínusoid namiesto exponenciálneho faktora tvorí rodinu diskrétnych kosínusových transformácií (DCT). Vyznačujú sa dobrou dekoreláciu vstupu a energetickou kompresiou, čiže pomerne veľká časť celkovej spektrálnej energie je sústredená v málo koeficientoch [24]. Navyše oproti DFT umožňuje redukciu výpočtovej náročnosti odstránením súčinov v komplexných číslach.

$$W_N^{nk} = \cos \left[\left(n + \frac{1}{2} \right) k \frac{\pi}{N} \right] \quad (\text{DCT-II})$$

$$W_N^{nk} = \cos \left[n \left(k + \frac{1}{2} \right) \frac{\pi}{N} \right] \quad (\text{DCT-III})$$

$$W_N^{nk} = \cos \left[\left(n + \frac{1}{2} \right) \left(k + \frac{1}{2} \right) \frac{\pi}{N} \right] \quad (\text{DCT-IV})$$

$$W_N^{nk} = \cos \left[\left(n + \frac{1}{2} + \frac{N}{2} \right) \left(k + \frac{1}{2} \right) \frac{\pi}{N} \right]; \quad n = 0, \dots, 2N-1 \quad (\text{MDCT})$$

Podľa charakteru obmien kosínusovej bázy rozlišujeme štyri typy DCT, z nich najvýznačnejšie sú transformácie DCT-II, ktorej inverziou je DCT-III, a DCT-IV,

ktorá je inverzná sama sebe [25]. Z DCT-IV vychádza MDCT, ktorá navyše spracováva prekrývajúce sa bloky, tak že druhá polovica vzoriek pochádza z prvej polovice ďalšieho bloku. Dokopy vytvorí z $2N$ vzoriek N koeficientov. MDCT sa hojne využíva pri stratovej kompresii zvuku, pretože sa prelínaním blokom vyvaruje artefaktom na hraniciach blokov [26].

2.4.2 Algoritmus FFT

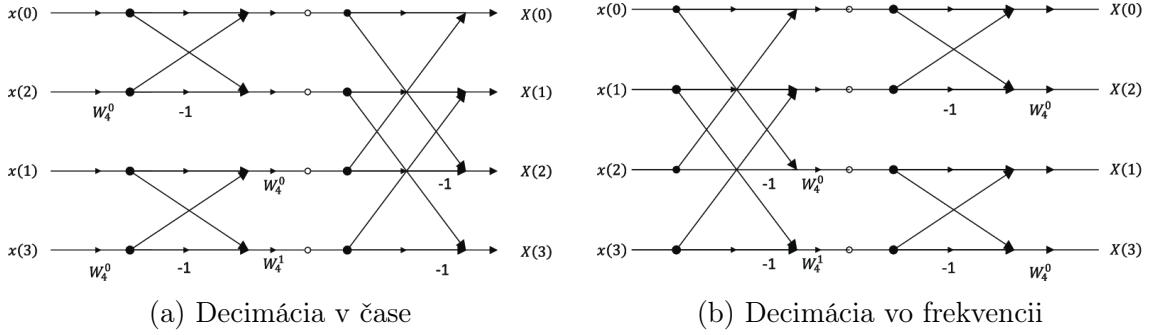
Priamočiarou implementáciou vzťahu na výpočet Fourierovej a kosínusovej transformácie dosiahneme časovú zložitosť rádu $\mathcal{O}(N^2)$. Aplikáciám v reálnom čase na prúdoch dát takáto výpočtová náročnosť zdáleka nepostačuje. Algoritmus rýchlej Fourierovej transformácie (FFT) uplatňujúci prístup rozdeľuj a panuj zvládne zrealizovať DFT v čase $\mathcal{O}(N \log N)$.

Celkovo pozostáva z N sčítaní a $N/2$ násobení v komplexných číslach. Najbežnejšia varianta algoritmu FFT radix-2 vyžaduje, aby veľkosť vstupu bol mocninou dvojkys: $N = 2^k$. Značná výpočtová úspora sa nadobúda uvažovaním s periodicitou goniometrických funkcií pri twiddle faktoroch W_N a z toho vyplývajúcich symetrií Fourierovej matice, keďže len N z N^2 prvkov matice je odlišných [27].

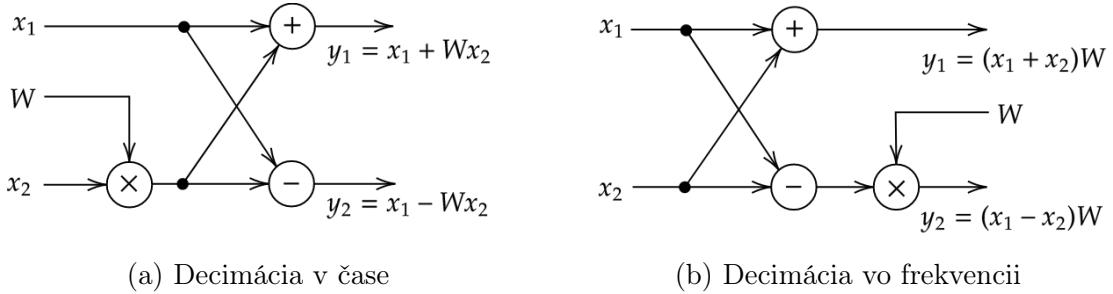
Podľa spôsobu dekompozície vstupného vektora sú známe dve verzie radix-2 FFT nazvané decimácia v čase (DIT) a decimácia vo frekvencii (DIF). Decimácia v čase (obr. 2.7a) rekurzívne delí hodnoty na párne a nepárne pozície v čase, zatiaľ čo decimácia vo frekvencii (obr. 2.7b) rozdeľuje na párne a nepárne frekvenčné vedierka [28]. Výpočet prebieha v $\log_2(n)$ deliacich fázach. Poradie prvkov vo výstupnom vektori je vždy bitovo invertované, čiže poradové číslo zapísané ako bitový reťazec má obrátené poradie. Na dosiahnutie výstupu poradí rastúcich pozícií musí byť pred spustením FFT vstup preusporiadany [27].

Základným prvkom schémy výpočtu je motýlikový diagram („butterfly”), ktorý je odlišný pre DIT (obr. 2.8a) a pre DIF verziu (obr. 2.8b). Motýlik obsahuje vynásobenie jedného z príchodzích operandov s vopred vypočítaným exponenciálnym členom W_N^j pre $j = 0, \dots, N/2 - 1$ [27], a následné prirátanie a tiež odpočítanie od druhého operánu.

Efektívnejšie na celkový počet aritmetický počet operácií oproti FFT s radixom 2 je *split-radix*, ktorý kombinuje výhody vyplývajúce z radix-4 pre nepárne členy DFT



Obr. 2.7: Radix-2 FFT na štyroch bodoch [28]



Obr. 2.8: Motýlikové diagramy algoritmu FFT

a radix-2 pre párne členy. Veľkosť vstupného vektora musí byť násobkom štyroch. Dosahuje okolo 30% zníženie počtu násobení a 10% pokles počtu sčítaní oproti radix-2 [29].

Algoritmus FFT je aplikovateľný taktiež na výpočet DCT vhodným zoradením vstupného vektora. DCT-II N -bodovej reálnej postupnosti \mathbf{x} sa odvodzuje z jej $2N$ -bodového párneho rozšírenia a vynásobením výsledku twiddle faktorom $2W_{2N}^k$ s ponechaním reálnej časti. Ďalej uvádzame prípad DCT 4-bodovej sekvenčie (x_1, x_2, x_3, x_4) , ktorej párnym rozšírením \mathbf{y} je $(x_1, x_2, x_3, x_4, x_4, x_3, x_2, x_1)$. Rovnako by postačovalo vyplniť pôvodnú postupnosť nulami do dĺžky $2N$, čím získame: $(x_1, x_2, x_3, x_4, 0, 0, 0, 0)$. Postačuje však realizovať N -bodovú FFT sekvencie párnych alebo nepárnych prvkov z \mathbf{y} , ktoré sú vzájomným reverzom. Na základe predošlého predošlého príkladu dostaneme postupnosť (x_1, x_3, x_4, x_2) [30].

Široké použitie FFT pri spracovaní signálov sa prejavuje dostupnosťou implementácií algoritmu v širokej škále programovacích jazykov, optimalizovaných pre konkrétné hardvérové platformy. V jazyku C stoja za zmienku knižnice: FFTW, FFTPACK, GNU Scientific Library, CMSIS DSP a Espressif DSP. Na účely analýzy údajov je FFT prítomná pre jazyk Python v balíkoch *numpy* a *scipy*.

2.4.3 Oknové funkcie

U stochastického signálu má zmysel delenie na rovnako dlhé úseky, pretože sa s časom mení jeho spektrálny obsah, ktorý je žiaduce zachytiť čo najpresnejšie. Krát-kodobá Fourierová transformácia zahŕňa preto ováhovanie meraní v časovej doméne koeficientmi posuvnej oknovej funkcie. Mimo intervalu pôsobnosti okna sú vzorky vynulované.

DFT predpokladá periodicitu časového radu do nekonečna, preto ak frekvencia sínusového vstupu nie je presným násobkom frekvenčného rozlíšenia, čiže priebeh exaktne nepripadá frekvenčnému vedierku, dochádza k úniku spektra (spectral leakage). Prejavom je hraničný efekt pre odlišnosť poslednej a prvej vzorky, ktorá je považovaná za nespojitosť a prejavuje sa zvlnením v okolí diskontinuity podľa Gibsovo javu [22].

Existuje množstvo oknových funkcií líšiacich sa mierou kompromisu medzi šírkou výsledných špičiek vo frekvenčnej doméne, presnosti v amplitúde a spôsobu poklesu úniku spektra do ostatných vedierok. Medzi najpoužívanejšie sa zaraďujú: obdlžníkové (2.40), Bartlettovo (2.41), Hannovo (2.42), Hammingovo (2.43) a Blackmanovo okno 2.44. Uvedené okná sú stredovo súmerné (obr. 2.9a). Plochejšie okná, napríklad obdlžníkové, sa vyznačujú ponechaním ostrejších špičiek s neskreslenou amplitúdou za cenu väčšieho spektrálneho úniku, čím sa znižuje odstup od šumu.

Predchádzanie hraničným javom sa dosahuje plynulým znižovaním hodnôt k okrajom okna až na nulu, čím špičky strácajú na amplitúde (scalloping loss) [31].

$$w(n) = 1, \quad n = 0, 1, \dots, N - 1 \quad (2.40)$$

$$w(n) = \frac{2}{N-1} \left(\frac{N-1}{2} - \left| n - \frac{N-1}{2} \right| \right) \quad (2.41)$$

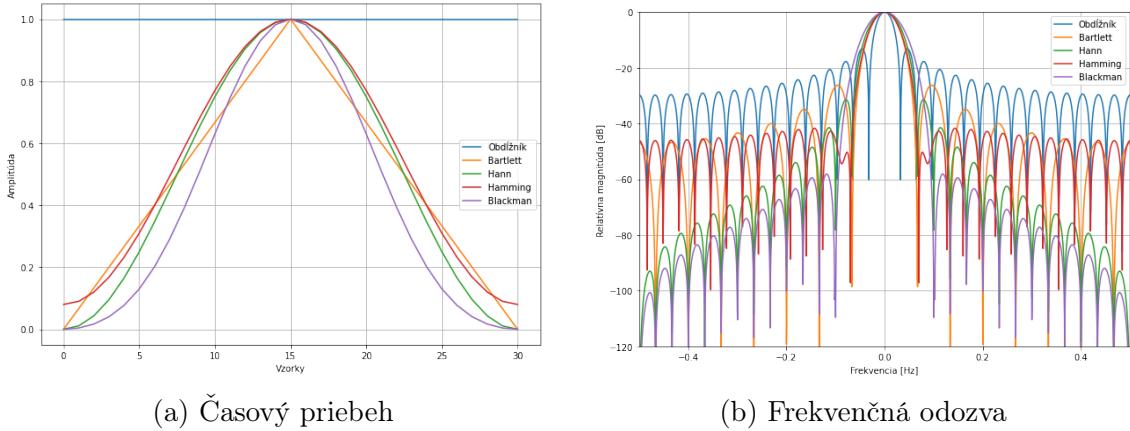
$$w(n) = \sin^2(\pi n/N) \quad (2.42)$$

$$w(n) = 0.54 - 0.46 \cos(2\pi n/N) \quad (2.43)$$

$$w(n) = 0.42 - 0.5 \cos(2\pi n/N) + 0.08 \cos(4\pi n/N) \quad (2.44)$$

Fourierou transformáciou okna dostávame frekvenčnú odozvu, ktorá má tvar funkcie $\text{sinc}(x) = \sin(x)/x$ (obr. 2.9b) Priebeh odozvy sa vyznačuje hlavným a vedľajšími vrcholmi (mainlobe a sidelobes). Hlavný vrchol sa snažia rôzne oknové

funkcie udržať čo najužší, lebo zodpovedá za šírku spektrálneho úniku do okolitých vedierok. Vedľajšie vrcholy sú nežiaduce a podmieňujú najmä úroveň odstupu od šumu.



Obr. 2.9: Tvar oknových funkcií s dĺžkou $N = 31$

Vzhľadom na skutočnosť, že oknové funkcie sa typicky blížia nule smerom k okrajom, bola by veľká časť pozorovaní časového radu ignorovaná. Prekrývaním okien vo vhodnom pomere je umožnený rovnomerný vplyv hodnôt, ktoré pripadnú na okraj niektorého okna. Pomer sa stanovuje štandardne na 50%, aj s ohľadom na rastúcu výpočtovú záťaž s väčším prekrývaním. Výnimkou je obdlžníkové okno kde to nemá zmysel. Platí, že pri iných užších oknách je potrebné rátať s väčším presahovaním ako pri širších. Vyhodnotenie veľkosti prekrývania sa zakladá na korelácii spektrogramov a plochosti amplitúdy, čiže pomeru minimálnej váhy na pozorovanie vo všetkých oknách ku maximálnej dosiahnitej amplitúde, ideálne rovnajúce sa jednotke [31].

Jediný odhad frekvenčných zložiek postupnosti vzoriek viedie k vysokej neurčitosti odhadov pre frekvenčné vedierka, keďže smerodajná odchýlka odhadu je totožná s odhadom samotným [31]. Welchova metóda spriemerovania upravených periodogramov spresní úrovne frekvencií cez priemer viacerých prekrývajúcich sa energetických spektier [32].

2.4.4 Filtre s konečnou impulznou odozvou

Predspracovanie signálu do podoby vhodnejšej na analýzu, extrakciu črt a detekciu udalostí sa vykonáva filtrovaním. Častými činnosťami býva odstránenie posunu

alebo jednosmernej zložky, eliminovanie šumu rozptýleného medzi vysokofrekvenčné komponenty, a oddelenie známeho frekvenčného pásma od zvyšku spektra. Dolná prieplust preustí nízke frekvencie až po medznú frekvenciu, od ktorej nahor frekvencie utlmuje. Horná prieplust sa správa opačne a potláča nižšie frekvencie. Pásmová prieplust ponechá frekvencie v obmedzenom rozsahu z oboch strán.

Ideálne filtre majú okamžitý útlm dovoľujúci prechod striktne vymedzeným zložkám signálu. Vo frekvenčnej doméne preto nadobúdajú tvar obdlžníkového okna. Transformáciou do časovej domény sa obdlžník zmení na konvolučnú masku, resp. impulznú odozvu h , s priebehom *sinc* funkcie, ktorá nie je vyjadriteľná nekonečne presne, čím vznikajú prechodové javy vo frekvenčnej odozve filtra a menšia strmost útlmu s kratším filtrom.

Konečná impulzná odozva v názve FIR filtra znamená, že pri vyjadrení filtra sa obmedzíme na konečný počet koeficientov orezaním impulznej odozvy rovného rádu filtra k . Výpočet upravenej hodnoty sa v časovej doméne počíta ako diskrétna konvolúcia s časovou zložitosťou $\mathcal{O}(nk)$, pre časový rad dĺžky n . Diagram výpočtu je zachytený na obr. 2.10.

$$y[i] = x[i] * h[i] = \sum_{j=0}^k h[j] \cdot x[i-j] \quad (2.45)$$

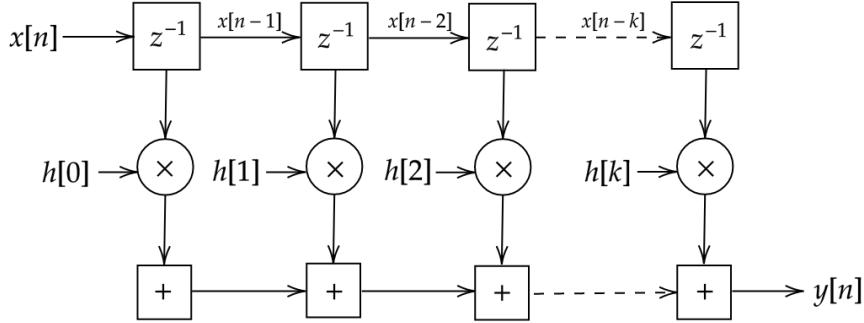
Masky dolnej (2.46), hornej (2.47) a pásmovej priepluste (2.48) popisujú uvedené vzťahy pre medznú normalizovanú frekvenciu: $f_c = f/f_s$. Zvlnenie v prechodovom pásme medzi rozsahmi so ziskom a útlmom je vylepšené návrhom filtra za použitia oknovej funkcie (napr. Blackman) na žiadanú frekvenčnú odozvu.

$$h_{LPF}[n] = \text{sinc}(2f_c n) \quad (2.46)$$

$$h_{HPF}[n] = (-1)^n \cdot h_{LPF}[n] \quad (2.47)$$

$$h_{BPF}[n] = \text{sinc}(2f_{c2} n) - \text{sinc}(2f_{c1} n) \quad (2.48)$$

Podľa konvolučnej vety platí, že konvolúcia v čase je násobením vo frekvenciach, čo umožňuje urýchlenie filtrovania pre veľké masky. Vtedy sa blíži zložitosť konvolúcie ku kvadratickej, ale na násobenie vo frekvenčnej doméne stačí vykonať FFT a IFFT dohromady v rámci $\mathcal{O}(n \log n)$.



Obr. 2.10: Bloková schéma FIR filtra rádu k

2.5 Senzorová siet'

Zber údajov meraní z prostredia zabezpečujú samočinné senzorové jednotky schopné dlhodobej prevádzky často za vystavenia nepriaznivým okolitým podmienkam. Hlavnou limitáciou prevádzky senzorky je spotreba energie, pretože doba funkčnosti zariadenia ohraničuje kapacita batérií, ktoré sú obtiažne vymeniteľné pri nasadení v nedostupných lokalitách alebo pozíciah. Senzor musí byť ideálne schopný autonómnej konfigurácie reakciou na zmenu nastatých okolností a s tým súvisí zotavenie z neočakávaných a chybových stavov [33].

Výpočtový výkon býva za cenu zníženia elektrického odberu redukovaný znížením taktovacej frekvencie a snahou o efektívny manažment periférií distribúciou hodín alebo použitím úsporných režimov. Menej dostupných cyklov procesora povoľuje realizáciu len jednoduchších výpočtov. Navyše niektoré aplikácií aj za týchto okolností očakávajú okamžitú odozvu. Aby sa zachovala nízka cena zariadení šetrí sa na lokálne dostupnom úložisku, ktoré sa počíta v kilobajtoch nanajvýš megabajtoch.

Senzorové jednotky si bud' získané dátá ukladajú na externú flash pamäť, alebo sa od nich vyžaduje komunikácia cez bezdrôtové spojenie. Prepojením na internet sa zaraďujú k zariadeniam Internetu vecí. Na rýchlosť sietového prenosu má dopad okrem šírky pásma a rézie protokolov vzdialenosť od sietovej brány, v prípade hviezdicovej topológie, alebo najbližšieho susedného uzla v mesh alebo point-to-point rozložení. Vynaloženým výkonom na príjem a vysielanie je postihnutý dosah, ktorý ovplyvňujú aj fyzické prekážky na trase a interferencie. Rozšírené bezdrôtové technológie v pásmach ISM sú uvedené v tabuľke 2.1.

Na harmonizáciu využívania rádiového frekvenčného spektra pre zariadenia s

Bezdrôtový protokol	Frekvenčné pásmo v EÚ	Prenos max. (Mbit/s)	Prenos typ. (Mbit/s)	Dosah cca (m)
Bluetooth LE 4	2,4 GHz	1	0,3	10 - 30
Bluetooth LE 5	2,4 GHz	2	1,3	30 - 50
Wifi: 803.11 b	2,4 GHz	11	5	35 - 140
Wifi: 803.11 n	2,4 GHz	54	25	35 - 140
Wifi: 803.11 g	2,4 / 5 GHz	300 / 600	150	70 - 250
ZigBee: 802.15.4	868 MHz 2,4 GHz		20 kbit/s 250 kbit/s	10 - 100
Z-Wave	868 MHz		40 - 100 kbit/s	30 - 100
LoRaWAN	863 MHz		0,3 - 50 kbit/s	5 - 20 km
Narrowband IoT	mobilná sieť		250 kbit/s	1 - 10 km

Tabuľka 2.1: Prehľad najpoužívanejších typov sietí pri IoT komunikácii

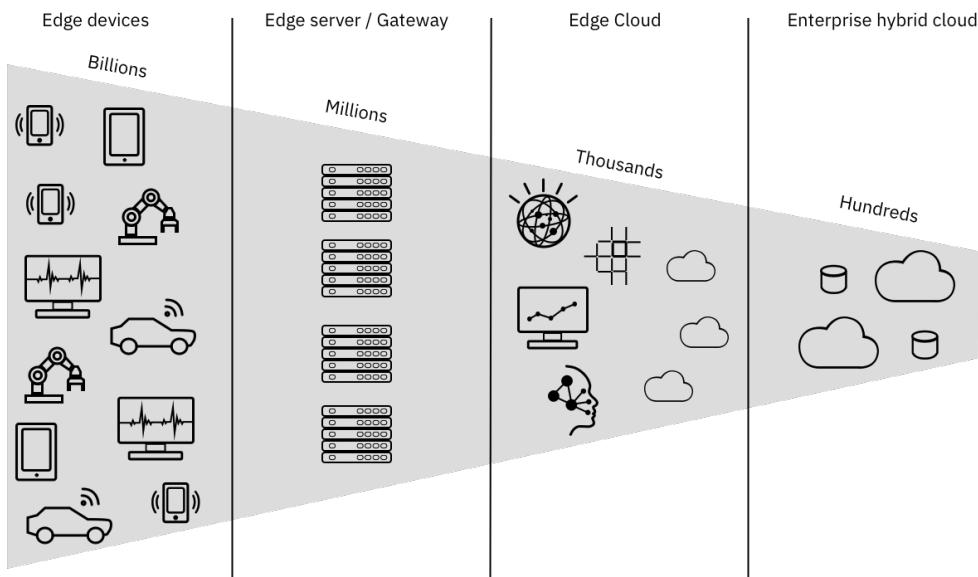
krátkym dosahom sa v Slovenskej republike vzťahuje vykonávacie rozhodnutie Komisie Európskej únie 2019/1345. Definujú sa tam voľné frekvenčné pásma s povolením príslušného maximálneho legálneho vysielacieho výkonu zariadení [34].

V Sub-1 GHz oblasti je k dispozícii rozsah 863 - 870 MHz využívaný LPWAN (Low-Power Wide Area Network) obmedzený časom vysielania na 0.1%, 1%, alebo 10% z hodiny a výkonom do 25 mW. WiFi (IEEE 803.11) a Bluetooth zaberajú rozsah 2400 - 2 483,5 MHz s povoleným výkonom do 100 mW na 100 KHz. Protokoly ako Bluetooth sa navyše označujú triedami podľa ponúkaného dosahu. Trieda 1 deklaruje dosah do 100 metrov za výkonu do 100 mW, trieda 2 je približne do 10 metrov a do 2,5 mW a trieda 3 je na 1 meter a 1 mW [35]. IoT tiež využíva na komunikáciu mobilné siete poskytované operátormi (napr. NB-IoT, GPRS, 4G LTE, 5G), tam sú frekvencie licencované.

Operácia uzlov sa rozdeľuje podľa podnecujúceho činiteľa, ktoré sú založené na udalostiach (event-driven), dopytoch (query-driven) alebo čase (time-driven) [36]. Event-driven systém nepretržite vyhodnocuje vstupy ale upozorní až po záchytení náhlej zmeny alebo prekročení prahovej úrovne. Query-driven systém reaguje na aktuálne požiadavky od používateľa a odpovie so sadou dát zodpovedajúcej požiadavke. Time-driven systém pravidelne odosiela zozbierané údaje do siete podľa nastavení od riadiaceho uzla.

IoT zariadenia na okrajoch siete vytvárajú veľký objem dát, ktorý sa tradične

posiela na zhromaždenie, spracovanie a analýzu na centrálny server alebo do cloutu. Posunom paradigmy s cieľom vyhodnotenia dát, čo najbližšie ku zdroju dát za zníženia latencie pri spracovaní, sieťovej premávky a záťaže na cloudové riešenie, a zvýšením bezpečnosti sa rozširuje „počítanie na okraji“ (edge computing) (obr. 2.11).



Obr. 2.11: Prvky architektúry Edge computing [37]

Edge computing je viacvrstvová distribuovaná architektúra vyvažujúca zodpovednosť a záťaž medzi tri vzájomne sa dopĺňajúce úrovne: Device edge, Local edge a Cloud [38]. Na okraji sieti v rámci device edge pôsobia samotné IoT zariadenia získavajúce dátá z fyzikálnych veličín prostredia a posielajú ich sieťovým edge bránam. Local Edge zahŕňa aktívne sieťové prvky a aplikácie s úlohami, ktoré nie je možné vykonať okrajovými zariadeniami. V cloude sa zhromažďujú dátá do dlhodobého úložiska pre komplexnú a celistvú analytiku. Cloud obsahuje zároveň softvér na spravovanie a monitorovanie zdrojov.

3 Návrh riešenia

V súlade so stanovenými kritériami navrhнемe postupnosť krokov úpravy nameraného pohybu dopravného prostriedku. Zhotovenú konfigurovateľnú sústavu uplatníme na zariadení senzorovej jednotky za účelom ohlasovania udalostí o vybraných pravidelných rysoch signálu. Prihliadať sa bude viac na rýchlu odozvu splnenia úloh pri dosiahnutelnom výkone v dostupnom pamäťovom priestore, ako na energetickú úsporu. Výmena údajov sa má odohrávať širko podporovaným formátom za redukcie nadbytočného sieťového prenosu.

3.1 Špecifikácia požiadaviek

Zariadenie internetu vecí určené na analýzu vibrácií z prostredia bude realizovať nasledujúce funkcionálne požiadavky:

- Zber trojosovej akcelerácie s nastaviteľnou vzorkovacou frekvenciou a dynamickým rozsahom akcelerometra, v hraniciach danými obmedzeniami hardvéru, najmenej však intenzity vyskytujúcej sa pri preprave konvenčnými pozemnými motorovými vozidlami.
- Spracovanie osí akcelerácie nezávisle s obmedzením výberu aktívnych osí.
- Vzdialene realizovateľná zmena parametrov jednotlivých stupňov sústavy na úpravu akceleračného signálu v posuvných oknách.
- Ukladanie nameranej akcelerácie na pamäťovú kartu s ohľadom na najvyššiu dosiahnutelnú rýchlosť zápisu.
- Identifikácia významných frekvencií so zachytením ich trvania a amplitúdy podľa aktuálnej konfigurácie detekčných algoritmov.
- Notifikácia detegovanej udalosti o zmene vibračného spektra bude odoslaná bezdrôtovou sieťovou linkou do 10 sekúnd od objavenia.

- Sumarizácia hodnôt akcelerácie po posuvných oknách do popisných štatistik.
- Odosielanie zackytených udalostí cez spoľahlivé sieťové spojenie za dosiahnutia redukcie množstva produkovaných dát.
- Poskytnutie možnosti odosielania výsledkov z podstatných medzikrokov spracovania za účelom ich poskytnutia ďalším úrovniam senzorovej siete alebo na skupinovú koordináciu meraní z viacerých uzlov.
- Výmena údajov cez sieťový protokol v šandardizovanom formáte hierarchickej štruktúry za najmenšej uskutočniteľnej rézie.

Z povahy okolností nasadenia firmvéru na relatívne zdrojovo oklieštené Edge zariadenie vyplývajú vymenované nefunkcionálne požiadavky, prevažne na účinnosť a prenositeľnosť:

- Firmvér sa zmestí do programovej pamäte s rezervou pre budúce rozširovanie detekčnej funkcionality.
- Ľubovoľný scenár spracovania musí prebehnuť v reálnom čase rádovo v jednotkách sekúnd.
- Platformová závislosť sa obmedzí na nevyhnutné súčasti systému ako sú hardvérové ovládače a akcelerácia náročných výpočtov.

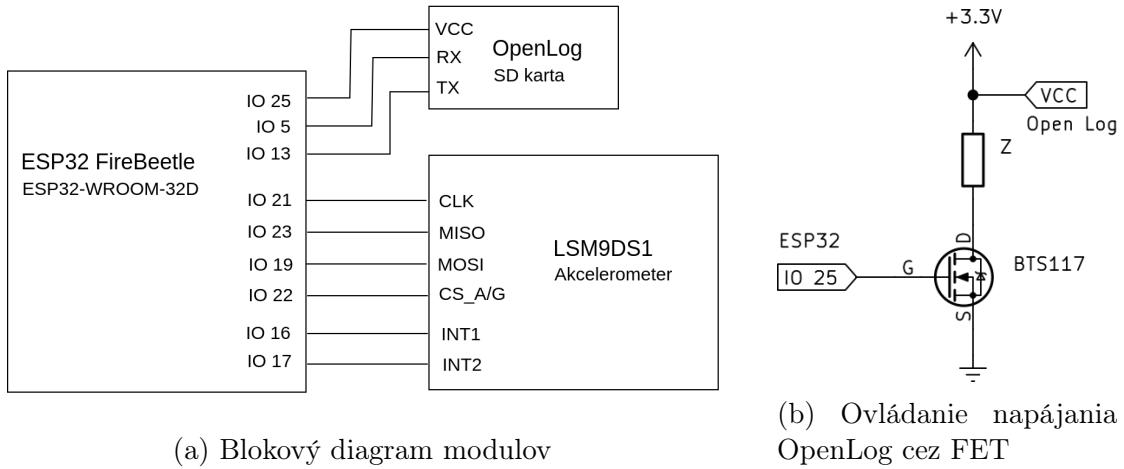
3.2 Hardvér senzorovej jednotky

Navrhované zariadenie je postavené na platforme mikrokontroléra ESP32 od Espressif. Za relatívne nízkej obstarávacej ceny ponúka možnosť konektivity na 2,4 GHz s Wifi 802.11 b/g/n a Bluetooth 4.2. V porovnaní s podobnými zariadeniami disponuje nebývalým výpočtovým výkonom a kapacitou pamäti. Univerzálny plošný spoj osadený kontrolérom a neskôr zmienenými komponentami je zabezpečený externým zhotoviteľom.

Konkrétnie je systém postavený na doske FireBeetle osadenej modulom ESP32-WROOM-32D s typickým napájacím napätím 3,3 V a dvoj-jadrovým 32-bitovým procesorom Xtensa s taktovacou frekvenciou od 80 do 240 MHz. Modul obsahuje až 520 kB SRAM logicky rozdelenej na 192 kB IRAM časť pre inštrukcie a 328 kB DRAM na dátu.

Použitý model akcelerometra je súčasťou MEMS inerciálnej meracej jednotky

LSM9DS1 (pozri 2.1.4), zabudovanej na adaptéri STEVAL-MKI159V1 pre púzdro DIL24. Akcelerometer komunikuje s MCU cez poloduplexnú SPI zbernicu s maximálnou frekvenciou hodín do 10 MHz. Navyše sa zapoja vývody prerušení INT1 a INT2 pre upozornenie prekročenia určených prahových úrovní. Blokový diagram zapojenia zachytáva schéma 3.1a.



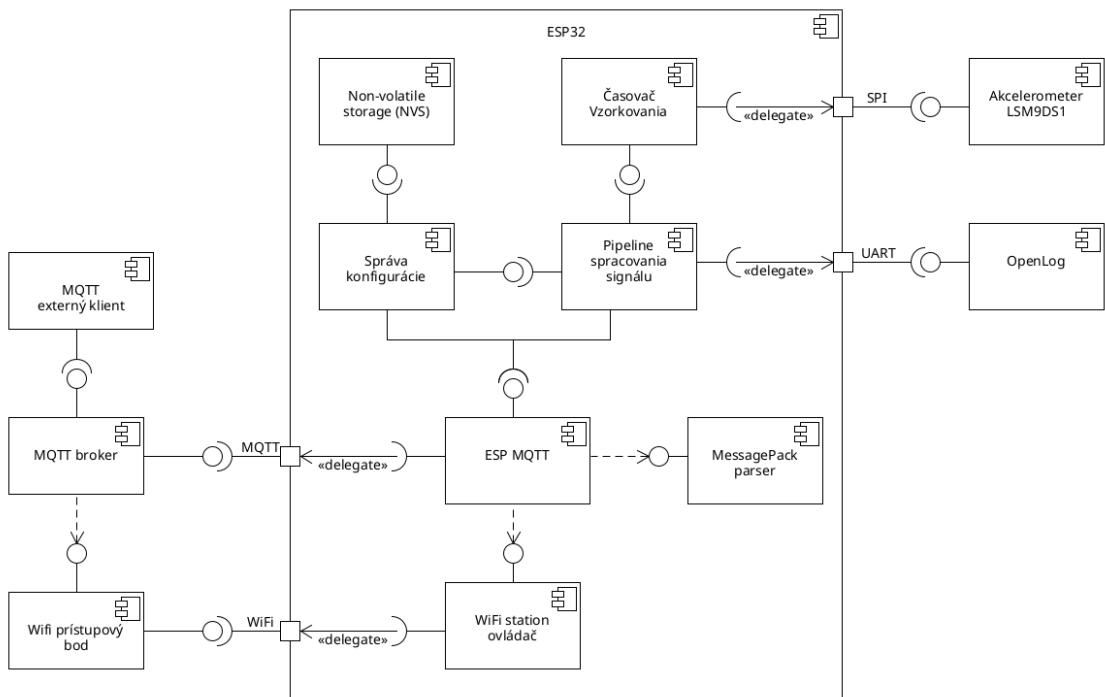
Obr. 3.1: Schéma zapojenia hardvéru

Pamäťová Micro SD karta so súborovým systémom FAT32 bude pripojená v module OpenLog od Sparkfun, ktorý zaznamená znaky prijímané cez UART do textových súborov podľa pravidiel zo súboru `config.txt` alebo povelmi odoslanými po zapnutí. Ukladanie na externé médium nie je vždy žiaduce, preto bude napájanie spínané cez pin mikrokontroléra. Vyšší prúdový odber než dodá výstup a požadované napätie rovné s napájacím napäťom procesora vyžaduje umiestnenie tranzistora riadeného polom N-kanál BTS117 na premostenie riadiaceho signálu (obr. 3.1b).

3.3 Architektúra systému

Celková skladba komponentov systému (3.2) pozostáva zo súčastí pôsobiacich na mikrokontroléri ESP32 interagujúcimi cez lokálne sériové zbernice s akcelerometrom a zapisovačom. Spracované vektorové akcelerácie sú odosielané do počítačovej siete prostredníctvom prístupového bodu WiFi aplikáčnym protokolom MQTT na server so službou broker správ. Odberané témy sú odtiaľ rozšírené klientom metódou publish-subscribe.

Odčítavanie úrovní z akcelerometra zabezpečuje **časovač vzorkovania**, ktorý si



Obr. 3.2: Komponenty navrhovaného systému

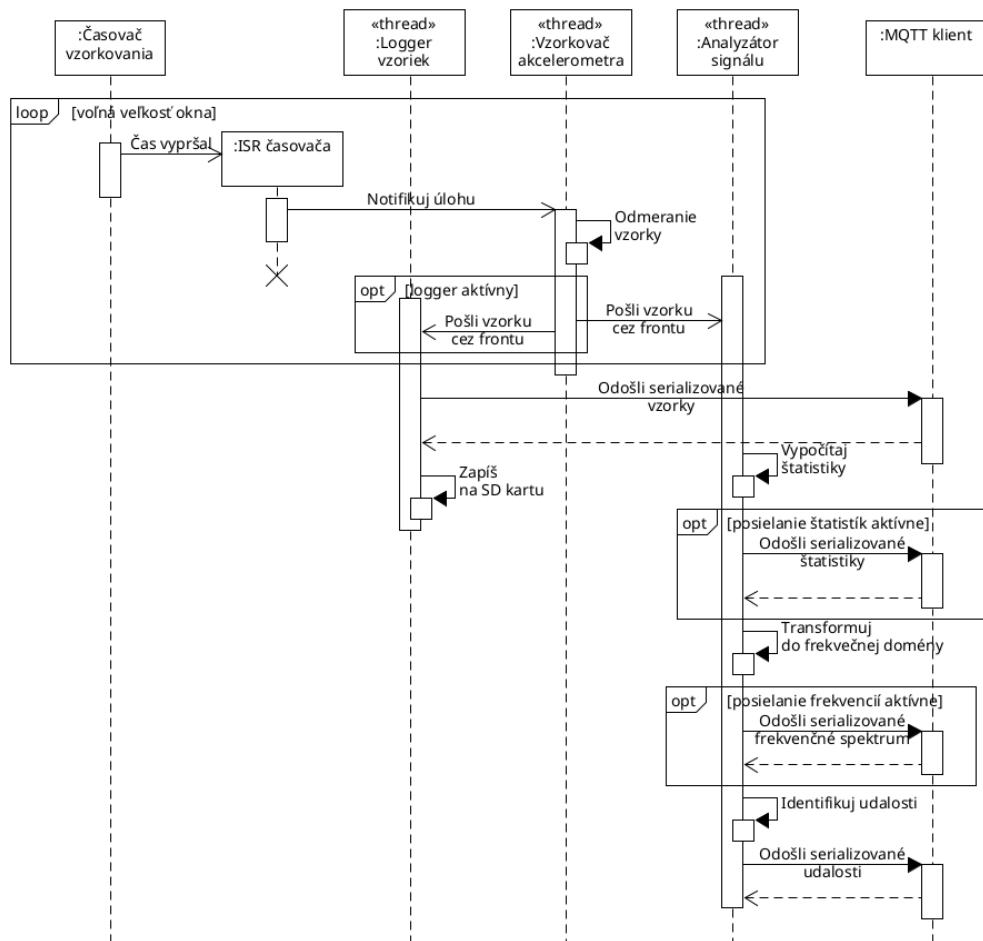
v pravidelných intervaloch pýta aktuálne hodnoty rozhraním periférneho adaptéra pre prístup ku SPI zbernicí. Po vypršaní vzorkovacej periódy je zavolaná obsluha prerušenia, ktorá odblokuje vlákno úlohy na synchronné načítanie okamžitého vektora zrýchlenia konvertovaného z číslicovej úrovne prevodníka na metre za sekundu na druhú (3.3). Zachytené hodnoty sú preposlané cez thread-safe fronty analyzátoru signálu zvlášť pre každú priestorovú os akcelerácie. V prípade zachytávania časového priebehu s voliteľným podvzorkovaním sa vektor umiestni do frontu do vlákna loggera.

Pipeline spracovania signálu (3.2) rozdeľuje vzorky do prelínajúcich sa posuvných okien, počíta z nich štatistiky a vyhľadáva udalosti vo frekvenčnom spektre. Nespracované hodnoty sú podľa potreby ukladané na pamäťovú kartu.

Správa konfigurácie zaobstaráva zmenu a uchovanie parametrov pipelinov pre jednotlivé bloky spracovania. Modifikované nastavenia sú medzi spusteniami zachované vo vyhradenej partícii nevolatilnej flash pamäte na záznamy dvojíc v asociatívnej štruktúre. Predvolené správanie načítané za nedostupnosti konfigurácie z flash úložiska určujú konštanty v programovej pamäti.

Binárny serializačný formát Message Pack zaobala vzorky, udalosti a konfiguráciu posielané na rozličné MQTT témy. Vychádza z formátu JSON (JavaScript

Object Notation), ale naroziel od neho sa sústredí na efektívne kódovanie dátových typov. Namiesto prevodu číselných údajov do znakového kódovania, napr. Unicode, ponecháva ich pôvodnú binárnu reprezentáciu so štandardom špecifikovanými bytovými značkami určujúcimi typ údaju. Hodnoty vyjadriteľné menším počtom bajtov reprezentuje dokonca v kratšom tvare než podmieňuje ich celkový rozsah. V zoznamoch a slovníkov sa zaobchádza bez oddelovacích znakov, ktoré nahradza informáciou o počte údajov. Dĺžka položky predchádzajúca zloženými atribútom uľahčuje následné parsovanie.



Obr. 3.3: Sekvenčný diagram vzorkovania signálu a spolupráce úloh

3.4 Fázy dátovej pipeline

Poskladaná dátová pipeline sa sústredí primárne na odhalenie pretrvávajúcich spektrálnych zložiek. Úloha jednotlivých zakomponovaných blokov spočíva v prevedení

pôvodne obdržaných vzoriek podľa globálnych pravidiel definujúcich ich správanie.

Vo všeobecnosti sa v nej nachádzajú bloky filtrácie, transformácie a serializácie signálu. Sústava sa vyznačuje modulárnosťou, čiže umožňuje doplnenie dodatočných etáp, ale iba v jednej na seba nadväzujúcej línii naraz, so spoločnou veľkosťou posuvného okna. Diagram aktivít 3.4 vizualizuje navrhovaný beh činností na zariadení.

3.4.1 Nastaviteľné vlastnosti

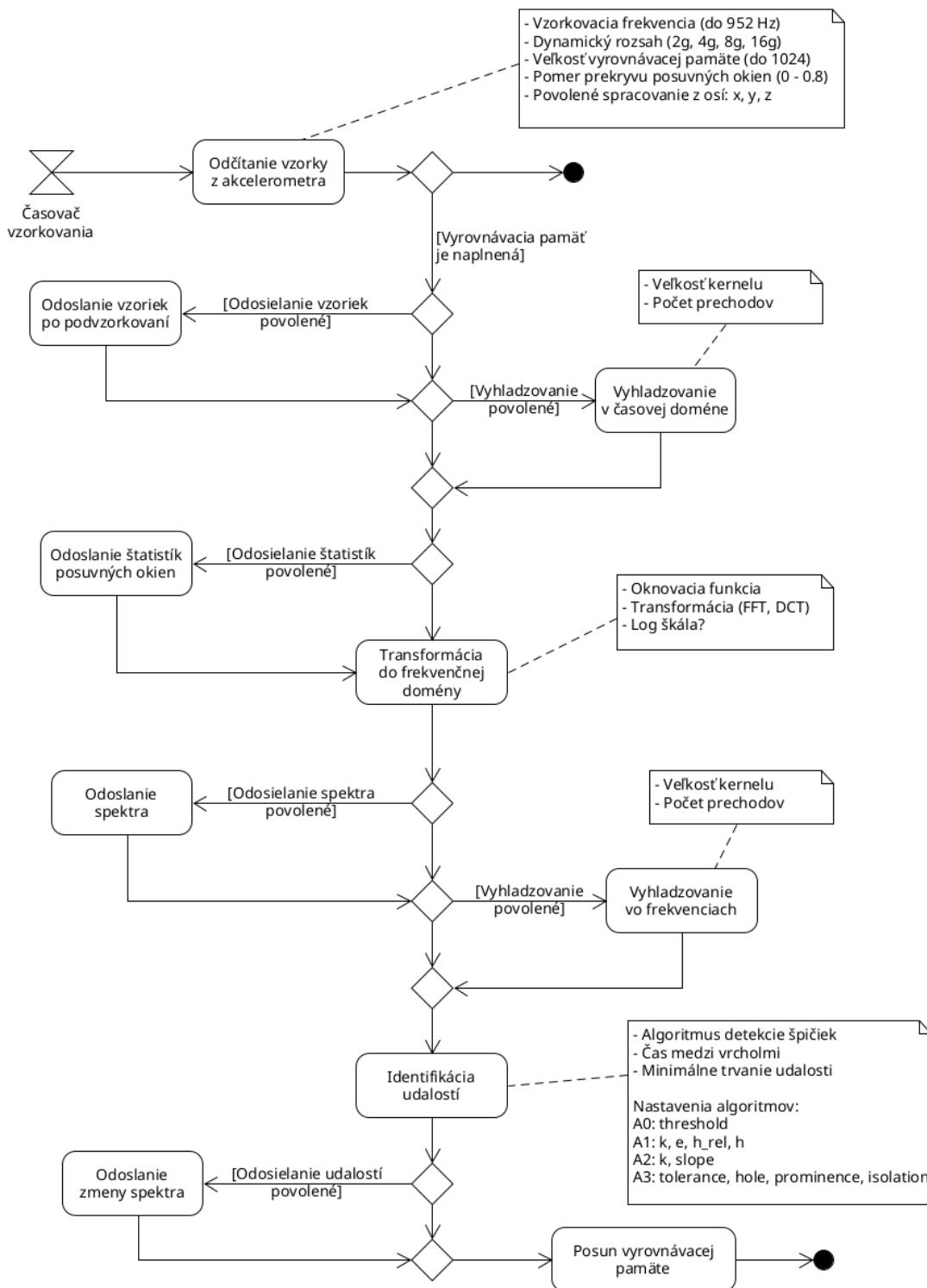
Utváranie charakteru výstupov sa deje hned na začiatku voľbou nastavení snímača. Vzťahuje sa naň vzorkovacia frekvencia časovača, podmieňujúca výstupný dátový tok konštrukčne obmedzený do 952 Hz, a dynamický rozsah v rozmedzí do 16g. Keďže sú priestorové osi zrýchlenia analyzované nezávisle, vyžadujú sa pre každú ďalšiu dimenziu systémové prostriedky navyše. Preto je výhodná možnosť aktivácia len niektorých osí.

Postupnosť hodnôt je rozkúskovaná podľa dĺžky vyrovnávacej pamäte pre prevod do frekvenčnej domény s počtom slotov o mocnine dvojky. Pomer prekryvu okien sa odporúča od 0, čo znamená bez zanechania predošlého obsahu, nanajvýš do 0,75 výhradne pre úzke oknové funkcie.

Po naplenení miest v cyklickej fronte sa pristúpi k prenásobeniu bodov s koeficientami okbovej funkcie z ponuky: obdlžnik, Bartlett, Hann, Hamming, Blackman, a následnej Fourierovej alebo kosínusovej transformácii o veľkosti totožnej s dĺžkou okna. Z frekvenčných vedierok sa zistí magnitúda v lineárnej alebo decibelovej škále. Pred a po transformácii sa môže doplnkovo uplatniť filter vyhľadzovania priemerom, v časovej i vo frekvenčnej doméne. Opierajú sa o predpísanú veľkosť masky filtra a počet prechodov koľkokrát má byť opakovane aplikovaný.

Identifikácia udalostí pozostáva z binárnej klasifikácie úrovni frekvenčných vedierok podľa prítomnosti lokálneho maxima a zo zlúčenia súvislého výskytu naprieč dlhším časovým intervalom do udalosti. Klasifikácia sa realizuje aktuálne nastaveným algoritmom na hľadanie špičiek podľa kombinácie potrebných číselných parametrov. K dispozícii sú algoritmy: nad prahovú úroveň, najvyššieho spomedzi susedov, prechodu nulou do záporu alebo horského turistu.

Medzi dôležitými fázami pipeline dochádza podľa povolených modulov na odosielanie správ k odovzdaniu medziproduktov na formátovanie cez Message Pack a



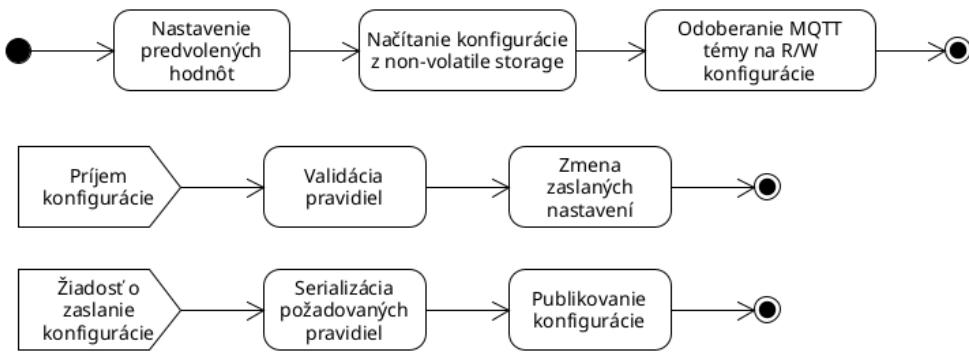
Obr. 3.4: Postup spracovania zaznamenaných vibrácií

synchrónne publikovanie na MQTT tému. Umožňuje sa poslanie vzoriek po decimácii celočíselným faktorom ≥ 1 , požadovaných štatistik z časového priebehu (minimum, maximum, stredná kvadratická odchýlka, priemer, rozptyl, smerodajná odchýlka, šikmost, špicatosť, medián, mediánová absolútна odchýlka, medzi-osová korelácia),

výsledku frekvenčnej transformácie alebo udalostí o zmene významných frekvencií.

Reguláciu toku dát a vlastnosti úpravy v blokoch pipeline ovplyvňuje globálna systémová konfigurácia. Obvykle sa nahrá z nevolatilnej pamäte, ale je umožnené aby pravidlá boli upraviteľné vzdialene správami vo formáte Message Pack (3.5). Zariadenie sa prihlási na odber MQTT témy na zmenu konfigurácie. Postačuje, aby na tému klient odoskal vlastnosti, ktoré mieni upraviť. Ak sú parametre syntakticky korektné a spadajú do dovoleného číselného intervalu príde k reštartu zariadenia a ich následnému uplatneniu. V opačnom prípade sa na samostatnú MQTT tému odošle chybová hláška.

Vzdialený klient si taktiež dokáže zobraziť všetky momentálne platné pravidlá odberom témy *config/request* a publikovaním na tému *config/response*, čím sa imituje query-driven vzor požiadavka – odpoveď.



Obr. 3.5: Príjem nových pravidiel a dopytovanie systémovej konfigurácie

3.4.2 Preskúmané obmeny pipeline

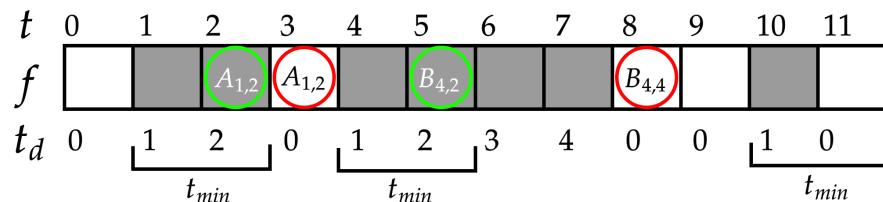
Uvažovalo sa o vložení voliteľného kroku zníženia šumu v transformovanom spektre Welchovou metódou, ktorý by vedel prispieť k eliminácii záchvevov krátkodobého výpadku frekvenčnej zložky. Nespolupracuje však dobre s ideou zdieľanej veľkosti okna a ich prekryvu. Vo všeestrannom riešení môžu byť body delené do viacerých segmentov, čo naráža na pamäťové obmedzenia zariadenia pri viacerých alebo dlhších segmentoch. Zároveň sa znižuje presnosť v časovej oblasti, ktorú by bolo potrebné zohľadniť v časových pečiatkach udalostí.

Exploratívnu analýzu sa preukázalo, že v uvažovanom kontexte dopravy nemá zmysel extrahovať zo snímaného zrýchlenia odhad o rýchlosťi a polohe numerickou

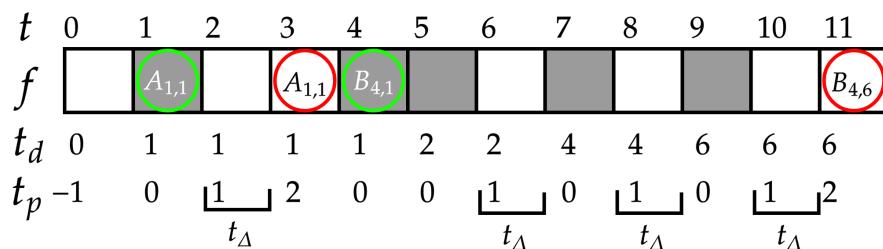
kvadratúrou korigovanou obálkami, respektíve pre vysoký šum sú úrovne veličín nerealistické. Korekcia má opodstatnenie iba pre stacionárne signály vytrácajúce sa v situáciach, keď sa os akcelerácie odchýli medzi rovnovážnymi bodmi, napríklad auto zastavené na svahu.

3.4.3 Prúdový algoritmus detekcie zmien frekvencií

Detekcia špičiek sa vzťahuje na frekvenčné spektrum práve prebiehajúceho posuvného okna, čím postráda širší pohľad na trválosť harmonických zložiek v časovo-frekvenčnom priebehu. Okrem toho prirodzene dochádza k dočasným výpadok v intenzite frekvenčného vedierka, či už skutočným prerušením, výkyvom do vedľajších vedierok, alebo spôsobeného nespoľahlivosťou označenia lokálnych extrémov. Všetko sú to javy, ktoré je žiaduce potlačiť v tolerovaných medziach. Doteraz videné body nie je akceptovateľné odložiť v svojej celistvosti, či úplnosti, aj s ohľadom na promptné ohlásenie udalostí.



(a) Minimálne trvanie udalosti $t_{min} = 2$



(b) Maximálna vzdialenosť špičiek $t_{\Delta} = 1$

Obr. 3.6: Parametre algoritmu na detekciu udalostí

Vývoj detegovaných vrcholov v diskrétnej frekvencii sa dá znázorniť ako binárna sekvencia, kedy prítomnosť vrchola na časovom úseku označíme logickou jednotkou, na obr. 3.6 tieňované šedou farbou. Zavedieme dve premenné definujúce aká postupnosť detekcií klasifikátora špičiek je považovaná za súvislú udalosť.

Parameter t_{min} je najkratšie akceptovateľné trvanie udalosti a predurčuje one-

skorenie notifikácie od počiatku objavenia sa zreteľne vyčlenenej magnitúdy. Čas t_Δ je najdlhšia medzera medzi špičkami tak aby pretrvávajúca udalosť nebola pri zákmite ukončená alebo príliš krátka zahodená, ale hluché miesto sa má preklenúť. Tiež vplýva na omeškanie upozornenia na záver udalosti. Obe premenné sú celočíselné a uvádzajú sa v počte posuvných okien. Najjednoduchší prípad, kedy je udalosťou neprerušovaná postupnosť jedničiek platí pri $t_{min} = 1$ a $t_\Delta = 0$.

Algoritmus 4 Detektor zmeny frekvenčnej zložky

Vstupy: $event, bin, t, t_{min}, t_\Delta$

```

1: if V predošlom okne  $t - 1$  bola emitovaná udalosť Koniec then
2:    Vynuluj udalosť:  $event.duration \leftarrow amplitude \leftarrow 0, lastSeen \leftarrow -1$ 
3: end if
4: if IsPeak(bin) then
5:     $link \leftarrow \max\{1, event.lastSeen + 1\}$ 
6:    if  $event.duration < t_{min} \leq event.duration + link$  then
7:        $event.start \leftarrow t - event.duration - link + 1$ 
8:       Emituj udalosť Štart výskytu frekvencie podľa event
9:    end if
10:   Inkrementuj  $event.duration$  o  $link$ 
11:   Inkrementuj  $event.amplitude$  o  $(bin - event.amplitude) / event.duration$ 
12:    $event.lastSeen \leftarrow 0$ 
13: else if  $event.lastSeen \geq 0$  then
14:    Inkrementuj  $event.lastSeen$  o 1
15:    if  $event.lastSeen > t_\Delta$  then
16:       if  $event.duration \geq t_{min}$  then
17:          Emituj udalosť Koniec výskytu frekvencie podľa event
18:       end if
19:    else
20:       Vynuluj udalosť:  $event.duration \leftarrow amplitude \leftarrow 0, lastSeen \leftarrow -1$ 
21:    end if
22: end if

```

S rastom t_{min} sa udalosť A , začínajúca v posuvnom okne s poradovým číslom $t = 1$, emituje (zelený kruh) až o $t_{min} - 1$ políčok neskôr (3.6a). Značka konca (červený kruh) je vytvorená ihneď s ukončením súvislého radu špičiek. Krátke udalosti, napr. s eventuálnym štartom v čase $t = 10$, sú takto ignorované. Dolný index označenia udalosti sa skladá z dvoch čísel: čas začiatku udalosti a jeho predbežné trvanie: A_{t,t_d} .

Za udalosť B sa zvyšovaním t_Δ považuje podľa 3.6b celý úsek od $t = 4$ s trvaním $t_d = 6$. Premennou t_p sa sleduje koľko miest do minulosti bola naposledy zistená jednička, za predpokladu neukončenej udalosti. Po inicializácii alebo prekročením hranice preskočiteľného rozostupu: $t_p > t_{Delta} + 1$, sa musí t_p rovnať -1 . Priemerná

amplitúda frekvencie sa počíta ako bežiaci priemer.

3.5 Datasetsy

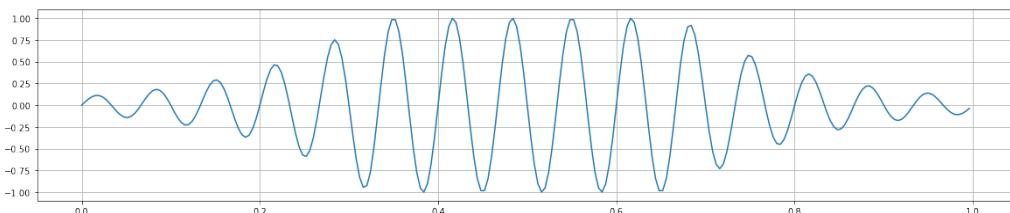
Úspešnosť klasifikácie odhadneme pomocou syntézy digitálneho časového radu so pseudonáhodným spektrálnym profilom. Vhodnosť navrhovaných metód detekcie udalostí vo frekvenčnej doméne posúdime podľa nameraných vibrácií z reálnej premávky električiek a autobusov mestskej hromadnej dopravy.

3.5.1 Syntéza časovo-premenného spektrálneho profilu

Exaktná manuálna anotácia vibračného záznamu je úmorný proces vedúci k nejednoznačným rozhodnutiam, ktoré vrcholy považovať za postačujúco vyčlenené od ostatných bodov. Kontrola presnosti sa však spolieha na zdroj pravdy výskytov harmonických komponentov.

Vygenerovaním predpisu rozmiestníme náhodnú sadu istého počtu frekvencií, s hodnotami do polovice definovej vzorkovacej frekvencie. Celkové trvanie časového radu arbitrárne rozparcelujeme na desať úsekov, pričom na každom segmente vymedzíme posun začiatku a konca, pre všetky frekvenčné zložky zo sady, náhodne až do dvoch pozícíí vzad alebo vpred. Amplitúdy sa zvolia z rozsahu 0,2 až 2 jednotky.

$$y = y_{max} \cdot \sin(2\pi f \cdot T_s \cdot i); \quad \forall i \in \mathbb{N} \wedge i \leq f_s t_d \quad (3.1)$$



Obr. 3.7: Základný tón v syntetickom signále

Diskrétne sinusoidy s exponenciálnym nábehom amplitúdy (obr. 3.7) mixujeme do postupne zlučovaného vlnového priebehu podľa pripraveného zoznamu pravidiel. Nábeh a dobeh ovplyvňujú tretinu dĺžky vlny. Vzorky základného tónu s trvaním t_d sú počítané z rovnice periodickej oscilácie (3.1), kde i -ty bod v poradí formuje očakávaná frekvencia f za vzorkovacej periódy T_s a maximálnej amplitúdy y_{max} .

Skutočné signály postihujú neurčitosti, ktoré napodobíme pridaním normálne rozdeleného bieleho šumu s úrovňou do 0,1 jednotiek.

Označkovanie aktivity vo frekvenčných vedierkach vznikne rovnako z generovanych deklarácií rozloženia frekvencií v intervaloch posuvných okien.

3.5.2 Zber vibrácií z premávky

Verný obraz o silách pôsobiacich na prevážané predmety nadobudneme meraniami priamo v doprave. Umiestnením snímača na podlahu v kabíne pre cestujúcich v blízkosti nápravy alebo nad motorom zachytíme otrasy najzreteľnejšie.

Záznam ukladaný na pamäťovú kartu spustíme pripojením napájania odporom zaťaženej power banky. Do novovytvoreného textového súboru sa zapíšu trojrozmerné súradnice vektora zrýchlenia v m/s^2 oddelené medzerami s presnosťou na 3 desatinné miesta. Os x je orientovaná proti smeru jazdy, os y mieri doľava, čo predurčuje smer osi z dohora. Dynamický rozsah akcelerometra 2 g sa odvodil z prieskumných meraní aplikáciou na mobilnom telefóne.

Na riadkoch súboru sú desatinné čísla so znamienkom do ± 20 prevažne 6 miestne, čím dosiahne jeden záznam súradníc dĺžku do 21 znakov. OpenLog stíha zapisovať pri 115 000 baud, jednom štart a stop bite na slabiku, teoreticky do vzorkovacej frekvencie 547 Hz. Pri zachovaní rezervy limitujeme záznam na 500 Hz.

4 Implementácia

Firmvér senzorovej jednotky je implementovaný v programovacom jazyku C so SDK Espressif IoT Development Framework (ESP-IDF). Súbežný beh úloh spravuje operačný systém reálneho času FreeRTOS. Optimalizované rutiny spracovania signálu poskytuje Espressif DSP Library. Knižnica MPack má na starosti kódovanie a dekódovanie formátu Message Pack. CMake riadi zostavovanie modulov zdrojového kódu. Eclipse Mosquitto pôsobí ako MQTT broker správ.

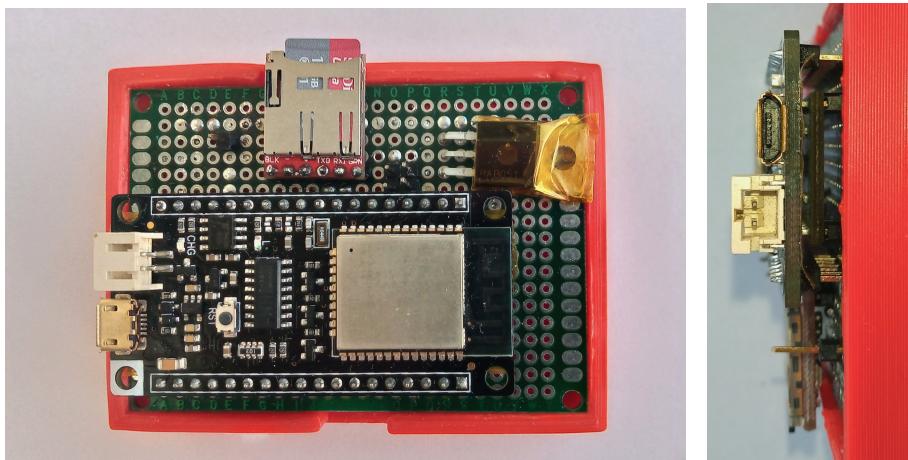
V jazyku Python sú napísané Jupyter notebooky na analýzu zozbieraných datasetov a otestovanie fáz navrhnutej dátovej pipeline, so závislosťami numpy, scipy, pandas a matplotlib. Rozhranie príkazového riadku na nahrávanie konfigurácie a náhľad odoberaných správ sa spolieha na balíčky Paho MQTT, cmd a msgpack.

4.1 Senzorová sieť

Súčiastky FireBeetle ESP32, OpenLog, STEVAL-MKI159V1 a BTS117 sú naspájkované na univerzálny plošný spoj rozmerov 5 x 7 cm (obr. 4.1). Doska je vsadená do plastovej krabičky s hrúbkou stien 3 mm vo výške 2 cm nad povrchom. Akcelerometer je namontovaný tesne pod modulom MCU. Externý 5 V zdroj sa pripája cez Micro USB konektor, alebo 3,7 V lítiovú batériu zapojíme cez JST PH 2 pin.

Po zapnutí si firmvér načíta systémové nastavenia cez SDK Storage API z flash. Inicializuje sa akcelerometer a naraz sa kompletne alokuje dynamická pamäť pre výpočty dátovej pipeline a pre synchronizačné primitíva. Na zamedzenie fragmentácie nie je odvtedy programom prideľovaná žiadna ďalšia pamäť okrem zásobníkov úloh.

Prebehne pokus o pripojenie s prístupovým bodom WiFi so zabezpečením WPA2 a so serverom MQTT broker, na základe prihlásovacích údajov a URL adresy v štruktúre Provisioning. Proces spustenia OpenLog čaká po zopnutí napájania



Obr. 4.1: Univerzálny plošný spoj v krabičke osadený modulmi

10 sekúnd na uvedenie periférie do prevádzkyschopného stavu.

```
Provisioning login = {
    .wifi_ssid="AccessPoint",
    .wifi_pass="password",
    .mqtt_url="mqtt://192.168.1.2:1883"
};
```

Priame prepísanie predvolenej hodnoty systémového nastavenia v zdrojovom kóde sa po nahratí firmvéru neprejaví za behu. Predtým sa musí naflashovať obslužný program vyvolaní direktívou FACTORY_RESET pri kompliacii, ktorý premietne tieto nastavenia do partície nevolatilnej pamäte.

Broker MQTT správ Eclipse Mosquitto nasadený v lokálnej sieti, má umožnené cez konfiguračný súbor mosquitto.conf počúvať premávku zo všetkých IP adres na TCP porte 1883, bez nutnosti klientov sa autentifikovať:

```
listener 1883 0.0.0.0
allow_anonymous true
```

Vlastný MQTT klient config_tool.py je interpreter príkazov na interaktívnu interakciu so senzorovými jednotkami pripojenými na broker. Ponúka nadstavbu nad binárnymi správami v Message Pack konverziou z a do ľudsky čitateľnejšej podoby vo formáte JSON. Povelom connect dôjde k nadviazaniu spojenia s brokerom správ, za filtrovania topics podľa zadaného identifikátora zariadenia. Prefix MQTT tém definuje konštanta firmvéru DEVICE_MQTT_TOPIC, ktorý konvenčne začína s „imu/[Device ID]“.

Režim na nastavenie systémovej konfigurácie IoT koncového uzla sa vyvolá po-

velom set a dopyt aktuálnych pravidiel príkazom config. Po aplikovaní zmien sa čaká sa opäťované nabehnutie systému alebo chybovú hlášku. Zobrazenie jednotkou publikovaných údajov na MQTT tému alebo skupinu tému sa určuje povelom topic.

Na SD karte data loggera sa nachádza súbor config.txt s uvedenou znakovou rýchlosťou rovnakou ako pre UART zbernicu mikrokontoléra. Preferované nastavenia sú najvyššia prenosová rýchlosť 115000 v móde 0 zakladajúcim nový log súbor po reštarte:

```
115200,36,3,0,1,1,0
baud,escape,esc#,mode,verb,echo,ignoreRX
```

4.2 Komunikácia medzi úlohami

Nezávislé činnosti aplikácie sú prerozdelené medzi vlákna, ktoré si cez fronty posielajú súradnice zrýchlenia. FreeRTOS úlohy sú funkcie vykonávajúce opakované sekvenčiu príkazov v nekonečnom cykle.

Procedúra obsluhy prerušenia hardvérového časovača nesmie čakať, preto je načítaná rovno z IRAM a upovedomí úlohu vzorkovania (kód 4.1). Prevzatím notifikácie synchrónne pošle riadiace slovo senzoru na sekvenčné odčítanie celého vektora akcelerácie od bázovej adresy registra x-ovej osi. Získaná trojica 16-bitových slov sa podľa aktuálneho rozlíšenia prevedie do štandardnej fyzikálnej jednotky v type float.

```
1 // Sampling task
2 if (ulTaskNotifyTake(pdTRUE, portMAX_DELAY)) {
3     imu_acceleration(&imu, &axis[0], &axis[1], &axis[2]);
4     for (i = 0; i < AXIS_COUNT; i++) {
5         if (conf.sensor.axis[i])
6             xQueueSend(pipeline.queue[i], &axis[i], 0);
7     }
8 }
9 // Pipeline task
10 if (xQueueReceive(k->queue[x], &p.stream[idx], portMAX_DELAY)) {
11     if (++idx < conf.sensor.n) continue;
12     // Process buffer p.stream
13     buffer_shift_left(p.stream, conf.sensor.n, leftover);
14     idx = conf.sensor.n - leftover;
15 }
```

Zdrojový kód 4.1: Posielanie vzoriek medzi úlohami cez fronty

Vkladanie vzoriek do fronty neblokuje, lebo sa nepočíta s úplným vyčerpaním voľných slotov. Posiela sa do fronty iba vtedy, ak beží úloha pipeline pre danú dimenziu. Na opačnom konci fronty sa čísla po jednom pripájajú do cirkulárnej vyrovňavacej pamäte. Pole sa prenechá zvyšku úlohy na spracovanie až po naplnení dosiahnutím `conf.sensor.n` položiek.

Dokončením analýzy posuvného okna sa presunú ponechané hodnoty z prekryvu časových úsekov na začiatok poľa: $\text{leftover} = n \cdot (1 - \text{overlap})$. Ďalej sa pokračuje prepisovaním už nepotrebných čísel od pozície `idx`.

```

1 xEventGroupSync(barrier, (1 << axis), task_mask, portMAX_DELAY);
2     float avg = mean(buffer, n);
3     std[axis] = sqrt(variance(buffer, n, avg));
4     for (uint16_t i = 0; i < n; i++)
5         diff[axis][i] = (buffer[i] - avg);
6 xEventGroupSync(barrier, (1 << axis), task_mask, portMAX_DELAY);
7
8 if (axis[0] && axis[1])
9     stats->corr_xy = correlation(diff[0],diff[1],n,std[0],std[1]);

```

Zdrojový kód 4.2: Synchronizácia úloh na výpočet korelácie osí

Okrem synchronizácie úloh posielaním správ cez fronty sa zužitkovajú bariéry. Event Groups riadia toku synchronizovaných úloh v bodoch stretu čakaním na nastavenie bitov podľa očakávanej bitovej masky. Počas autorizácie voči prístupovému bodu WiFi čaká podprogram hlavného vlákna na príznak pridelenia IP adresy od obsluhy udalosti nadviazania spojenia.

Koordinácia vlákiens je nevyhnutná tiež pri výpočte korelácie, pretože každá os je spracovaná nezávisle. Proces prezentuje zjednodušený kód 4.2.V sekciu medzi bariérami si úlohy predpočítajú smerodajnú odchýlku a zoznam rozdielov od aritmetického priemeru. Konštanta `task_mask` značí bitovými vlajkami, ktoré osi sú aktivované. Na základe $\text{axis} \in \{0, 1, 2\}$ signalizuje konkrétnie vlákno, že prišlo ku bariére. Mimo kritickej oblasti si jednotlivé vlákna, disponujúce medzivýsledkami za každú zložku vektora, dorátajú momentálne povolené kombinácie dvojíc súradníc individuálne.

4.3 Udalosti vo frekvenčnom spektre

ESP DSP knižnica optimalizuje pre naše účely transformáciu do frekvenčnej domény, algoritmami FFT s radixom 2 alebo DCT-II, a vyhľadenie signálu konvolúciou. Avšak dostupná implementácia kosínusovej transformácie nie je adekvátnej. Vyžaduje štvornásobnú veľkosť tabuľky koeficientov ku veľkosti transformácie a vnútorné volá generický algoritmus FFT. Prišlo k úprave zdrojového kódu knižnice, aby sa aspoň použila platforme prispôsobená verzia.

Pred oboma typmi transformácií sa vynásobia vzorky v posuvnom okne s pripravenými váhami oknovej funkcie (kód 4.3). Líši sa spôsob napĺňania vstupného poľa, kde u FFT tvoria reálne čísla na párnom a nepárnom mieste za sebou spoľočné komplexné číslo. DCT ponecháva následnosť reálnych vstupov, zato prázdná druhá polovica poľa zostáva na pracovné účely funkcie.

Poradie vedierok výsledného spektra FFT sa musí explicitne bitovo invertovať a previesť späť na striedanie reálnych a imaginárnych zložiek. Na záver sú zistené magnitúdy komplexných čísel. Prepočet na decibely berie za referenčnú úroveň frekvenciu s najväčšou intenzitou.

```

1 case DFT:
2     for (uint16_t i = 0; i < n; i++) {
3         spectrum[2*i+0] = buffer[i] * window[i];
4         spectrum[2*i+1] = 0;
5     }
6     dsps_fft2r_fc32_ae32(spectrum, n);
7     dsps_bit_rev2r_fc32(spectrum, n);
8     dsps_cplx2reC_fc32(spectrum, n);
9 case DCT:
10    for (uint16_t i = 0; i < n; i++)
11        spectrum[i] = buffer[i] * window[i];
12    dsps_dct_f32(spectrum, n);

```

Zdrojový kód 4.3: Fourierová a kosínusová transformácia s ESP DSP knižnicou

Stav detektora udalostí pozostáva z poľa štruktúr 4.4, ktoré odvádzajú časové značky počiatku, trvania a naposledy videnej špičky od počítadla prebehnutých posuvných okien. Upozornenie na zmeny vo frekvencii sa pri prechode prúdovým algoritmom poznačí do vymenovaného typu aktuálnej akcie `SpectrumEventAction`. Nadobúda konštanty z množiny „nič“, „start“ alebo „koniec“ a obnovujú sa na začiatku každého ďalšieho kola, na stav neprítomnosti akejkoľvek zmeny. Udalosti na

odoslanie sú sice pri serializácii správy lineárne prehľadávané, ale vytráca sa potreba udržiavať zásobník emitovaných udalostí.

```
1 typedef struct {
2     SpectrumEventAction action;
3     uint32_t start;
4     uint32_t duration;
5     int32_t last_seen;
6     float amplitude;
7 } SpectrumEvent;
```

Zdrojový kód 4.4: Štruktúra udalosti frekvenčného vedierka

4.4 Systémová konfigurácia

Správanie blokov dátovej pipeline určuje globálna inštancia zloženej štruktúry *Configuration* 4.5. Vnorené členské premenné definujú vlastnosti jednotlivých funkčných blokov. Po zapnutí je posledná konfigurácia zobrazená ako blob z nevolatilnej pamäte pod kľúčom „config“. Chýbajúca podpora iných ako celočíselných typov znamená, že po akejkoľvek úprave musí byť štruktúra znova nahratá do flash pamäte ako celok.

```
1 typedef struct {
2     SamplingConfig sensor;
3     SmoothingConfig tsmooth;
4     StatisticsConfig stats;
5     FFTTransformConfig transform;
6     SmoothingConfig fsmooth;
7     EventDetectionConfig peak;
8     SaveFormatConfig logger;
9 } Configuration;
```

Zdrojový kód 4.5: Štruktúra systémovej konfigurácie

Názvy, typy a prípustné hodnoty atribútov vo formáte Message Pack sú vyjadrené imperatívne priamo vo funkciách na konštruovanie a rozklad reťazca dátového obsahu paketu. Parsovanie prebieha v jednom prechode s MPack Expect API umožnením flexibilného poradia kľúčov, ktoré umožňuje priradiť modifikované pravidlo do kópie štruktúry bez dodatočného syntaktického stromu.

5 Overenie riešenia

Funkčnosť a efektivitu riešenia v súlade s kladenými požiadavkami overíme v rozličných scenároch. Zároveň experimentálne demonštrujeme odvodenie hyperparametrov klasifikácie špičiek s mriežkovým vyhľadávaním (grid search) a vyjadríme úspešnosť zaužívanými metrikami.

5.1 Pamäťová efektivita

Skompilovaný program senzorovej jednotky sa zmestí do pamäte inštrukcií so značnou rezervou. Kódový segment zaberá 64,42% alebo 81,8 kB dostupného priestoru vynímajúc vyhradené časti na vektory prerušení a vyrovnávacie pamäte procesora. Obsadením 43,4 kB v segmente .bss, prevažne na staticky alokované polia reťazcov odosielaných správ, a nárokovania si 14,7 kB konštánt, zostáva 82,3% DRAM na haldu. Spotrebu SRAM v bajtoch podľa segmentov objektového súboru podľa nástroja *GNU size* uvádzajú tabuľka 5.1.

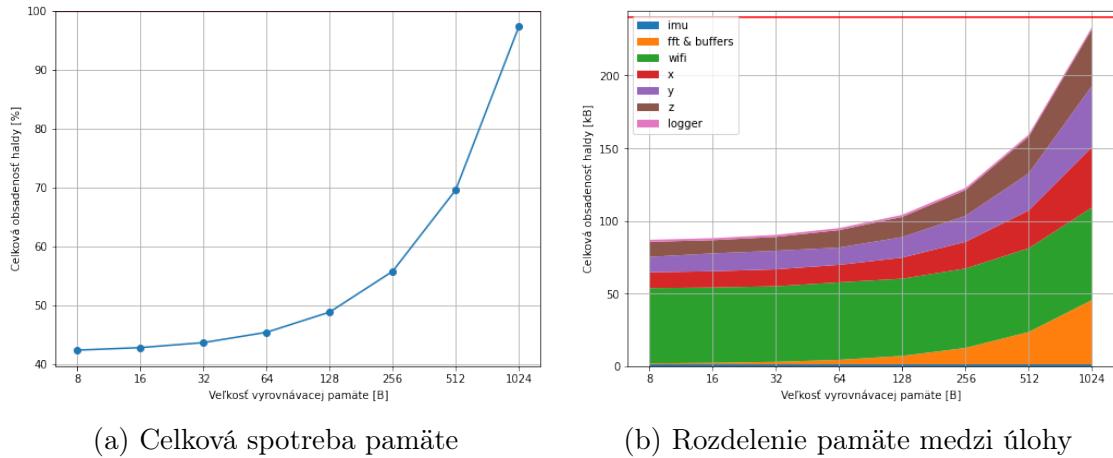
	IRAM (192 kB)				DRAM (328 kB)		
Sekcia	CPU cache	.vectors	.text	voľné	.bss	.data	voľné (heap)
Veľkosť	65536	1027	83780	46265	44392	15040	276440

Tabuľka 5.1: Rozdelenie pamäte v bajtoch medzi sekcie

Vyťaženie dynamickej pamäte z haldy lineárne závisí od počtu súčasne využívaných údajových bodov. Trend sa prejavuje v grafe celkovej percentuálnej naplnenosťi haldy 5.1a. Markantný stály podiel z voľného priestoru až okolo 55 kB sa poskytne na komunikáciu cez WiFi a na TCP/IP protokolový zásobník (graf 5.1b)

Zvyšok majú k dispozícii vlákna úloh na manipuláciu so vzorkami osí zrýchlenia (x, y, z), na zdieľané koeficienty FFT a konvolučné masky a relativne nepatrne si

obsadia úlohy vzorkovania (imu) a zápisu na pamäťovú kartu (logger). FreeRTOS je nastavený na dynamickú alokáciu zásobníkov, preto už pri veľkosti okna 8 potrebuje spracovateľská úloha 10 kB. Najdlhšie akceptovateľné posuvné okno a tým aj veľkosť frekvenčnej transformácie je 1024 bodov, ktoré vzhľadom na 97% spotreby haldy za istých okolností vykazuje nestabilitu systému. Odporuča sa pri zložitejšom procese úpravy signálu vystačiť si s 512 bodmi.



Obr. 5.1: Profilovanie dynamickej pamäte z haldy v DRAM

Veľkosť vyrovnávacej pamäte má priamy dopad na objem posielanej sieťovej premávky, ako vyčísluje tabuľka 5.2 v bajtoch. Sekvencia n meraní má za následok m frekvenčných vedierok s násobiacim faktorom veľkosti dátového typu. Nesúrodé štruktúry sú vyjadrené úhrnnou mierou informácie.

MQTT topic	Min. topic	Veľkosť v RAM	Hlavička (h)		Prvok (p)		Max. celková veľkosť
			Min.	Max.	Min.	Max.	
config/response	21	124	-	-	-	-	450
samples	13	$4 \cdot n$	1	3	5	5	$h + p \cdot n$
spectrum/+	16	$4 \cdot m$	14	22	5	5	$h + p \cdot m$
stats/+	13	52	4	8	9	12	127
events/+	14	$20 \cdot m$	18	26	17	27	$h + p \cdot m$

Tabuľka 5.2: Veľkosti Message Pack správ podľa MQTT topic

Používaný formát Message Pack máp sa zväčša skladá z hlavičky, čo je názov pre dvojice spoločne opisujúce variabilný počet obsiahnutých prvkov. Rozpätie v objeme hlavičky a položiek vyplýva z balenia menších hodnôt pod kratšiu binárnu reprezentáciu. Produkovaný obsah sa rozčleňuje na základe logických kategórií do

MQTT topics vkladané do hlavičky protokolu s dĺžkou vyjadrenou vrátane najkratšieho prefixu.

Aby sme detekciou udalostí dosiahli úsporu v množstve prenášaných údajov, je žiaduce dosiahnuť kratšiu správu ako zaslaním frekvenčných vedierok bez úpravy. Maximálna celková veľkosť dátovej nálože na tému *events* musí byť menšia ako na tému *spectrum*. Pri $m = 16$ to činí 2 udalosti na okno (12,5% z celkového počtu vedierok) a pri $m = 512$ sa musí vyskytnúť menej ako 93 udalostí na okno (18,16%). Datasetsy z autobusov vykazujú emisiu udalostí najviac do približne 6% z úhrnného počtu frekvencií a 0,5% v priemere. Štatistiky sa oplatí vytvárať pri najmenšom $n = 32$.

Protokol	Ethernet II	IPv4	TCP	MQTT	Spolu
Veľkosť hlavičky	14	20	20	>5	>55

Tabuľka 5.3: Rézia sieťového prenosu v bajtoch

Vysielané správy zapúzdrené v sieťových protokoloch nižších vrstiev OSI modelu pridávajú hlavičky na správne doručenie adresátovi, čím zvyšujú celkovú réziu. V lokálnej WiFi sieti, kde bola infraštruktúra prítomná, sa protokol MQTT pôsobiaci nad TCP, prenášal cez IPv4 v Ethernet-ových rámcoch. Každý paket má preto veľkosť vždy najmenej podľa tabuľky 5.3. Prenášaný obsah môže prevyšovať MTU, čo zapríčiní fragmentáciu do viacerých TCP segmentov a ďalší nárast nadbytku.

5.2 Časová efektivita

Naplnenie kritérií na rýchlosť odozvy odmeriame systémovým časovačom s mikrosekundovou presnosťou. Vplyv jednotlivých algoritmov na trvanie procesu úpravy signálu spriemerovaním 10 behov je patrný z tabuľky 5.4. Aktívna bola len úloha pre vybranú os zrýchlenia. S ohľadom na odchýlky najmä v dôsledku prerušení od vzorkovania a obsluhy plánovača operačného systému sa potrebný čas navyšuje priamo úmerne s dĺžkou postupnosti bodov.

Výpočet štatistik sa najvýraznejšie podieľa na predĺžovaní obratu spracovania rádovo v desiatkach milisekúnd, zatiaľ čo väčšina krokov prebehne aspoň 40-krát rýchlejšie. Nevyrovnanosť zapríčinujú miery vychádzajúce z mediánu, pretože sa

Veľkosť okna	32	64	128	256	512	1024
Štatistiky bez korelácií	3673	7471	14652	29574	59158	112871
DFT	80	162	306	611	1243	2620
DCT	91	165	310	612	1226	2532
Špičky: susedia	45	102	216	451	913	1812
Špičky: nulou do záporu	6	10	17	33	62	121
Špičky: horský turista	19	32	54	109	199	431
Udalosti	7	10	17	31	58	114

Tabuľka 5.4: Čas vykonávania algoritmov od veľkosti posuvného okna v μs pri taktovacej frekvencii 160 MHz a intervale plánovania 100 Hz

opakovane aplikuje Quickselect. Okrem toho si povšimneme, že v rýchlosťi vykonávania nie je žiadnen rozdiel medzi frekvenčnou transformáciou s FFT a DCT, z dôvodu spomenutých nedostatkov knižničnej implementácie DCT-II.

Najpomalším hľadaním špičiek je metóda najvyššieho spomedzi susedov, ktorá má najhoršiu asymptotickú časovú zložitosť spomedzi preberaných spôsobov. Dosahuje do 4-krát dlhšie časy ako algoritmus horského turista a do 14-krát oproti prechodu nulou do záporu. Výber prístupu ku klasifikácii špičiek nezáleží len od rýchlosťi, ale tiež od charakteru rozloženia vrcholov líšiaceho sa medzi algoritmami.

Vyhľadzovanie v časovej alebo frekvenčnej doméne sa vyznačuje meniteľnou dĺžkou konvolučnej masky a počtom opakovaných prechodov. Čas na dokončenie rovnako stúpa lineárne podľa oboch vlastností.

Veľkosť masky	4	16	64
1x	108	262	891
4x	413	1041	3697
8x	819	2065	7209

Tabuľka 5.5: Čas v μs na vyhľadzovanie v závislosti od veľkosti konvolučného jadra pri N = 512 a počtu opakovaní

Porovnávanie variant fáz spracovania separátne nezohľadňuje serializáciu a publikovanie správ zvolených tém, či dopad plánovania a synchronizácie na konkurentné úlohy. Pozrieme sa na výkonnosť dátovej pipeline v dvoch odlišných prípadoch pri frekvencii procesora 160 MHz a spriemerovaním desiatich spustení.

Správanie zariadenia v pokoji, bez odvodzovania akýchkoľvek štatistik, netvoriacie sieťovú premávku, približuje tabuľka 5.6a. Časy po pridaní výpočtu dostup-

ných štatistik vrátane korelácií a upozorňovanie na udalosti cez bezdrôtovú linku s RSSI na hladine cca -40 dBm popisuje tabuľka 5.6b. Vyhodnocuje sa trvanie behu v mikrosekundách vzhladom na veľkosť posuvného okna podľa algoritmu na hľadanie špičiek (číslovanie: A1, A2, A3, podľa kapitoly analýzy) pre aktivovaný počet rozmerov akcelerácie. U troch dimenzií sa zohľadní do priemeru úloha s najdlhším časom vykonávania. Frekvenčná transformácia je použitá za každej situácie FFT.

	N	32	256	1024		N	32	256	1024
1 os	A1	518	2616	6401	1 os	A1	14750	34883	129190
	A2	435	1598	6209		A2	8824	34351	139451
	A3	467	1695	3864		A3	13795	34890	137346
	A1	2503	3077	10177		A1	23851	101978	273696
	A2	556	3340	10334		A2	22981	78972	272161
	A3	591	1295	4670		A3	24232	100156	270110

(a) V pokoji bez posielania správ

(b) Štatistiky s koreláciami a udalostí

Tabuľka 5.6: Čas na spracovanie okna vzoriek v μs .

Hraničný čas t pokiaľ si vystačíme s tzv. „double buffering”, čiže sa neoneskorujeme od prúdu prichádzajúcich vzoriek o viac ako jednu dĺžku vyrovnávacej pamäte N nastáva, keď platí: $t \leq N/f_s$. Na určenie teoreticky najvyššej vzorkovacej frekvencie pri danej dĺžke N vezmeme časový parameter z tabuľky 5.6.

Lubovoľné nastavenie spracovania signálu zvládne za okolnosti podľa 5.6a $f_s = 61,8$ kHz pre jednorozmernú sekvenciu a $f_s = 12,8$ kHz pre trojrozumnú. Posielanie štatistik a udalostí z 5.6b pri jednej osi dovoľuje $f_s = 2,1$ kHz, ale pri väčších N sa f_s pohybuje nad 7 kHz. Tri dimenzie v tejto náročnej konfigurácii stíhajú nanajvýš f_s od 1,3 kHz ($N = 32$) do 3,7 kHz ($N = 1024$). Limit vzorkovacej frekvencie senzora na 952 Hz zaručuje, že bežné prevádzkové situácie sa stíhajú uskutočniť pred naplnením následného posuvného okna.

5.3 Úspešnosť detekcie špičiek

Syntetický signál so známymi časovo-frekvenčné spektrom sa zložil zo sinusoíd s exponenciálnym nábehom a dobehom. Očakávaná kvantita zlúčených tónov na základný úsek sa stanovila podľa frekvencie vzorkovania ovplyvňujúcej rozlišovaciu

schopnosť susediacich komponentov. Pri 238 Hz sa zmiešalo 8 rozdielnych frekven- cií, pri 476 Hz je ich 16 a pri 952 Hz sa ich rozmiestnilo 32. Pseudonáhodný generátor inicializovaný so semienkom 10 vytvoril trénovaciu množinu s trvaním 60 sekúnd a vzápäť 20 sekundovú testovaciu množinu.

Prehľadávaním parametrov detekcie špičiek hrubou silou nad trénovacím signálom boli odhalené najlepšie hodnoty z preddefinovanej sady. Na testovacom signále sa zistili relevantné metriky úspešnosti klasifikácie vrcholov makro-priemerovaním medzi posuvnými oknami o veľkosti n . Signál neboli dodatočne filtrovaný. Obdržanie spektrálneho obsahu v decibeloch prebehlo cez FFT s Hannovou oknovou funkciou a 50% prekryvom okien. Vyčíslené sú presnosti (tab. 5.7), senzitivita (tab. 5.8) a chybovosť (tab. 5.9) detekčných stratégií v percentách.

	Algoritmus 1			Algoritmus 2			Algoritmus 3		
$f_s \backslash n$	238	476	952	238	476	952	238	476	952
128	84.46	73.76	59.49	83.67	74.12	54.96	83.69	73.41	51.87
256	91.19	85.41	73.59	90.63	84.39	71.84	91.30	84.78	72.00
512	95.54	91.91	85.19	95.45	91.92	84.05	95.54	91.93	85.05

Tabuľka 5.7: Percentuálna presnosť klasifikácie frekvenčných špičiek

	Algoritmus 1			Algoritmus 2			Algoritmus 3		
$f_s \backslash n$	238	476	952	238	476	952	238	476	952
128	23.77	35.52	41.04	16.11	22.33	21.94	9.65	14.58	9.47
256	10.62	27.42	29.98	6.58	15.33	28.03	13.26	11.88	9.51
512	12.08	14.52	32.09	3.68	0.00	17.13	2.35	1.80	15.26

Tabuľka 5.8: Percentuálna senzitivita (TPR) klasifikácie frekvenčných špičiek

	Algoritmus 1			Algoritmus 2			Algoritmus 3		
$f_s \backslash n$	238	476	952	238	476	952	238	476	952
128	3.67	11.31	21.59	3.06	5.38	11.38	1.75	3.36	4.87
256	1.08	4.10	8.66	1.26	3.12	10.33	1.19	1.97	2.41
512	5.27	1.30	5.00	0.96	0.00	3.48	0.96	0.15	1.95

Tabuľka 5.9: Percentuálna chybovosť (FPR) klasifikácie frekvenčných špičiek

Platnosť získaných percentuálnych metrik v absolútnej škále a hyperparamet- rov sa vzťahuje výlučne na zvolenú techniku syntézy signálového priebehu, a teda

nevieme potvrdiť podobné výsledky pre reálnu prepravu. Napriek tomu môžeme z pravidelných tendencií dedukovať, že pomerne vysoká presnosť cca 84% zachováva júca senzitivitu a chybovosťou do 5% sa stabilne objavuje na diagonálach, kde n je polovicou f_s . Vtedy nachádzame ideálny balans medzi rozlíšením v čase a frekvencii aj v spektrogramoch.

Veľmi nízka senzitivita nezriedka 10 - 30% je dôsledkom nedokonalej spätej rekonštrukcie spektrálneho profilu a exaktnou lokalizáciou vrchola, čím sa sinusoidy môžu ocitnúť posunuté o pár frekvenčných vedierok vedľa v porovnaní s označením v datasete. Malý počet význačných frekvencií spôsobuje, že na prejavenie sa efektu zdanlivého poklesu senzitivity stačí, keď sa mierne zmení poloha jedinej.

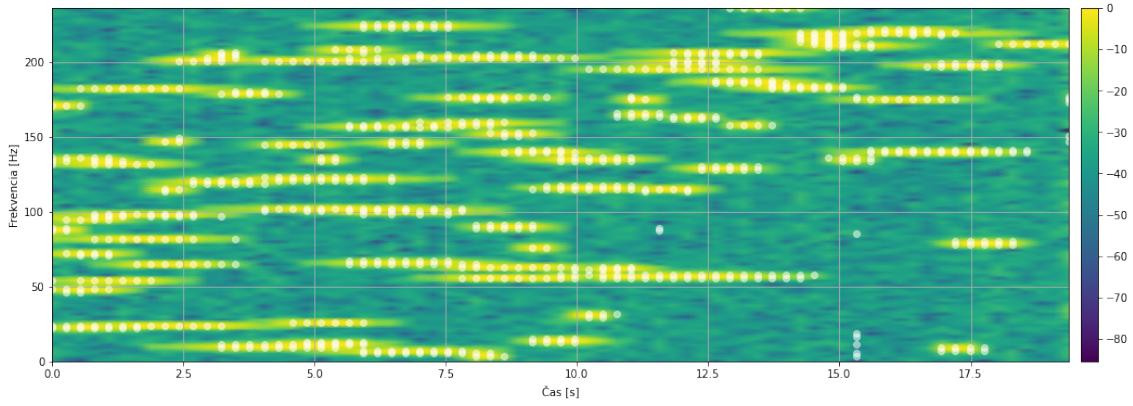
Spektrogram 5.2 znázorňuje na testovacích dátach schopnosť prúdového algoritmu na detekciu zmien frekvencií sa vysporiadať s nesprávne identifikovanými špičkami. Na vizualizácii je patrné, že registratie prvho vrchola na súvisom výstupku nastávalo oneskorene prekročením určitej amplitúdy. Ostatné algoritmy sa prejavovali obdobne, najvýraznejšie rozdiely sú v rozmiestnení falošných vrcholov.

Parametre klasifikácie špičiek nájdené mriežkovým hľadaním, s ktorými sme dosiahli na konkrétnom syntetickom signále najvyššie presnosti sú v tabuľke 5.10.

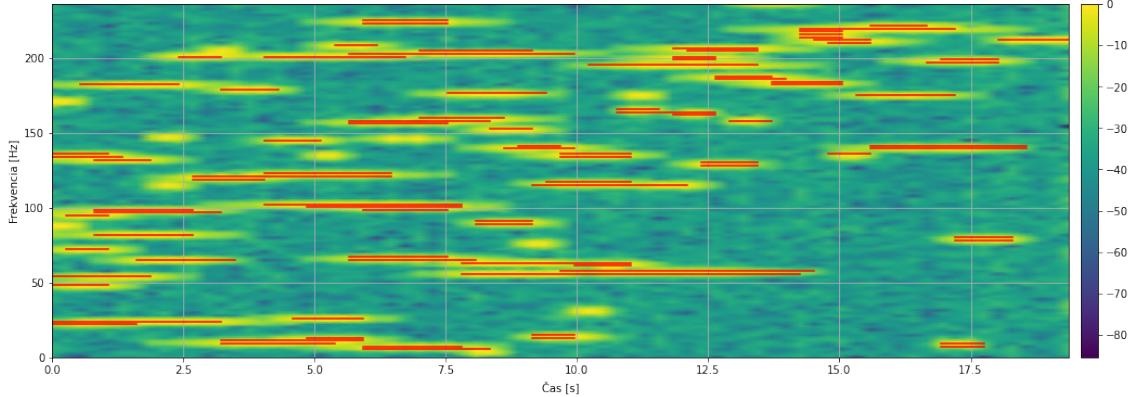
f_s	n	Algoritmus 1			Algoritmus 2		Algoritmus 2			
		k	ϵ	h_{rel}	k	s	t	h	p	i
238	128	12	3	32	2	12	16	0	10	8
238	256	3	1	32	2	16	16	0	10	12
238	512	3	4	32	2	26	10	4	38	0
476	128	3	4	16	2	9	10	0	10	0
476	256	12	4	32	2	12	16	0	10	0
476	512	3	4	32	1	20	10	4	33	0
952	128	3	4	0	2	5	10	0	10	0
952	256	6	4	24	2	5	16	0	10	0
952	512	6	4	32	2	12	16	0	10	0

Tabuľka 5.10: Parametre algoritmov detekcie špičiek na syntetických dátach (názvy sú skrátené na prvé písmená)

Manuálnym odladením parametrov na ukážkových záznamoch z vozidiel sme odskúšali rozdiely v schopnostiach algoritmov na hľadanie špičiek povšimnúť si pretrvávajúce harmonické zložky. Výrez časovo-premenného spektra na obr. 5.3 s nájdenými špičkami zvýrazňuje prekážky správneho odlíšenia šumu alebo navzájom splývajú-



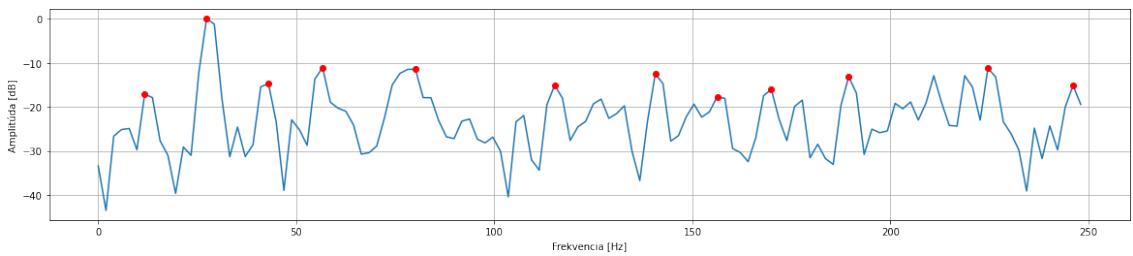
(a) Nájdené špičky algoritmom č.1 v posuvných oknách vyznačené bielym kruhom



(b) Zachytený priebeh udalostí frekvenčnej zmeny pri $t_{min} = 4$ a $t_{\Delta} = 1$

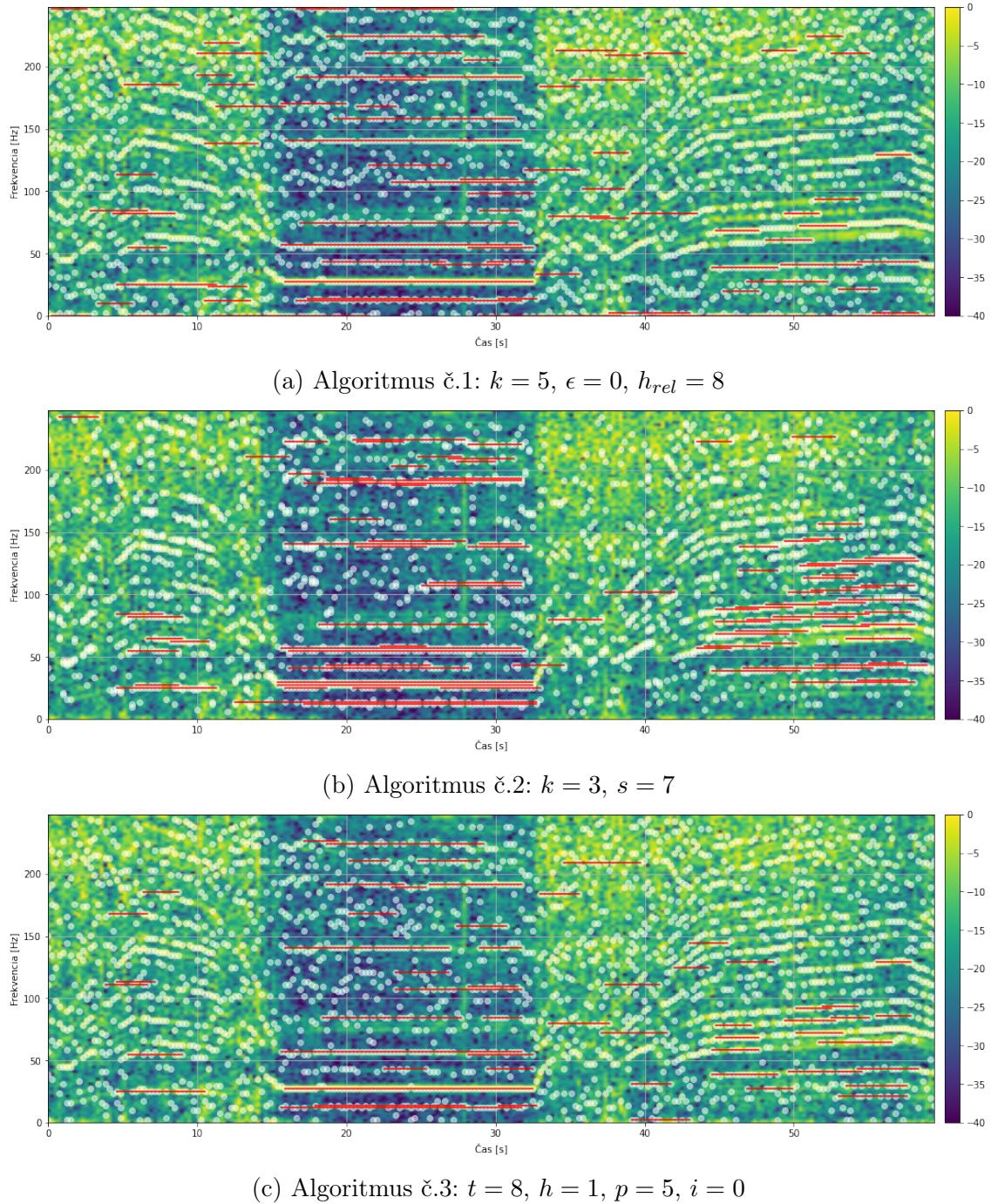
Obr. 5.2: Spektrogramy detegovaných špičiek a udalostí pri $f_s = 476$ Hz a $N = 256$

cich kopcov. Niektoré lokálne extrémy sú prehliadnuté pre nevýraznosť nad svojím okolím alebo pre prílišnú sploštenosť.



Obr. 5.3: Prierez spektrogramu okna 256 vzoriek s vrcholmi označenými algoritmom č.1 v 20. sekunde záznamu *L83_4940_Alexyho_Svantnerova.csv*

Nie je úplne zrejmé aké vodorovné konštelácie výrazných črt z obr. 5.4 sú korektné vyznačené, a ktoré sú primerane agregované do spoločných udalostí. Podstatné frekvencie v stabilnej oblasti medzi časmi 22 a 33 sekundou ako napr. 11, 26, 56 Hz boli univerzálne zaevidované. Detekcie z iného datasetu sú ilustrované v prílohe D.



Obr. 5.4: Detekcia udalostí v datasete *L83_4940_Alexyho_Svantnerova.csv* s $f_s = 500$ Hz, trvaním 60 s, dĺžkou okna 256, pri $t_{min} = 10$ a $t_\Delta = 4$

6 Zhodnotenie

Zaoberali sme sa meraním vibrácií snímačom trojrozmerného zrýchlenia. Signálových priebehy sme analyzovali viacerými existujúcimi metódami v časovej a frekvenčnej oblasti. Podstatou riešeného problému bolo usporiť bezdrôtovo prenášané množstvo informácií, poskytnutím prehľadu nad sledovanou situáciou upozornením na výskyt a amplitúdu významných frekvencií, či súhrnom časových úsekov deskriptívnymi štatistikami. Dôležitú etapu v automatizovanej extrakcii podstatných harmonických zložiek predstavovalo hľadanie špičiek, kde sme porovnali tri odlišné elementárne koncepcie v literatúre často uplatňované na biologické signály.

Prínos spočíva v navrhnutí postupnosti krovov spracovania dátovej pipeline, špecifikovaním modifikovateľných parametrov každého stupňa tejto sústavy, a implementácii vzdialene nastaviteľnej pipeline do firmvéru senzorovej jednotky. Hardvér zariadenia bol poskytnutý už zhotovený.

Na identifikáciu udalostí zmien spektrálneho obsahu vibrácií v čase sme vytvorili nový prúdový algoritmus. Poradí si s krátkodobými záchvevmi v označenej prítomnosti sekvencie vrcholov minimálnej dĺžky. Účinne pôsobí v podstate na redukcii šumu z náhlych jednorázových výskytov špičiek. Cez minimálnu testovaciu infraštruktúru zvolenými sieťovými protokolmi a serializačným formátom sme schopní posieláť tematicky kategorizované správy vlastnej štruktúry. Uznávame, že zotavenie firmvéru z chybových stavov spojené s nedostatkom prostriedkov sa mohlo hlásiť obšírnejšie ako reštartom alebo zaslaním všeobecnej hlášky.

Validácia detekčných schopností sa operala o originálny generátor syntetického signálu so šumom podľa predpisu požadovaných zložiek, ktoré sme na vyjadrenie úspešnosti detekcie pridávali pseudonáhodne. Stratégie spracovania sa konfrontovali s otrasmami v reálnej premávke datasetmi zozbieraných s dostupným zariadením.

Program obsadzujúci 65% voľnej pamäte na inštrukcie je schopný spoľahlivo

pracovať s posuvnými oknami do 512 bodov, v priamočiarejších scenároch kde sú výstupom len udalosti až do 1024 vzoriek. Posielanie všetkých štatistik sa oplatí nad postupnosť 32 bodov. Počet naraz oznamovaných zmien frekvencií by za zachovania efektívnosti nemal presiahnuť 12 - 18% vedierok posuvného okna. Skutočná priemerná prevalencia 0.5% a maximálna na úrovni 6% dokazuje ušetrenie v prenášanom obsahu. Systém splňa kladené obmedzenia na rýchlosť spracovania pri takto-vacej frekvencii procesora 160 Hz s ľubovoľnými nastaveniami. Vyrovnavacia pamäť dokončí obrat cez etapy pipeline pred skompletizovaním ďalšieho posuvného okna.

Vybudovaný model úspešne zakomponovaný na Edge IoT zariadenie tvorí dobrý základ pre početné rozšírenia vzťahujúce sa hlavne na detailnejšie preskúmanie vplyvov krokov frekvenčnej transformácie a filtrovania, a možnosti vyplývajúcich z viac-cestnej dátovej pipeline. Mohlo by sa jednať o využitie lepších kompresných vlastností energetických koeficientov od DCT-IV a MDCT, než FFT. S filtrovaním sa viaže upravenie údaju o tolerancii podľa aplikovaného kernelu vyhľadzovania.

Pokiaľ je predmetom záujmu konkrétny frekvenčný rozsah mohlo by sa autonómne stanoviť ich filtrovanie a najlepšie rozlíšenie. Rovnako tak sa pre potenciálne produkčné účely ukazuje nepostrádateľnosť samostatnej kalibrácie parametrov hľadania špičiek. Zaujímavé by bolo prísť s ošetrením cyklického frekvenčného driftu a umožniť označenie profilu známych javov s ich odlišením pri notifikáciach. Ďalšia pridaná hodnota by spočívala v určení prevažného priestorového smeru frekvencie alebo koordinácií viacerých senzorov v spoločnom transportnom boxe.

Dosiaľ dosiahnuté výsledky môžu byť obohatené o silnejšie závery úspešnosti detekcie z opakovaného snímania rôznych stavov vozidla na vybraných cestách za vypracovania metodiky anotovania skompilovaného datasetu. Za kontrolovanejších podmienok by sa zariadenie mohlo podrobiť skúšaniu na testovacej lavici.

Softvérové riešenie vychádzajúce z mnohostrannej analýzy problematiky je funkčné a napĺňa intencie zadania tejto bakalárskej práce.

Literatúra

1. BROCH, Jens Trampe. *Mechanical Vibration and Shock Measurements*. 2. vyd. Brüel & Kjær, 1984. ISBN 87-87355-34-5.
2. MOHAMMED, Zakriya; ELFADEL, Ibrahim (Abe) M.; RASRAS, Mahmoud. Monolithic Multi Degree of Freedom (MDoF) Capacitive MEMS Accelerometers. *Micromachines*. 2018, roč. 9, č. 11. ISSN 2072-666X. Dostupné z DOI: 10.3390/mi9110602.
3. TSAI, Ming-Han; LIU, Yu-Chia; SUN, Chih-Ming; WANG, Chuanwei; CHENG, Chun-Wen; FANG, Weileun. A $400 \times 400 \mu\text{m}^2$ 3-axis CMOS-MEMS accelerometer with vertically integrated fully-differential sensing electrodes. In: *2011 16th International Solid-State Sensors, Actuators and Microsystems Conference*. 2011, s. 811–814. Dostupné z DOI: 10.1109/TRANSDUCERS.2011.5969154.
4. DADAFSHAR, Majid. *Accelerometer and Gyroscopes Sensors: Operation, Sensing, and Applications*. 2014.
5. MÜLLER, Meinard. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. 1st. Springer, Inc., 2015. ISBN 978-3-319-21945-5. Dostupné z DOI: 10.1007/978-3-319-21945-5.
6. *iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer*. 2015. Č. LSM9DS1. Rev. 3.
7. *Micropower, 3-Axis, $\pm 2 \text{ g}/\pm 4 \text{ g}/\pm 8 \text{ g}$ Digital Output MEMS Accelerometer*. 2019. Č. ADXL362. Rev. F.
8. *Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers*. 2020. Č. ADXL356. Rev. A.

9. QUARTERONI, Alfio; SACCO, Riccardo; SALERI, Fausto. Numerical Mathematics. In: Springer Inc., 2000, kap. 9. Numerical Integration, s. 371–398. ISBN 0-387-98959-5.
10. YANG, Yanli; ZHAO, Yanfei; KANG, Dali. Integration on acceleration signals by adjusting with envelopes. *Journal of Measurements in Engineering*, 2016, roč. 4, s. 117–121.
11. MUTHUKRISHNAN, S. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*. 2005, roč. 1, č. 2, s. 117–236. ISSN 1551-305X. Dostupné z DOI: 10.1561/0400000002.
12. PAJUREK, Tomáš. *Online Anomaly Detection in Time-Series*. Fakulta informačních technologií, České vysoké učení technické v Praze, 2018. Dostupné tiež z: <https://dspace.cvut.cz/bitstream/handle/10467/76417/F8-DP-2018-Pajurek-Tomas-thesis.pdf>. Dipl. pr.
13. NIELSEN, Aileen. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media, 2019. ISBN 978-1-4920-4165-8.
14. NEUBAUER, Jiří; SEDLÁČEK, Marek; KRÍŽ, Oldřich. *Základy statistiky - Aplikace v technických a ekonomických oborech*. 1. vyd. Grada Publishing, a.s., 2012. ISBN 978-80-247-4273-1.
15. KNUTH, Donald E. The Art of Computer Programming. In: 2. vyd. Addison-Wesley, 1981, zv. 2, kap. 4.2.2, s. 216. ISBN 0-201-03822-6.
16. SCHNEIDER, Roger. Survey of Peaks/Valleys identification in Time Series. 2011. Department of Informatics, University of Zürich.
17. YANG, Chao; HE, Zengyou; YU, Weichuan. Comparison of public peak detection algorithms for MALDI mass spectrometry data analysis. *BMC bioinformatics*. 2009, roč. 10, s. 4. Dostupné z DOI: 10.1186/1471-2105-10-4.
18. O'HAVER, Thomas. A Pragmatic Introduction to Signal Processing. In: 2020, kap. Peak Finding and Measurement. ISBN 979-86-11-26668-7. Dostupné tiež z: <https://terpconnect.umd.edu/~toh/spectrum/>.

19. ARGÜELLO-PRADA, Erick Javier. The mountaineer's method for peak detection in photoplethysmographic signals. *Revista Facultad de Ingeniería Universidad de Antioquia*. 2019, č. 90, s. 42–50. Dostupné z DOI: 10.17533/udea.redin.n90a06.
20. RASCHKA, Sebastian. An Overview of General Performance Metrics of Binary Classifier Systems. *CoRR*. 2014, roč. abs/1410.5330. Dostupné z arXiv: 1410 . 5330.
21. FAWCETT, Tom. An introduction to ROC analysis. *Pattern Recognition Letters*. 2006, roč. 27, č. 8, s. 861–874. ISSN 0167-8655. Dostupné z DOI: <https://doi.org/10.1016/j.patrec.2005.10.010>. ROC Analysis in Pattern Recognition.
22. LYONS, Richard G. *Understanding Digital Signal Processing*. 3. vyd. Pearson Education, Inc., 2011. ISBN 978-0-13-702741-5.
23. PRANDONI, Paolo; VETTERLI, Martin. *Signal Processing for Communications*. EPFL Press, 2008. ISBN 978-2-940222-20-9.
24. KHAYAM, Syed Ali. The Discrete Cosine Transform (DCT): Theory and Application. *Course Notes, Department of Electrical & Computer Engineering*. 2003.
25. STRANG, Gilbert. The Discrete Cosine Transform. *SIAM Rev.* 1999, roč. 41, č. 1, s. 135–147. ISSN 0036-1445. Dostupné z DOI: 10.1137/S0036144598336745.
26. SHAO, Xuancheng; JOHNSON, Steven G. Type-IV DCT, DST, and MDCT algorithms with reduced numbers of arithmetic operations. *CoRR*. 2007. Dostupné tiež z: <http://arxiv.org/abs/0708.4399>.
27. CHU, Eleanor; GEORGE, Alan. *Inside the FFT Blackbox: Serial and Parallel Fast Fourier Transform Algorithms*. CRC Press LLC, 2000. Computational Mathematics. ISBN 0-8493-0270-6.
28. KUMAR, G Ganesh; SAHOO, Subhendu; MEHER, P.K. 50 Years of FFT Algorithms and Applications. *Circuits, Systems, and Signal Processing*. 2019, roč. 38. Dostupné z DOI: 10.1007/s00034-019-01136-8.

29. DUHAMEL, P.; HOLLMANN, Henk. ‘Split radix’ FFT algorithm. *Electronics Letters*. 1984, roč. 20, s. 14–16. Dostupné z DOI: 10.1049/el:19840012.
30. MAKHOUL, J. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1980, roč. 28, č. 1, s. 27–34. Dostupné z DOI: 10.1109/TASSP.1980.1163351.
31. HEINZEL, G.; RÜDIGER, A.; SCHILLING, R. Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new at-top windows. In: 2002.
32. WELCH, P. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*. 1967, roč. 15, č. 2, s. 70–73. Dostupné z DOI: 10.1109/TAU.1967.1161901.
33. MATIN, M.A.; ISLAM, M.M. Overview of Wireless Sensor Network. In: MATIN, Mohammad A. (ed.). *Wireless Sensor Networks*. IntechOpen, 2012, kap. 1, s. 1–22. ISBN 978-953-51-0735-4. Dostupné z DOI: 10.5772/49376.
34. Vykonávacie rozhodnutie Komisie (EÚ) 2019/1345 z 2. augusta 2019, ktorým sa mení rozhodnutie 2006/771/ES s cieľom aktualizovať harmonizované technické podmienky v oblasti využívania rádiového frekvenčného spektra pre zariadenia s krátkym dosahom. *OJ*. 2019-08-13, roč. L 212, s. 53–72.
35. *Bluetooth Technology Website / The Official Website of Bluetooth Technology* [online] [cit. 2021-11-15]. Dostupné z : <https://www.bluetooth.com/>.
36. DJEDOUBOUM, Asside; ARI, Ado; GUEROUI, Abdelhak; MOHAMADOU, Alidou; ALIOUAT, Zibouda. Big Data Collection in Large-Scale Wireless Sensor Networks. *Sensors*. 2018, roč. 18. Dostupné z DOI: 10.3390/s18124474.
37. IYENGAR, Ashok; OUYANG, Christine. *Edge computing architecture* [online] [cit. 2021-11-15]. Dostupné z : <https://www.ibm.com/cloud/architecture/architectures/edge-computing/>.
38. MAGESH, S.; INDUMATHI, J.; S, Radha RamMohan.; R, Niveditha V.; PRABHA, P. Shanmuga. Concepts and Contributions of Edge Computing in Internet of Things (IoT): A Survey. In: 2020, zv. 7, s. 146–156. Č. 5. Dostupné z DOI: 10.22247/ijcna/2020/203914.

Príloha A: Plán práce

A.1 Zimný semester

Obdobie	Náplň práce
1. týždeň	Základný prehľad relevantnej literatúry.
2. týždeň	Štúdium literatúry ohľadom montorovania 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 autobusu.
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 dátovej pipeline.
4. týždeň	Zakomponovanie posielania vzoriek cez MQTT. Validácia na syntetických dátach rozdelených na trénovaciu a testovaciu sadu.
5. týždeň	Implementácia kostry dátovej pipeline na IoT zariadenie. Message Pack serializácia konfigurácie a jej publikovanie cez MQTT.
6. týždeň	Parser priatej konfigurácie, uloženie a aplikácia nastavení na zariadení. Optimalizácia alokowania 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 kompletnejšia daná 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 mala byť funkčný záznamom 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 vyhodnotenie, 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`
- **Dátová pipeline** (B.1.5) - Fázy spracovania zdrojového signálu. Zdrojový kód: `pipeline.h`
 - **Oknové funkcie** - `window.c`
 - **Správa pamäti pipeline** - `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 akcelometra 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 Dátová pipeline

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 pipeline 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 celú pipeline.

struct BufferPipelineAxis - Vyrovnávacie pamäte samostatné pre každú os akcelerácie.

struct Sender - Fronta 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 vo fronte (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 vyhľadzovacieho filtra (8)

- LARGEST_MESSAGE: Najväčšia veľkosť vyrovnávacej 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 pipeline

```
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 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 pamäte pre dynamické vyrovnávacie pamäte.

Parametre:

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

```
void axis_release (BufferPipelineAxis *p)
```

Uvoľnenie pamäte pre dynamické vyrovnávacie pamäte pre jednu os akcelerácie.

Parametre:

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

```
void sender_release (Sender *sender)
```

Odstránenie fronty na odosielanie nameraných vzoriek.

Parametre

- **sender**: Fronta s vyhradenou kapacitou
-

Fázy spracovania oknovaného signálu

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

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

Parametre:

- **buffer**: Vyrovnávacia pamäť, ktorej obsah bude posunutý
 - **n** (in): Dĺžka vyrovnávacej 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 povolené 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
- **tmp**: Pomocné pole o dĺžke n + c.n - 1
- **window** (in): Váhy oknovej funkcie s dĺžkou n
- **n** (in): Dĺžka posuvného okna
- **kernel** (in): Konvolučná maska vyhľadzovania

- **c** (in): Nastavenia vyhľadzovanie
-

```
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
- **n** (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
)
```

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 parsovani
 - **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 parsovani
-

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:

- 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)
```

Špicatost'.

Parametre:

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

```
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 tématicky odlíšený 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 poohľadu externého klienta, ktorý má záujem vykonávať zber údajov zo senzorovej jednotky. Štatistiky, frekvenčné spektrum a udalosti sa produkujú pre viaceré priesitorové 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*”: Prihlásovacie 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 sieťového 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 povolojujú. 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 sú 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 sieťovom pripojení s prázdnym obsahom. Odpoveď sa očakáva na tému *login/response*.
- **login/response** (*Subscribe*) - Aktuálne sieťové pripojenie: WiFi SSID príspuvkového bodu, heslo a URL adresa na MQTT broker. Neodporúča sa ponechá-

vať heslo zahrnuté v tomto zobrazení, ale napomáha sa tým testovaniu.

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

B.3 Datasety z premávky

Datasety z autobusov a električiek boli zaznamenané v dátumoch 1.11. a 3.11.2021 so vzorkovacou frekvenciou 500 Hz a rozlíšením ± 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

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

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, hned' za pravotočivou zákrutou križovatky na smer Lamač)
- **Vozidlo:** midibus Solaris Urbino 8,6 #1915
- **Umiestnenie:** pravá zadná náprava, predposledné zadné sedenie proti smeru jazdy

L20_3014_Zaluhy_Drobneho.csv

- **Linka:** 20
- **Trvanie:** 113001 vzoriek (226 s)

PRÍLOHA B. TECHNICKÁ DOKUMENTÁCIA

- **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

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 zvažujúcom kopci)
- **Vozidlo:** klíbový autobus Mercedes-Benz O 530 GL CapaCity #4940
- **Umiestnenie:** nad motorom vpravo vzadu, posledné zadné priečne sedadlo pred plošinou na batožinu

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

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 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 pipeline na syntetickom signále sa nachádza v Jupyter notebookoch, ktoré sa otvárajú z priečinku measurements príkazom:

```
$ jupyter notebook
```

Experiment: Časová efektivita algoritmov na spracovanie signálu

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 na spracovanie signálu

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 komplilácie: 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 dátovej pipeline

Direktíva podmienenej komplilácie: 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"}}

```

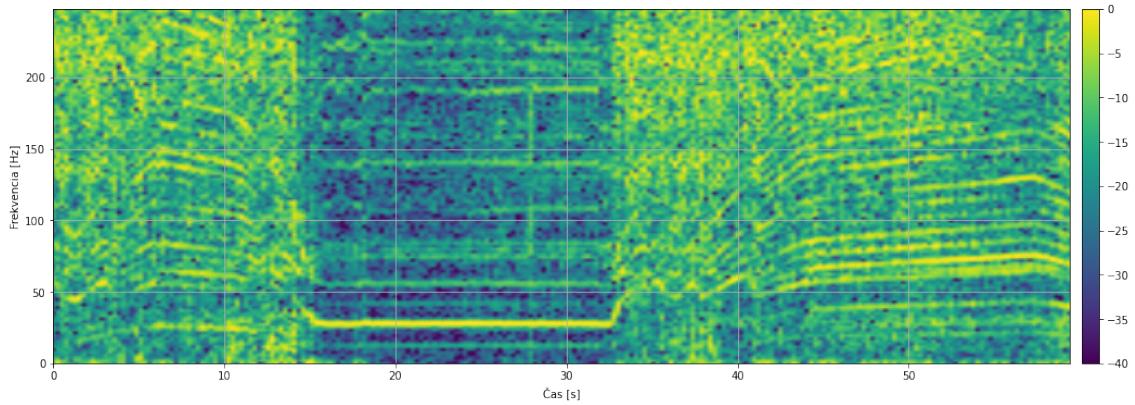
Výpisy zo separátnych experimentov A, B, C, D sú prehľadané na výskyt riadkov s časmi a odtiaľ manuálne upravené do finálnej podoby

```

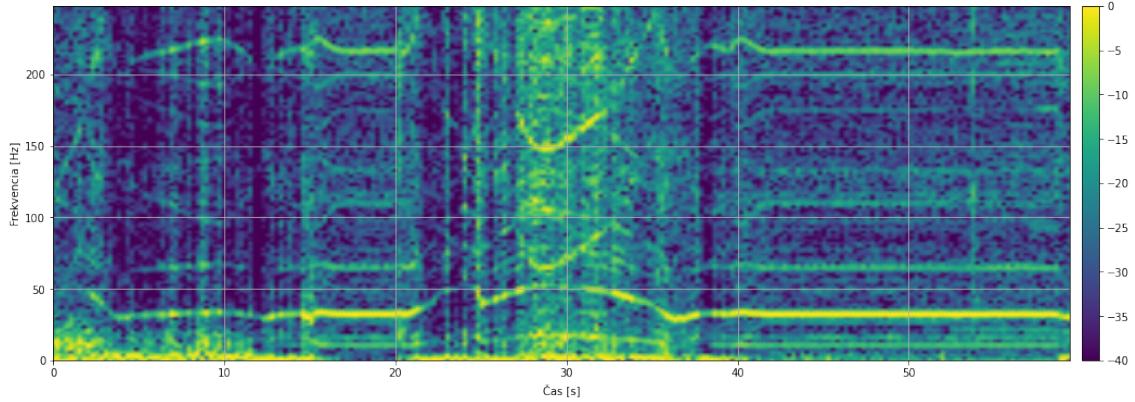
sed -rn '/^(.*) (main:.*$ /p' _pipeline_time-X.txt
>_pipeline_time-X_filter.csv

```


Príloha D: Spektrogramy

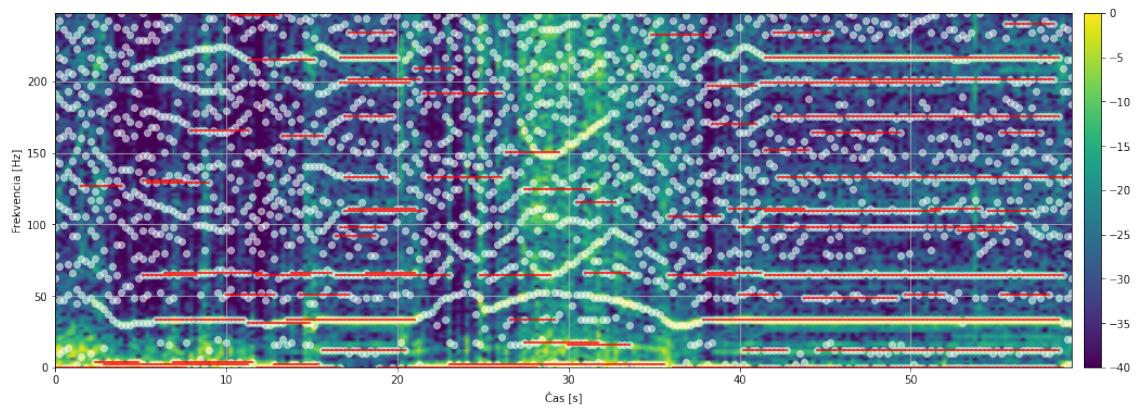


Obr. D.1: Spektrogram záznamu z vozidla *L83_4940_Alexyho_Svantnerova.csv*

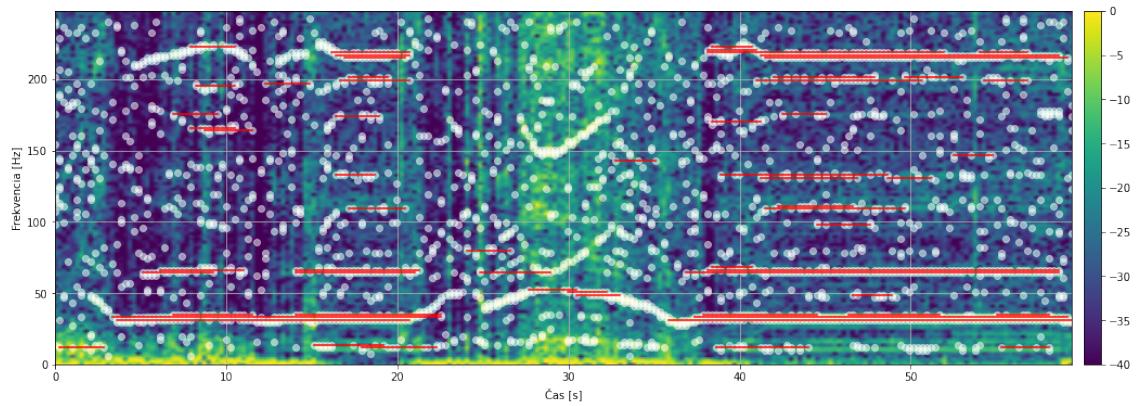


Obr. D.2: Spektrogram záznamu z vozidla *L35_1915_Borska_Zaluhy.csv*

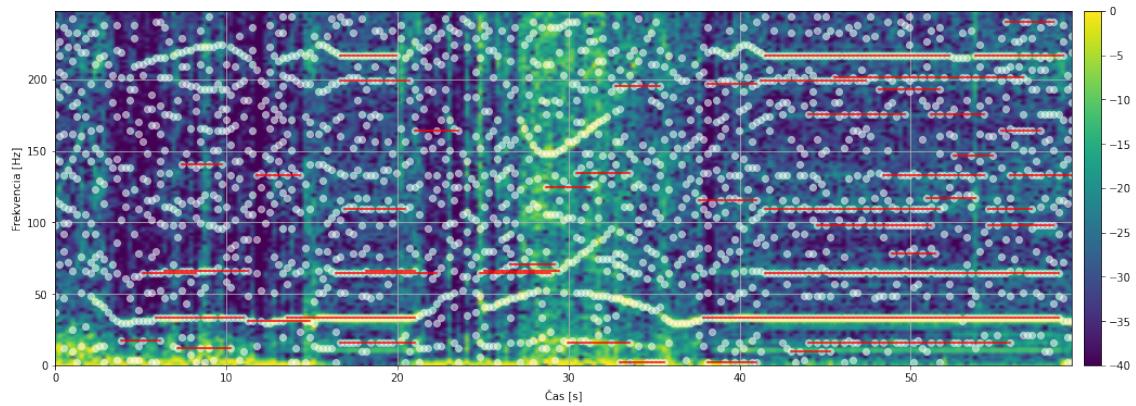
Detekcia udalostí pri vzorkovacej frekvencii 500 Hz, dĺžkou FFT 256 bodov, $t_{min} = 10$ a $t_{\Delta} = 4$ na datasete *L35_1915_Borska_Zaluhy.csv*:



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 celej dátovej pipeline.
 - > **memory-usage** - Spotreba flash pamäte.
 - > **network** - Odchytená sieťová komunikácie z MQTT broker v *.pcap* súboroch.
- > **signals** - Užitočné funkcie ku prieskumnej analýze, vizualizácii a tvorbe modelov v Jupyter notebookoch.