VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ELEKTRONICKÝ MODUL PRO AKUSTICKOU DETEKCI
ELECTRONIC MODULE FOR ACOUSTIC DETECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Martin Maršál

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Zdeněk Havránek, Ph.D.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Martin Maršál ***ID:*** 146055
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Elektronický modul pro akustickou detekci

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a realizace akustického detekčního modulu pro zabezpečovací účely. Zadání lze shrnout do následujících bodů:
1) Zpracujte teorii metod pro analýzu, identifikaci a klasifikaci různých druhů zvuků včetně rozboru způsobu volby vhodných příznaků a tvorby modelu pro strojové učení.
2) Navrhněte modul pro detekci vybraných zvuků s MEMS mikrofonem, mikrokontrolerem a rozhraním Ethernet. Popište zvolené HW komponenty a použité metody pro digitální zpracování signálů v modulu.
3) Popište možnost zapojení více modulů do distribuované sítě tak, aby nadřazený systém byl schopen identifikovat i polohu zdroje zvuku pomocí analýzy zpoždění příchodu akustických signálu do jednotlivých modulů s využitím přesné časové synchronizace modulů po síti.
4) Realizujte navržený modul včetně firmware pro mikrokontroler. Hardwarová realizace modulu může využívat vhodný komerčně dostupný vývojový modul.
5) Ověřte správnost dat vysílaných pro síti a spolehlivost identifikace a klasifikace zdroje zvuku.

DOPORUČENÁ LITERATURA:

[1] Mohri, M., Rostamiyadeh, A., Talwalkar, A. Foundations of Machine Learning. MIT Press, 2012. 414 p. ISBN 978-0262018258.

[2] Thampi, S. M., Gelbukh, A., Mukhopadhyay, J. Advances in Signal Processing and Intelligent Recognition Systems. Springer, 2014. 612 p. ISBN 978-3319049595.

Termín zadání: 8.2.2016 ***Termín odevzdání:*** 16.5.2016

Vedoucí práce: Ing. Zdeněk Havránek, Ph.D.
Konzultant diplomové práce:

doc. Ing. Václav Jirsík, CSc., *předseda oborové rady*

ABSTRAKT

Tato diplomová práce se zabývá návrhem a realizací elektronického modulu pro akustickou detekci. Modul má za úkol detekovat předem definované akustické signály pomocí na ně naučeného klasifikačního modelu. Modul slouží převážně pro zabezpečovací účely. Pro identifikaci a klasifikaci je navržen model pomocí technik strojového učení. Vzhledem k možnosti přeučení na jinou sadu zvuků se modul stává univerzálním akustickým detektorem. Pro snímání akustického zvuku je použit digitální MEMS mikrofon, pro který je navržen a realizován převodní filtr. Výsledný systém je implementován do firmwaru mikrokontroléru s operačním systémem reálného času. Jednotlivé funkce systému jsou realizovány s ohledem na možnou optimalizaci (méně výkonný MCU nebo bateriové napájení). Modul předává výsledky detekce nadřazené stanici pomocí Ethernetové sítě. V případě více modulů připojených do sítě se vytvoří distribuovaný systém, pro který je navržena přesná časová synchronizace pomocí PTP protokolu definovaného normou IEEE-1588.

KLÍČOVÁ SLOVA

rozpoznávání akustického signálu, klasifikace akustického signálu, identifikace akustického signálu, detekce akustického signálu, Goertzelův algoritmus, MEMS mikrofon, strojové učení, PTP protokol, FRDM-K64F, I2S interface, DMA, PDM signál, CIC filtr, Naive Bayes

ABSTRACT

This diploma thesis deals with the design and implementation of an electronic module for acoustic detection. The module has the task of detecting a predetermined acoustic signals through them learned classification model. The module is used mainly for security purposes. To identify and classify the proposed model using machine learning techniques. Given the possibility of retraining for a different set of sounds, the module becomes a universal sound detector. With acoustic sound using the digital MEMS microphone, for which it is designed and implemented conversion filter. The resulting system is implemented into firmware microcontroller with real time operating system. The various functions of the system are realized with regard to the possible optimization (less powerful MCU or battery power). The module transmits the detection results of the master station via Ethernet network. In the case of multiple modules connected to the network to create a distributed system, which is designed for precise time synchronization using PTP protocol defined by the IEEE-1588 standard.

KEYWORDS

acoustic signal recognizing, acoustic signal classification, acoustic signal identifying, acoustic signal detection, Goertzel algorithm, MEMS microphone, Machine Learning, PTP protocol, FRDM-K64F, I2S interface, DMA, PDM signal, CIC filter, Naive Bayes

MARŠÁL, M. *Elektronický modul pro akustickou detekci*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 87 s. Vedoucí diplomové práce Ing. Zdeněk Havránek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Elektronický modul pro akustickou detekci“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno …….. ……

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu diplomové práce panu Ing. Zdeňkovi Havránkovi, Ph.D. za spolupráci, čas strávený na konzultacích a odborné rady při řešení mé diplomové práce.

Brno …….. ……
(podpis autora)

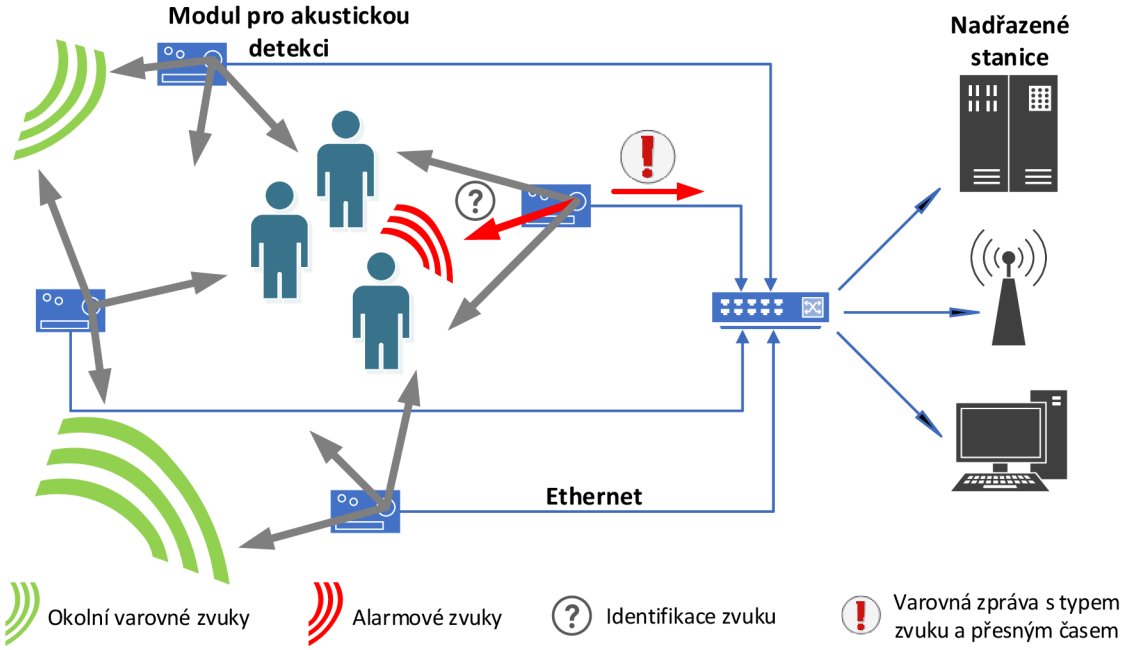
OBSAH

Úvod

Komunikace pomocí zvukových signálů je jedním z nejrozšířenějších stylů lidské komunikace. Smysl, kterým člověk vnímá okolí je velice důležitý a dokážeme díky němu rozpoznávat a poznávat mnohé dění kolem nás i bez použití ostatních smyslů. To je hlavní myšlenka k vytvoření této práce, která by svým výsledkem měla přispět veřejnému společenství. V dnešní době se s moderními technologiemi setkáváme ve všech odvětvích našeho žití a potkáváme se s nimi opravdu každý den. Proč tedy nevytvořit umělý smysl, který alespoň z části vnímá zvukové podměty a dokáže na ně reagovat? Díky takovému umělému smyslu, systému by se mohlo zamezit mnohým nebezpečím nebo alespoň včas varovat. Systém by tedy mohl mít význam v mnoha odvětvích běžného života například zabezpečení majetku či hlídání osob.

1 ROZVRŽENÍ PRÁCE

Cílem mé práce je vytvoření multifunkčního systému pro detekci akustického signálu a návrh řešení následné lokalizace jeho zdroje. Výsledkem bude elektronický modul pro detekci a lokalizaci akustického signálu v prostředí. Ideové schéma vlastností je zobrazeno na Obr. 1.

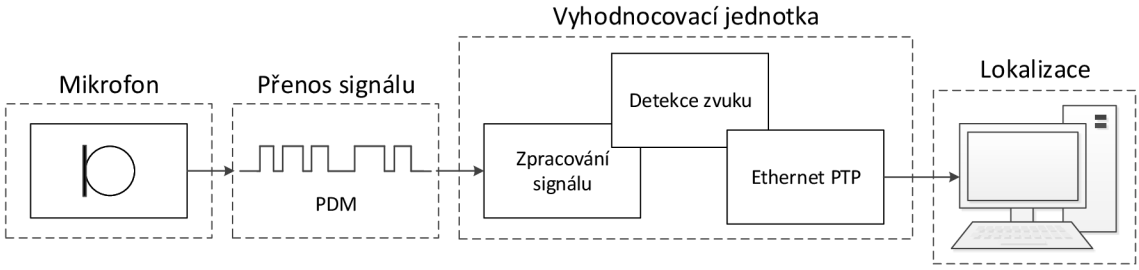


Obr. 1: Ideové schéma funkcí systému

Vlastnosti systému lze tedy rozdělit do dvou oblastí. První oblast je detekční, zde má systém za úkol rozpoznávat zvuky z prostředí a následně na ně adekvátně reagovat, například přivolání pomoci. Systém je navržen především na varovné zvuky - alarmové zvuky, nepřirozené rány (vloupání), rozbití skel a únik plynu. Systém by mohl být interaktivně doučován na okolní podněty, ve smyslu sledování aktivity osob v pokročilém věku. Vzhledem k tomu, že se bude detekční část skládat z předem naučeného modelu a bude možné tento model přeučit na jinou trénovací množinu a tak změnit detekční subjekty. Druhá oblast je lokalizační, zde systém poskytne potřebné údaje pro následnou lokalizaci zdroje signálu. Pro tuto vlastnost je ale zapotřebí zapojení více senzorů v jedné síti a vznikne tzv. distribuovaný systém. Počet senzorů v síti je volitelný, ale počet odpovídá směrovosti počtu směrů, který je schopen rozpoznávat. Proto je pro základní stranovou lokalizaci potřeba, aby v síti byly připojeny minimálně dva senzory.

Z technologického hlediska lze systém rozdělit na čtyři hlavní části, viz blokové zobrazení systému Obr. 2. První část je snímání akustického zvuku z prostředí. K tomu slouží mikrofon. Pro jeho výběr provedu podrobnou rešerši dostupných mikrofonů na našem trhu. Druhá část je přenos dat mezi mikrofonem a vyhodnocovacím prvkem a následné převedení signálu. Tato část je závislá na typu výstupního signálu z mikrofonu, podle kterého se bude dále postupovat. Třetí část je výběr samotné vyhodnocovací

jednotky, která musí být dostatečně výkonná a zahrnovat vhodné periferie pro potřeby systému a zároveň realizovatelná ve vyhodnocovací jednotce. Zde provedu průzkum trhu a podle požadovaných kritérií vyberu nejvhodnějšího kandidáta. Ve vyhodnocovací jednotce se budou provádět úkony týkající se oboustranné komunikace a především samotné detekce. Poslední, čtvrtou částí je připojení systému k vnějšímu světu, které bude zajišťovat ethernetové rozhraní. Zmiňovaná vlastnost lokalizace zdroje zvuku bude realizována pomocí přesných časových značek. Aby byl čas opravdu přesný, je nutné všechny zařízení v síti pravidelně synchronizovat. Pro synchronizaci použiji standardu IEEE-1588 označovaného PTP (*Precision Time Protocol*). Díky tomu lze dosáhnout údajů pro velice přesnou lokalizaci zdroje akustického signálu.



Obr. 2: Blokové schéma systému

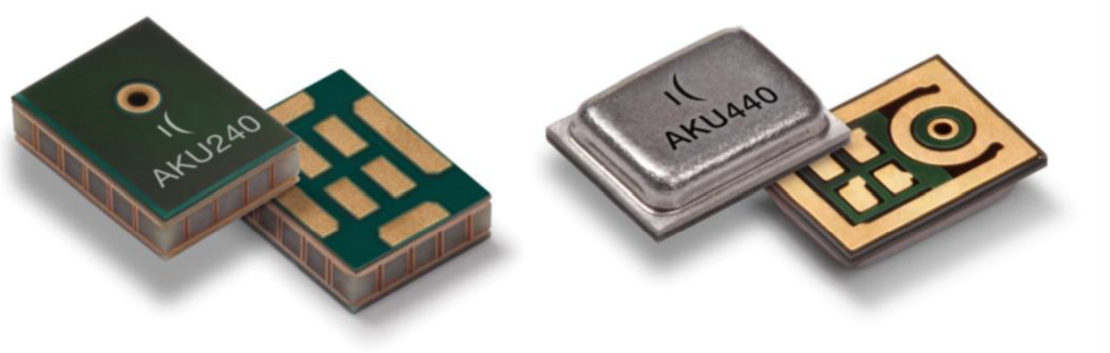
Tyto části shrnují strukturu tohoto dokumentu a v následujícím popisu se jim podrobně věnuji a provádím jejich teoretický rozbor. V dalších kapitolách dokumentu je popsán samotný realizační postup navazující a vycházející z těchto znalostí.

2 MIKROFONY

Mikrofon je v mém systému využíván pro snímání akustického signálu z prostředí. Na obchodním trhu je dostupné velké množství různých typů mikrofonů, které se liší nejen konstrukcí, ale především vlastnostmi přenášeného akustického tlaku na elektrický signál. Proto jsem před samotnou realizací systému nejprve provedl rešerši dostupných druhů mikrofonů, abych našel mikrofon s vyhovujícími vlastnostmi pro můj systém. V této rešerši jsem kvůli praktičnosti použití upřednostnil MEMS mikrofony, kterým tím pádem věnuji dále větší prostor. U ostatních druhů mikrofonů proto vysvětluji pouze stručně základní principy.

2.1 MEMS kapacitní mikrofony

MEMS (Mikro-Elektro-Mechanický Sytém) kapacitní mikrofon (MCM – *MEMS Capacitive Microphone*), jak už název napovídá, je kapacitní čidlo. Je založen na stejném principu jako kondenzátorový (kapacitní) mikrofon s tím rozdílem, že MEMS mikrofon je značně miniaturizován.



Obr. 3: MCM s horním a spodním portem od firmy AKUSTICA [4]

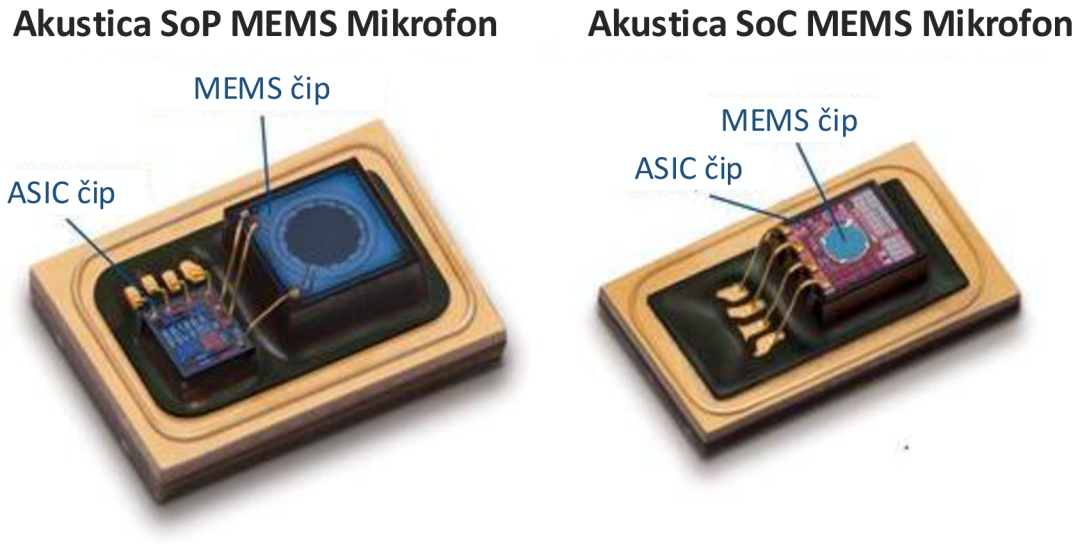
Tento typ mikrofonů je hojně využíván v moderní technice a to díky pozitivním vlastnostem. Proti nejpoužívanějšímu elektretovému mikrofonu (*ECM - Electret Condenser Microphone*), jsou jeho výhodou malé rozměry a nenáchylnost na mechanické otřesy. Nejvíce se MCM využívají ve spotřební technice a v lékařství. Ve spotřební technice je zejména využíván v mobilních telefonech, noteboocích, atd. V lékařství slouží MCM především jako hlavní součást pomůcek pro nedoslýchavé. Díky těmto odvětvím se pro MCM stále vyvíjejí a zdokonalují se jejich vlastnosti. V lékařství u pomůcek pro nedoslýchavé se jedná o co největší zmenšení rozměrů a u spotřební techniky je to především cena a zahrnutí co nejvíce integrovaných funkcí. Hlavním trendem vylepšování u všech odvětvích techniky je zvyšování SRN (odstup signál-šum, signal-to-noise), s kterým však úměrně roste cena. [1][5]

Pro zajímavost, první komerčně prodávaný MEMS mikrofon byl prodáván firmou Akustica, která zahájila prodej v polovině roku 2006.

2.1.1 Výroba

Výroba MCM se může provádět dvěma technologickými postupy, což je vlastně styl spojení dvou hlavních částí MCM. První část je mechanická, nazýváme jí MEMS. Při výrobě MEMS části se používají specializované výrobní procesy, například mikroobrábění kusu křemíku, nebo jiného materiálu používaného pro MEMS výrobu. Druhá část je specifický integrovaný obvod, tzv. ASIC (*Application-specific integrated circuit*), zpravidla jde o zesilovač, ADC, filtry a další. Tyto dvě části společně vytváří výsledný MCM, který je vkládán do ochranného pouzdra, které vždy obsahuje MEMS a ASIC část.

Jak už jsem zmínil výše, výroba MCM se může vytvářet dvěma technologickými postupy. Prvním je takzvaný SoP (*System-on-a-Packed*), někdy nazývaný jako two-chip MCM, který je modulárním spojením ASIC a MEMS částí. V praxi to znamená vytvoření součástky z těchto dvou částí umístěných vedle sebe na společném základu a vzájemné propojení pomocí vnějších spojů (drátků), viz Obr. 4. Tento technologický postup výroby MCM je používán nejčastěji. Druhý typ se nazývá SoC (*System-on-a-Chip*), někdy nazývaný jako one-chip MCM, to znamená monolitická integrace ASIC a MEMS částí na jeden čip, viz obr Obr. 4. Tento technologický postup výroby MCM má patentovaná firma *Akustica*. Rozdíl mezi těmito dvěma technologiemi je především ve velikosti, kdy MCM vyrobený technologií SoP je rozměrově menší, ale pozor velikost nezávisí pouze na technologii výroby, ale také na dalších faktorech, jako jsou vnitřní obvody a výstupní interface. [1][7][4]



Obr. 4: Přehled dvou technologických postupů výroby MCM [7]

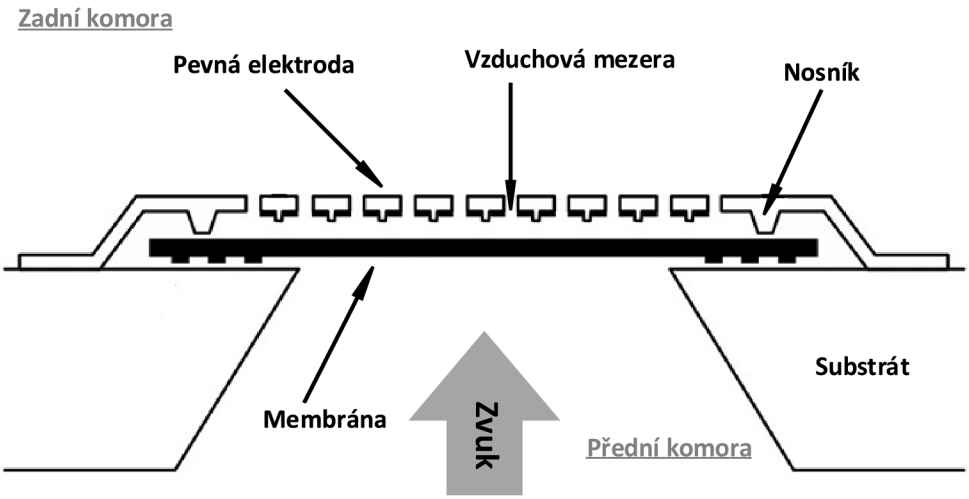
2.1.2 Princip

Mechanická část MCM je vyráběna, jak už bylo zmíněno některou z MEMS technologií. Vyrábějí se ve dvou provedení s horním portem (*top port*) a se spodním

portem (*bottom port*), což jsou otvory, kterými se dostává akustický signál na pohyblivou část MCM.

Na Obr. 5 je vidět detailní řez strukturou mechanické časti MCM se spodním portem. Hlavními prvky celé mechanické části jsou obvykle dvě elektrody, které jsou umístěny na substrátu. První elektroda nazvaná pevná, má v sobě otvory pro necitlivost na změny akustického tlaku. Druhá se nazývá membrána, to protože je pohyblivá a reaguje na změnu tlaku vyvolanou akustickým signálem (zvukovou vlnou). Membrána bývá obvykle vyrobena z tenké vrstvy křemíku a má definovanou tuhost. Tyto dvě elektrody společně tvoří základ kondenzátoru, který se díky pohyblivé membráně stává proměnným v závislosti na velikosti vstupního akustického signálu. Pohyblivá membrána odděluje od sebe dva prostory, tzv. komory. Přední komora je na straně vstupu akustického signálu a zadní komora se nachází uvnitř MCM. Tlak v přední komoře je proměnný vlivem akustického signálu, tlak v zadní komoře se nemění a je dán výrobou MCM. Prostor je vzduchotěsný od všech okolních částí MCM.

Pohyblivá membrána může být vychylována směrem k pevné elektrodě, pouze do vyrovnání rozdílů tlaků mezi komorami. To znamená, že pohyb membrány je tak velký dokud se vlivem akustického tlaku a pohybem membrány do prostoru zadní komory, tlaky v komorách nevyrovnají. Zadní komora tedy pracuje jako rezonátor, proto mají její rozměry vliv na dynamiku membrány. Pokud by se jednalo o MCM s předním portem, označení komor by se otočilo. Také by se otočily membrány, pohyblivá membrána by byla na horní straně substrátu a vstupní otvor (port) pro akustický signál by se nacházel v opačném rohu než je samotný MEMS prvek. Výsledný MCM je ještě zapouzdřen do vnějšího obalu, který má v sobě otvory (porty). Podle varianty spodní nebo horní port, viz Obr. 3. [1][7]

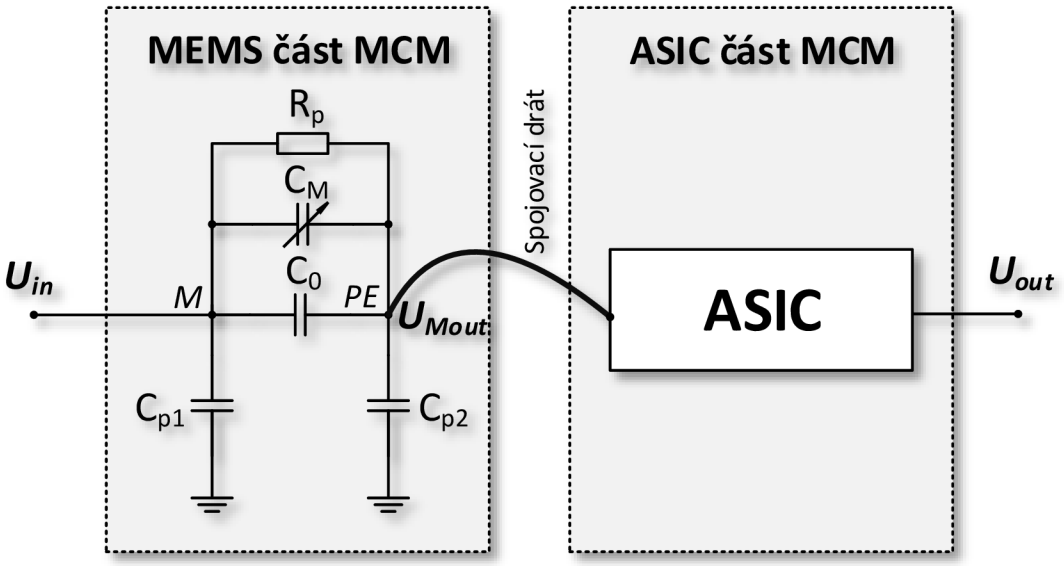


Obr. 5: Struktura mechanické části MCM se spodním portem [6]

Po připojení DC napětí na vstup MCM, na pohyblivou membránu, nastane polarizace struktury a vytvoření malého vychylovacího náboje mezi elektrodami. Vznikne tzv. elastická síla, která přitáhne pohyblivou membránu ke statické elektrodě a vytvoří tak novou výchozí pozici pohyblivé membrány. Velikost vzduchové mezery se tedy změní,

tato změna je závislá na velikosti napětí připojeného na pohyblivou membránu a tuhosti membrány. Tímto napětím se nastavuje citlivost MCM, čím je hodnota napětí větší, tím větší je i citlivost mikrofonu. Napětí má však své omezení - prahovou hodnotu tzv., pull-in. Tato hodnota napětí odpovídá hraniční pozici pohyblivé membrány, kdy už se pohyblivá membrána přitáhne k pevné elektrodě a to také odpovídá maximální citlivosti. Pull-in nastává v 1/3 vzdálenosti mezi elektrodami nepřipojeného mikrofonu na napětí. V praxi se hodnota napětí obvykle volí na 70 % prahového napětí.

Náhradní schéma MEMS části MCM je zobrazeno na Obr. 6. Vstupní DC napětí U_{in} je zapojeno na pohyblivou elektrodu *M* (membránu). Mezi pohyblivou elektrodou *M* a pevnou elektrodou *PE* je kapacita C_0. Parazitní odpor R_p je odpor vzduchové mezery mezi elektrodami a jeho hodnota vysoká. Obě dvě elektrody mají parazitní kapacity C_{p1} a C_{p2} proti zemi. Kapacita C_M je proměnná kapacita mezi elektrodami, která je proměnná vlivem akustického signálu. [1]



Obr. 6: Blokové schéma MCM v SoP

Výstupní AC napětí U_{Mout} je přímo úměrné změně kapacity celé MEMS části MCM. Napětí lze vypočítat podle vzorce (1), kde lze zanedbat parazitní odpor R_p. Tento výstup je u MCM v provedení SoP propojen s ASIC vnějším spojením (spojovacím drátem). Vnitřní struktura ASIC je zobrazena obecně, každý výrobce MCM ji obsazuje různými komponenty, popsané v následující kapitole. Tvar a paramenty výstupního napětí U_{out} podléhají vnitřní struktuře ASIC. [1]

$$U_{Mout} = \frac{C_M U_{in}}{C_{p1} + C_{p2} + C_0} \; [V] \qquad (1)$$

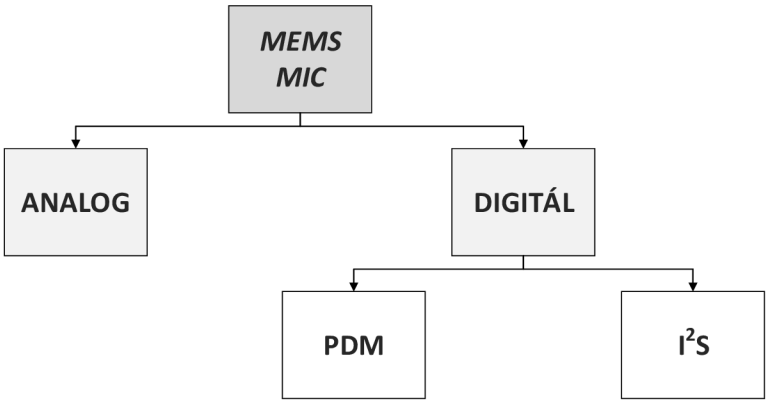
Odlišné řešení přináší firma *Vesper*, která využívá piezoelektrických materiálů k výrobě MEMS mikrofonů. Ty umožňují lepší reakci na akustický signál něž u MCM. Je to z důvodu vzduchové mezery mezi elektrodami, viz Obr. 5, která působí jako tlumení membrány, a tak brání volnému pohybu. Díky tomu, že *Vesper* mikrofon nepodléhá

tomuto tlumení je dosaženo vysokého citlivosti a SRN, s kterým je také spojena snímací vzdálenost mikrofonu. [8]

ASIC část, také nazývaná IC nebo jako RI (*Readout Interface*), je část MCM, kde se převedený akustický signál z MEMS části upravuje a moduluje na výstupní signál. Existuje mnoho variant provedení, záleží na typu MCM, viz následující kapitola.

2.1.3 Rozdělení

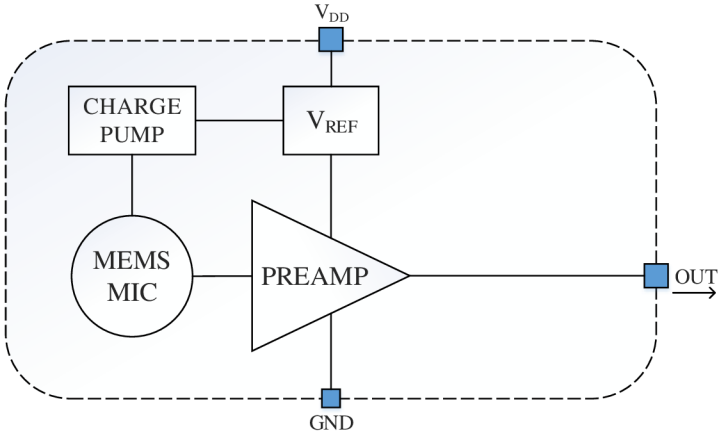
MEMS kapacitní mikrofony se dají dělit podle více kritérií. Jak můžete vidět na obrázku Obr. 7, kde je vidět hlavní rozdělení MCM na dvě základní varianty - analogové a digitální. Název je odvozen z typu jejich výstupního signálu.



Obr. 7: Rozdělení MEMS kapacitních mikrofonů

MEMS část je u obou variant stejná, liší se pouze obvody v ASIC části. U obou variant jsou v ASIC části obsaženy pomocné obvody, které se skládají z referenčních obvodů a nábojové pumpy. Nábojová pumpa má za úkol zvýšit vstupní napětí do MEMS části. Vyšší napájecí napětí vede k dosažení větší citlivosti, tato vlastnost vychází z prahové hodnoty napětí MEMS části MCM.

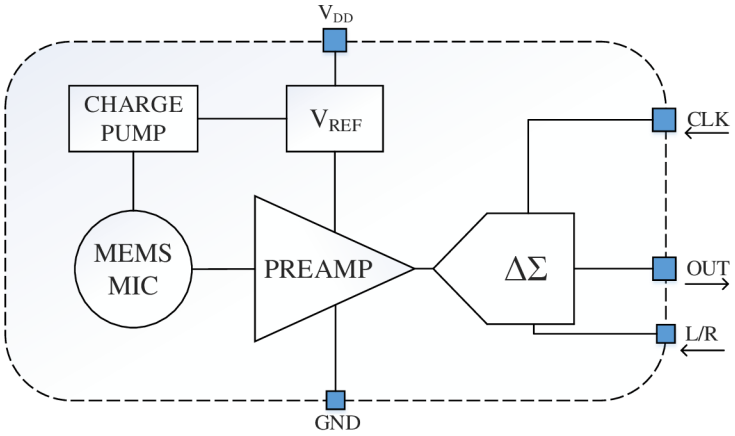
U analogové varianty mikrofonu, viz Obr. 8, je v ASIC části kromě výše zmíněných pomocných obvodů zařazen předzesilovač, obvykle operační zesilovač v neinvertujícím zapojení. Analogový MCM má na připojení pouze tři svorky - napájecí napětí, zem a výstupní analogový signál.



Obr. 8: Analogový MEMS kapacitní mikrofon

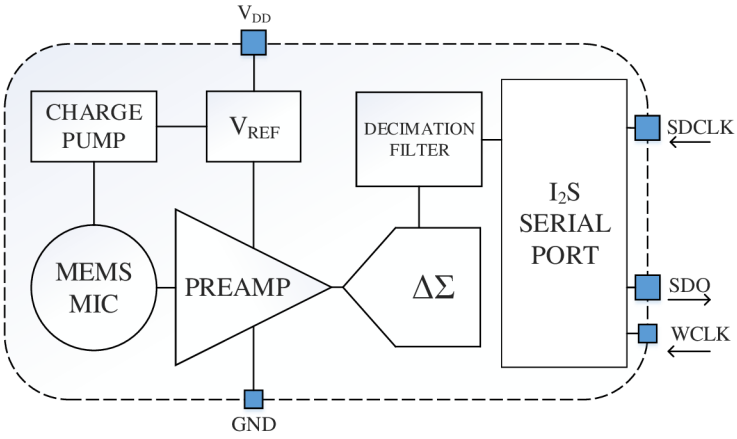
Digitální MCM se dělí podle formátu výstupního signálu na dva typy PDM (*Pulse-Density Modulation*) a I^2S. Oba tyto typy obsahují, stejně jako u analogových MCM, pomocné obvody a předzesilovač. Rozdíl nastává ve formátu výstupního signálu.

První typ má formát výstupního signálu PDM, jehož výstupní signál je tzv. bitstream. Pro převod analogového signálu na digitální modulovaný v PDM formátu se používá sigma-delta A/D převodník (ΔΣ *ADC*), který disponuje vysokou rozlišovací schopností. Tato varianta MCM má pro připojení pět svorek. Dvě svorky slouží pro napájení (napájecí napětí, zem), další svorky slouží pro nastavení A/D převodníku. Svorka CLK slouží pro přivedení hodinového signálu, který se obvykle pohybuje v rozmezí od 1 – 3,25 *MHz* (záleží na výrobci) a typicky se tato hodnota volí 2,4 *MHz*. Svorky L/R slouží pro nastavení pravého nebo levého kanálu. Neznamená to ale, že MCM obsahuje dva mikrofony, jedná se o nastavení reakce na nástupnou nebo sestupnou hranu hodinového signálu. Toho se využívá pro spojení dvou mikrofonů na jednom vedení. Poslední svorka OUT obsahuje výstupní signál ve formátu PDM. [9]



Obr. 9: Digitální MEMS kapacitní mikrofon typ PDM

Druhý typ digitálního MCM má formát výstupního signálu typu I^2S. Pro převedení výstupního signálu z MEMS části na typu I^2S formát se používá stejná ASIC část, jako u předchozího typu PDM. Navíc se ale za A/D převodník přidá decimační filtr, který decimuje signál modulovaný ve formátu PDM a převádí ho do formátu PCM. Takto připravený signál je přiveden do I^2S sériového portu, odkud je připraven na odesílání pomocí I^2S sériové komunikace. ASIC část tohoto typu MCM je složitější než u předchozích typů, což se projeví na velikosti a spotřebě, která je větší. Pouzdro obsahuje pět svorek, dvě napájecí a jedna SDCLK (*Serial Data Clock*) hodiny sériové komunikace, WCLK (*Word Clock*) velikost posílaného slova a SDO (*Serial Data Out*). Tento typ MCM vyrábí pouze firma *INVENSENSE*. [9]



Obr. 10: Digitální MEMS kapacitní mikrofon typ I²S

Shrnutí variant MCM - Analogové MCM mají malé rozměry, menší spotřebu, dosahují většího SRN a větší citlivost o proti digitálním MCM. Digitální mají stejné vlastnosti, liší se pouze ve velikosti a spotřebě. Z nich je I^2S rozměrově největší i má největší spotřebu. Typ I^2S se podle dostupných produktů nevyrábí v provedení s horním portem. Prodává se jenom ve dvou provedeních, takže má omezený výběr podle parametrů a vyrábí ho pouze jedna společnost.[9]

2.1.4 Shrnutí

Výběr MCM je vždy závislý na aplikaci pro, kterou má být použit. Z výše uvedeného vyplývají výrobní postupy, principy a varianty MCM, na jejichž základě lze vybrat ten správný typ.

2.2 Ostatní mikrofony

Tyto mikrofony nejsou v provedení MEMS. Jedná se o mikrofony s velkými pouzdry. Obvykle se skládají pouze z mechanické části a neosahují žádné elektronické integrované obvody (ASIC), pak se jedná o analogovou variantu. V malé míře se vyrábí i digitální varianta s integrovaným obvodem (ASIC) ve velkém pouzdře. Některé typy těchto mikrofonů dosahují lepších vlastností než MCM a proto jsou používány pro profesionální zachycení akustického signálu, například v hudební technice.

2.2.1 Kondenzátorový mikrofon

Kondenzátorový mikrofon je založen na principu změny kapacity. Skládá se z pevné elektrody a velice tenké a lehké vodivé pohyblivé membrány, která tvoří druhou elektrodu kondenzátoru.

Dopadající zvukové vlny pohybují s membránou a tím mění kapacitu kondenzátoru. Změna kapacity moduluje procházející proud. Jedná se tedy o pasivní mikrofon.

Výroba kondenzátorového mikrofonu je velice nákladná přesto má vynikající frekvenční charakteristiku a další vlastnosti, proto se využívá pro studiové nahrávání. [10]

2.2.2 Elektretový mikrofon

Elektretový mikrofon je typem kondenzátorového mikrofonu. Je rovněž založen na principu změny kapacity. Skládá se z velmi lehké, pohyblivé elektretové membrány a pevné elektrody. Elektret je materiál permanentně elektricky nabitý. Elektrody jsou od sebe rozmístěny velmi blízko a tvoří tak kondenzátor.

Dopadající zvukové vlny pohybují s elektretovou membránou. Pohybem způsobuje změnu kapacity, kterou vyhodnotí vestavěný zesilovač tvořen JFET tranzistorem, ten mění výstupní napětí. Jedná se tedy o pasivní mikrofon.

Elektretový mikrofon má téměř rovnou frekvenční charakteristiku, vysokou citlivost, malé zkreslení, je dostupný a má nízkou cenu. [10]

2.2.3 Dynamický mikrofon

Dynamický mikrofon využívá princip elektromagnetické indukce. Je založen na pohybu cívky v magnetickém poli. Membrána je upevněna na pohyblivé cívce, která se nachází ve statickém poli permanentního magnetu.

Dopadající zvukové vlny rozkmitají membránu a spolu s ní se začne pohybovat cívka v magnetickém poli a začne vytvářet střídavé napětí přímo úměrné dopadajícímu zvuku. Jedná se tedy o aktivní mikrofon.

Dynamický mikrofon nepotřebuje napájení, je odolný proti vysokým hladinám zvuku, mechanicky odolný, nevýhody jsou malá citlivost a velké rozměry. [10] [11]

2.2.4 Uhlíkový mikrofon

Uhlíkový mikrofon je založen na principu změny elektrického odporu. Skládá se ze dvou kovových desek a velkého počtu uhlíkových zrnek. Jedna z desek je velmi tenká a tvoří membránu. Mezi membránou a pevnou deskou jsou těsně umístěna uhlíková zrnka, která jsou v trvalém kontaktu s deskami.

Zvukové vlny dopadající na membránu mění tlak zrnek, který je přímo úměrný změně elektrického odporu a mění tak stále procházející stejnosměrný proud. Jedná se tedy o pasivní mikrofon.

V dnešní době se tyto mikrofony používají zřídka. Dříve se používaly v komunikační technice. Hlavní výhodou je schopnost pracovat i na velmi malém napětí oproti ostatním mikrofonům, nevýhodou vysoký šum a zkreslení. [10]

2.2.5 Piezoelektrický mikrofon

Piezoelektrický mikrofon je založen na piezoelektrickém jevu vytvoření elektrického náboje na stranách krystalu při mechanickém namáhání. Skládá se z pouzdra a membrány, která je spojena s tenkým proužekem piezoelektrického materiálu.

Působení zvukových vln na membránu vytváří mechanický tlak na piezokrystal, který vytváří úměrný elektrický náboj. Jedná se tedy o aktivní mikrofon.

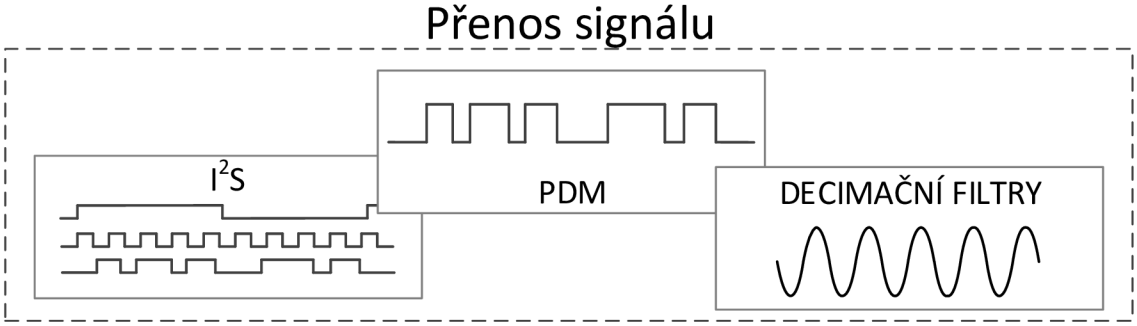
Piezoelektrický mikrofon je citlivý, nevýhodou je špatná frekvenční odezva. [11]

2.3 Shrnutí mikrofonů

Jak už jsem zmínil na začátku, prováděl jsem rešerši především MEMS mikrofonů. Z výsledků rešerše jsem pro můj navrhovaný systém volil MEMS mikrofon s digitálním výstupem formátu PDM. Především proto, že se v praxi nejvíce využívá, a tedy je mezi nimi na trhu největší výběr. Také jejich podpora bude dlouhodobá a lze jednoduše mikrofon vyměnit za novější kus. Jedním z důvodů byl také fakt, že pro tento formát výstupu není poskytnuta taková podpora pro mikrokontroléry jako pro FPGA. Proto je v této problematice prostor pro vytvoření nových přístupů.

3 PŘENOS SIGNÁLU

Dalším krokem při návrhu mého systému bylo zpracování PDM signálu v mikrokontroléru, přesněji na vývojové platformě. Blokové zapojení skupiny přenos signálu je znázorněno na Obr. 11. Vzhledem k tomu, že PDM signál z mikrofonu má frekvenci jednotek *MHz*, bylo důležité použít spolehlivý způsob přenosu dat mezi mikrofonem a vývojovou platformou, aby nedošlo k zahlcení procesoru. Pro přenos jsem využil I^{2}S interface, kterým posílám PDM signál z mikrofonu do vývojové platformy. K ukládání dat do paměti využívám přímého přístupu přes DMA. V poslední části skupiny přenos signálu je zpracování PDM signálu. Tím je myšleno převedení na PCM formát s využitím decimačního filtru.



Obr. 11: Hlavní části skupiny přenos signálu

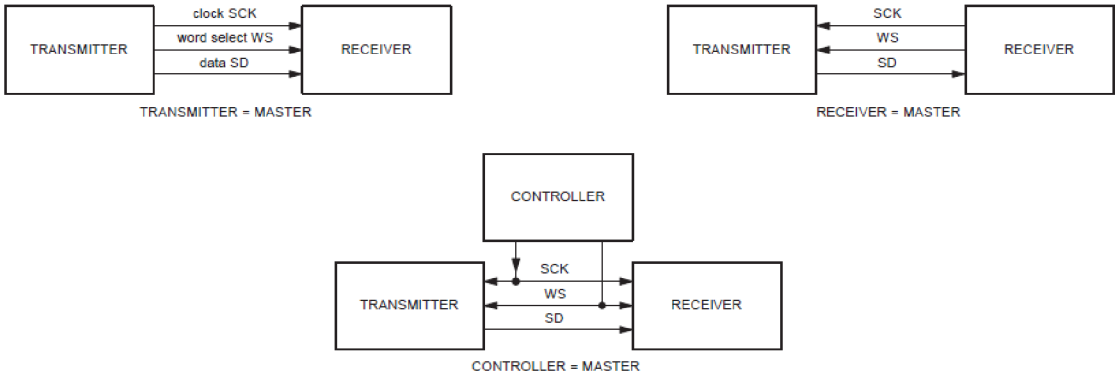
V následujících částech teoreticky popíši princip jednotlivých částí této skupiny. Samotnou realizaci z těchto principů naleznete v kapitole praktické řešení, kde je popsána praktická realizace.

3.1 I^{2}S interface

I^{2}S (*Integrated Interchip Sound*) je sériová sběrnice vytvořená pro komunikaci audio zařízení, taktéž někdy nazývaná SAI (*Synchronous Audio Interface*). Byla vytvořena pro přenos digitalizovaného audio signálu ve formátu PCM, například pro komunikaci mezi kodekem a DSP. Interface poskytuje obousměrnou, synchronní sériovou komunikaci. Pro přenos je využívána třívodičová sériová sběrnice - sériové hodiny (*serial clock*, SCK) známé taky jako bit clock (BCLK), rámcové hodiny (*frame clock*) označované jako velikost slova (*word select*, WS) a samotná data (*serial data*, SD). Značení pinů má různé varianty, záleží na jednotlivých výrobcích, zde jsem vybral pouze ty nejpoužívanější. Když jsou bitové i rámcové hodiny generovány vysílačem, jedná se o transmitter master, když jsou generovány přijímačem, jde o reicever master nebo mohou být přivedeny z externího zdroje hodin například z kontroleru, viz Obr. 12. Obvykle lze I^{2}S interface nastavit, jestli má pracovat v režimu jako master nebo slave modul.

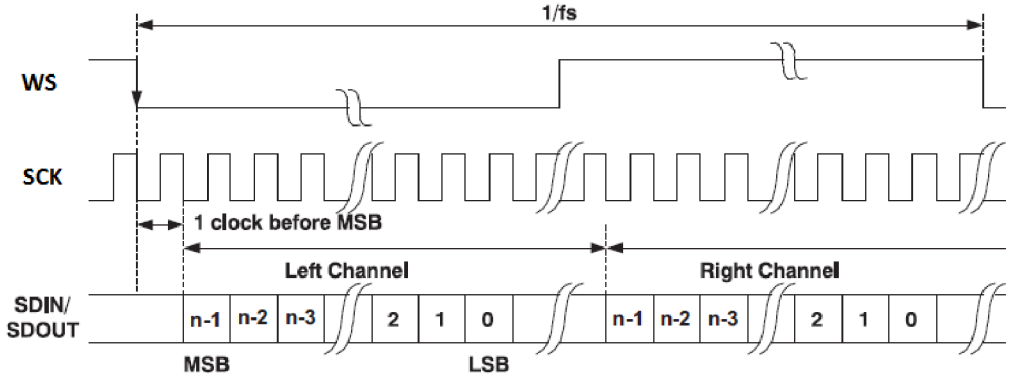
I^{2}S interface pracuje ve čtyřech základních režimech (formátech přenosu dat): I^{2}S, levé-zarovnání (*left-justified*), pravé-zarovnání (*right-justified*) a DSP. Tyto režimy jsou ve standardním třívodičovém zapojení, liší se uspořádáním datového slova, polaritou

hodin nebo velikostí datového rámce. V těchto režimech jsou data přenášena ve dvojkovém doplňku s nejvýznamnějším bitem (MSB) na první pozici. Občas se používá ještě pátý režim zvaný TDM (*Time Division Multiplexed*). Tento režim se využívá pro přenos více než dvou datových zdrojů na jednom vodiči. Pro TDM režim není stanovena pevná norma uspořádání dat v rámci jako například pro režim I²S. Proto nelze tento režim pevně specifikovat. [20]



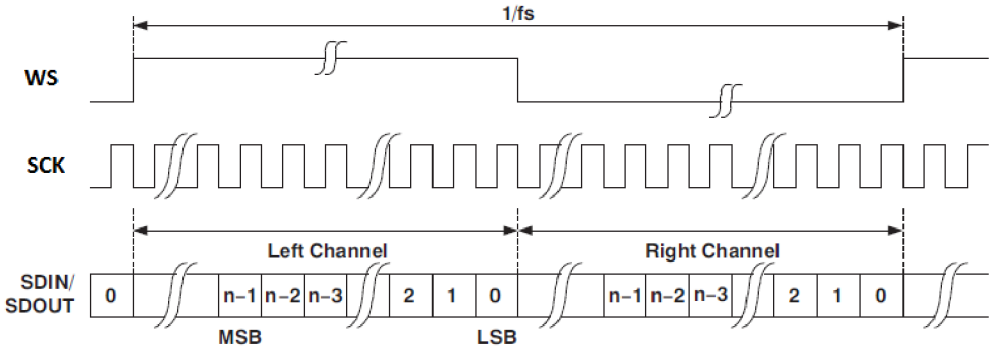
Obr. 12: Zapojení I²S mezi přijímačem a vysílačem [22]

Režim I²S je standardem firmy *Philips*, dnes *NXP*. Tento režim je používaný pro dvou kanálovou komunikaci. Frekvence bitových hodin je obvykle mezi 512 *kHz* až 12,3 *MHz*. Délku slova je možné nastavit od *8* do 32 *bitů*. Pokud je délka slova na přijímači a vysílači nastavena na jinou velikost, mohou nastat dva případy. V prvním případě je signál doplněn nulami a v druhém zkrácen. V obou případech se jedná o nejméně významné bity. Změna kanálu signálem WS se provádí na sestupnou hranu hodinového signálu. Změna hodnoty bitu je prováděna na sestupné hraně hodinového signálu a vyčítání dat se provádí na nástupnou hranu hodinového signálu. Signál data je zpožděn o jeden bit proti změně signálu WS, průběh signálů je znázorněn na Obr. 13. [20]

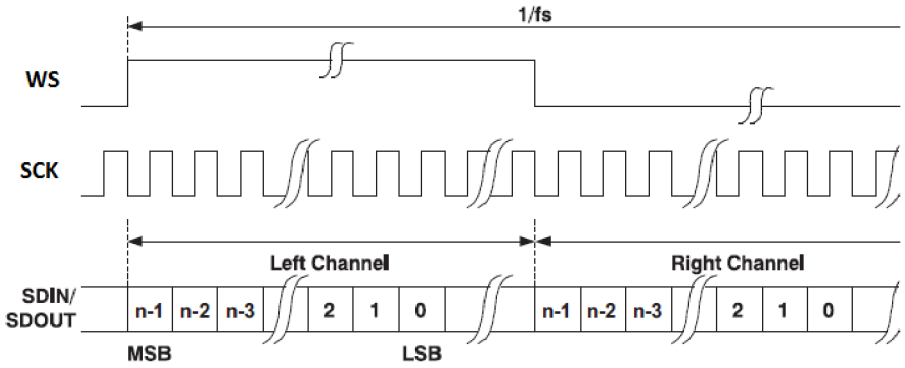


Obr. 13: Průběh signálů v I²S režimu [21]

Režimy right-justified a left-justified jsou odvozené od režimu I²S, liší se pouze ve způsobu zarovnání. Pravé zarovnání na Obr. 14, se zarovnává podle nejméně významného bitu a na Obr. 15 podle nejvýznamnějšího bitu. U pravého zarovnání je velice důležitá správnost velikosti slova u obou zařízení. [20] [21]

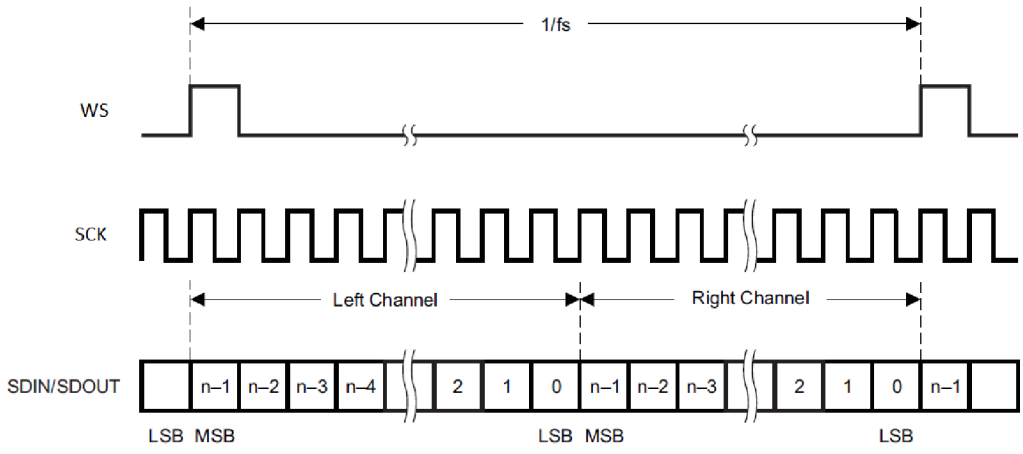


Obr. 14: Průběh signálů v režimu pravého zarovnání [21]



Obr. 15: Průběh signálů v režimu levého zarovnání [21]

Režim DSP, někdy také PCM, je určen pro kontinuální posílání nebo pro průběh více zdrojů za sebou bez nastavovacího signálu. Nástupnou hranou signálu WS se nastaví začátek nového slova a zároveň konec starého slova. Perioda těchto pulzů WS udává velikost rámce. [21]

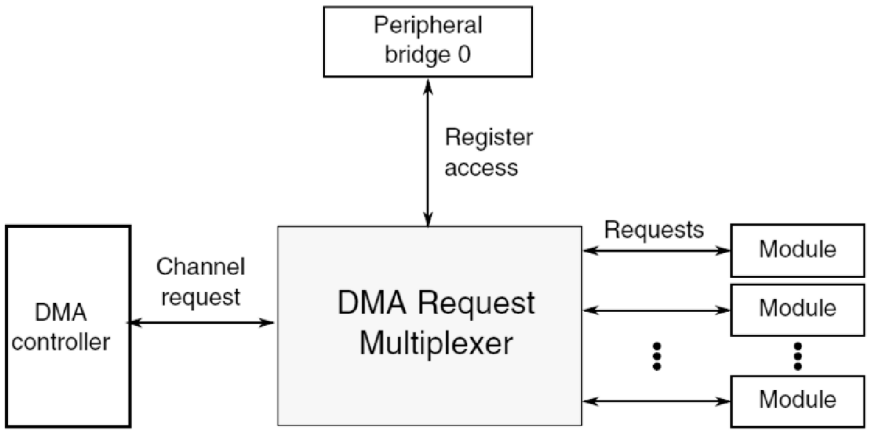


Obr. 16: Průběh signálů v režimu DSP/PCM [21]

3.2 DMA

DMA (*Direct Memory Access*) znamená přímý přístup do paměti bez použití CPU. Přímý přístup řídí DMA kontroler (DMA řadič). DMA multiplexor (DMAMUX) na základě

příkazů od kontroleru mapuje určité žádosti na nezávislé DMA kanály. Například FRDM-K64F umožňuje přenést 63 žádostí od periferií a mapovat je do 16 nezávislých DMA kanálů. To znamená, že FRDM-K64F umožňuje přenášet data z 16 periferií najednou přímo do paměti RAM nebo na jiné adresní místo bez požití CPU. Řazení kanálů se řídí podle nastavené priority kanálu. Po naplnění nastavené velikosti v paměti DMA kontroler vygeneruje přerušení o naplnění paměti. DMA zle nastavit i jako kruhový buffer, například pro přenos vysokofrekvenční komunikace, která by zahlcovala procesor, například I^2S. Zde jsou data do paměti zapisována periodicky a nepřetržitě. Blokové schéma zapojení DMA je zobrazeno na Obr. 17. [19]



Obr. 17: Blokové zapojení DMA [19]

3.3 Zpracování PDM signálu

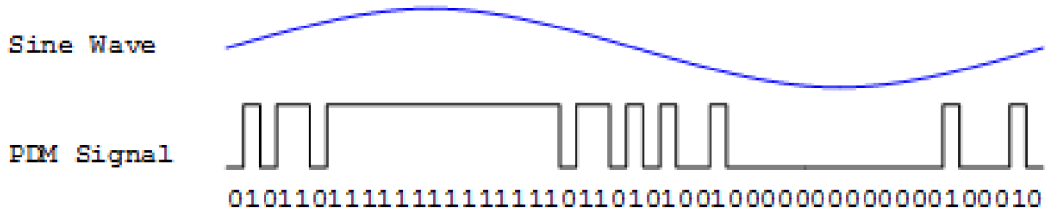
V následujících odstavcích jsou popsány témata, se kterými jsem pracoval při zpracování signálu z mikrofonu. První kapitolou je popis PDM signálu, dále princip a funkce PDM modulátoru a v poslední kapitole se věnuji převedení a zpracování signálu pomocí filtrů.

3.3.1 PDM signál

PDM signál je jednou z forem modulací používaných pro reprezentaci diskrétních signálů. Pro převod ze spojitého signálu do diskrétního PDM signálu se požívají tzv. PDM modulátory, viz následující kapitola. PDM signál je řada skokových změn referenčních úrovní, také nazývaná jako bitstream, tedy binární bitový tok. Pro zakódování informace používá pouze dvou úrovní: 0 - 1. Úrovně představují jednotlivé bity binární soustavy, ze kterých se následným převodem pomocí filtrů získá PCM signál. Jak je vidět na Obr. 18, princip modulace spočívá v relativní hustotě pulzů, ze které vychází i název PDM (*Pulse Density Modulation*). Relativní hustota pulzů odpovídá velikosti amplitudy spojitého signálu. [15][17]

Rozdíl mezi PDM modulací a PCM modulací je v tom, že PDM modulace má pouze 1bitové rozlišení. Z toho vyplývá, že přesnost modulace je závislá pouze na velikosti

frekvence vzorkování. Po převedení PDM na PCM lze získat signál s vysokým rozlišením bez nežádoucího šumu. To znamená, že je potřeba vysoká výpočetní rychlost a proto se většinou tento typ modulace používá na FPGA. Pokud je ale použit správný přístup, lze ji použít i na výkonných MCU.



Obr. 18: Průběh PDM signálu [16]

Díky vysokému rozlišení se tento typ modulace nejčastěji používá v audio průmyslu. V hudebním průmyslu se pro PDM signál také používá název DSD.

3.3.2 PDM modulátor

PDM modulátor slouží k převedení spojitého signálu na diskrétní. Neznámějším PDM modulátorem je sigma-delta A/D převodník, který je obsažený ve všech typech digitálních mikrofonů. Výstupní signál modulátoru obsahuje pouze dvě úrovně, proto je rozlišení výstupního signálu 1bitové, viz výše. Doba trvání jednoho bitu je dána frekvencí externích hodin, na které závisí přesnost modulátoru. Frekvence externích hodin je rovna vzorkovací frekvenci A/D převodníku. Čím bude tato frekvence větší, tím bude signál více převzorkován. Proto je tento typu A/D převodníků zařazen do oversampling převodníků. Díky tomu bude prostor pro kvantizační šum větší a v propustném pásu ho bude méně. Důležitým parametrem je řád modulátoru, ten také určuje podíl kvantizačního šumu v užitečném signálu. Řád modulátoru určuje počet sériově zapojených integrátorů. Modulátor svým chováním představuje filtr, pro užitečný signál se chová jako dolní propust a pro šum naopak jako horní propust. S řádem modulátoru roste schopnost oddělovat propustné pásmo šumu. Modulátory vyšších řádů je ale obtížné navrhnout, jsou více náchylné k nestabilitě. Charakterem PDM modulátoru je nízký podíl šumu v propustném pásmu a poměrně vysoký podíl šumu v části nad propustným pásem.[13][15]

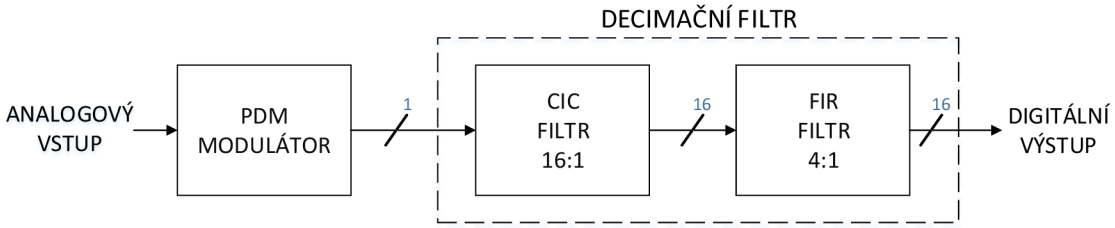
U mikrofonů se typicky nacházejí modulátory čtvrtého řádu, což je kompromis mezi funkčností a složitostí. Řád modulátoru zjistíme z frekvenčního spektra nepřevedeného PDM signálu a z jeho strmosti signálu (šumu) nad propustným pásmem.

3.3.3 Decimační filtr

Aby se mohlo s PDM signálem dále pracovat je potřeba nejdříve odfiltrovat vysoké frekvence, které obsahují šum. Dále je potřeba zmenšit počet vzorků, při kterém se

1bitový signál transformuje na signál s vyšším rozlišením například na 16bitové. Výsledkem je výstupní signál stejný jako z Nyquist-rate A/D převodníků tedy PCM signál, ale s daleko menším podílem kvantizačního šumu. Pro tuto transformaci se používá proces zvaný decimace. Jednoduše jde o redukci a průměrování vzorků v jedné operaci. Jsou hlavní dva body, které má decimační filtr za úkol.

Prvním úkolem je odstranění vysokých frekvencí. Jsou to frekvence nad propustným pásmem, ve kterých je obsažena velká část kvantizačního šumu. PDM modulátor potlačuje kvantizační šum v propustné oblasti a většina šumu je obsažena nad touto oblastí. Decimační filtr má za úkol odstranit tuto oblast a zbavit se tím většiny kvantizačního šumu. Poté co je oblast s šumem odfiltrována, je možné zredukovat počet vzorků podle Nyquistova kritéria. Tedy na polovinu vzorkovací frekvence po decimaci, což je druhý úkol. Proto je důležité dopředu vědět, s jakými frekvencemi chceme pracovat, a podle nich zvolit parametry výstupního signálu.

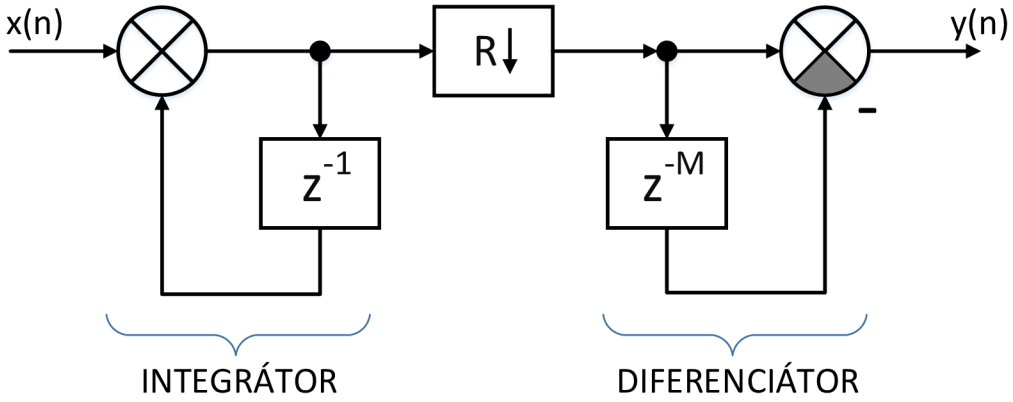


Obr. 19: Blokové zapojení vzorového decimačního filtru (CIC – R=16, N=3, M=2)

Decimační filtr je potřeba tedy navrhnout tak, aby výsledný signál nebyl amplitudově ani fázově deformovaný. Také je důležité, aby byl výpočetně co nejjednodušší pro možnost kontinuálního snímání. Nejjednodušším a výpočetně nenáročným filtrem pro snížení počtu vzorků a oříznutí vysokých frekvencí je kaskádový hřebenový integrační filtr (*CIC - Cascaded Integrator-Comb filter*). Tento typ nevyžaduje násobení, protože nepotřebuje násobící koeficienty, respektive mají hodnotu jedna. CIC filtr však není tolik efektivní v odstranění zbytkového kvantizačního šumu, proto se v praxi používá ještě ve spojení s jinými digitálními filtry, například FIR. Příklad vzorové blokové schéma decimačního filtru je zobrazeno na Obr. 19, kde modré číslice udávají bitové rozlišení výstupního signálu. [13][14]

3.3.3.1 CIC filtr

V předchozí kapitole jsem představil decimační filtr, který se skládá z více částí. První částí je CIC filtr. Jak už jsem výše zmínil, jedná se o efektivní filtr pro snížení počtu vzorků, tzv. decimaci a pro odfiltrování vysokých frekvencí ze signálu, tedy funkce dolní propusti. Základními stavebními prvky jsou integrátor a diferenciátor. CIC filtry se také využívají pro zvýšení počtu vzorků tzv. interpolaci, ale ta není předmětem popisu. Dále se budu věnovat pouze decimaci spojenou s CIC filtry. CIC filtr se skládá ze tří základních částí, viz schéma CIC filtru Obr. 20.



Obr. 20: Zapojení CIC filtru

První částí je tzv. integrátor *I*, jedná se o obyčejný integrační článek, viz jeho rovnice v časové oblasti (2) a přenosová funkce ve frekvenční oblasti (3). Do integrátoru vstupuje vstupní převzorkovaný signál, tedy pracuje s velkým počtem vzorků. Větší počet integrátorů může být mezi sebou sériově spojováno v určitém počtu, kterým pak společně s diferenciátorem udávají řád výsledného filtru.

$$y(n) = y(n-1) + x(n) \tag{2}$$

$$H_I(z^{-1}) = \frac{1}{1 - z^{-1}} \tag{3}$$

Druhou částí je decimátor *R*, který jednoduše vynechává jednotlivé vzorky v daném poměru R:1. Udává poměr počtu vzorků mezi vstupním a výstupním signálem. Počet vzorků výstupního signálu z tohoto členu je zmenšen oproti vstupnímu signálu daným poměrem.

Třetí poslední částí je diferenciátor *D*, někdy také nazývaný comb *C*. Vstupním signálem do tohoto členu je už decimovaný signál. Diferenciátor je derivační článek, ve kterém působí diferenciální zpoždění, viz rovnice pro časovou (4) a frekvenční oblast (5).

$$y(n) = x(n) - x(n-M) \tag{4}$$

$$Hc(z^{-1}) = 1 - z^{-RM} \tag{5}$$

Diferenciální zpoždění *M* udává počet nul ve frekvenční odezvě na CIC filtr. V praxi se toto zpoždění volí 1 nebo 2. Vliv diferenciálního zpoždění je vidět na průbězích, viz Obr. 21 a). Stejně jako u integrátoru můžou být diferenciátory sériově spojovány, platí ale, že počet integrátorů a diferenciátorů je totožný. Společně udává počet integrátoru a diferenciátoru řád celého filtru.

Výsledný CIC filtr se chová jako jednoduchý akumulátor, který provádí klouzavý průměr. Výstupní signál má novou vzorkovací frekvenci z původní *fs* na *fs/R*. Nová vzorkovací frekvence udává oblast frekvencí, které signál obsahuje, a ostatní jsou odfiltrovány. Výsledný přenos CIC filtru je zapsán v rovnici (6),

$$H(z^{-1}) = H_I^N(z^{-1}) \cdot H_C^N(z^{-1}) = \frac{(1 - z^{-RM})^N}{(1 - z^{-1})^N} = \left(\sum_{k=0}^{RM-1} z^{-k} \right)^N \qquad (6)$$

kde *N* je řád filtru, *R* decimační faktor a *M* diferenciální zpoždění.

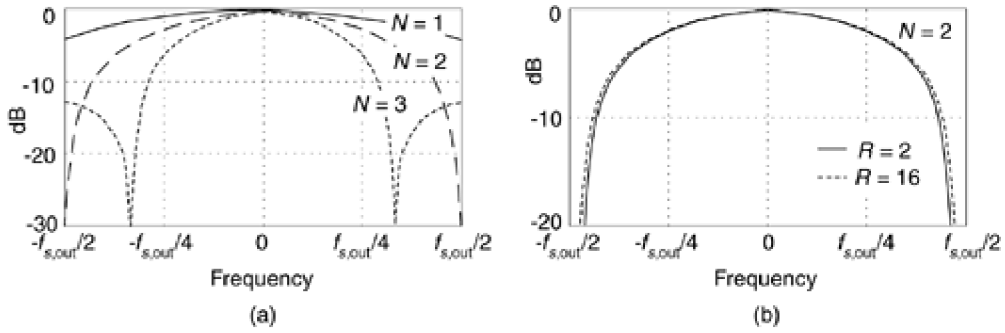
Pokud je jako vstupní signál s n-bitovým rozlišením, výsledné rozlišení je dáno vztahem (7),

$$Bout = N \cdot log_2(RM) + Bin \quad [bit] \qquad (7)$$

kde *Bout* je *n*-bitobé výstupní rozlišení, *N* je řád filtru, *R* decimační faktor, *M* velikost diferenčního zpoždění a *Bin* je n-bitové rozlišení vstupního signálu.

$$G = RM^N \quad [-] \qquad (8)$$

Zesílení celkového CIC filtru vychází ze vztahu (8).



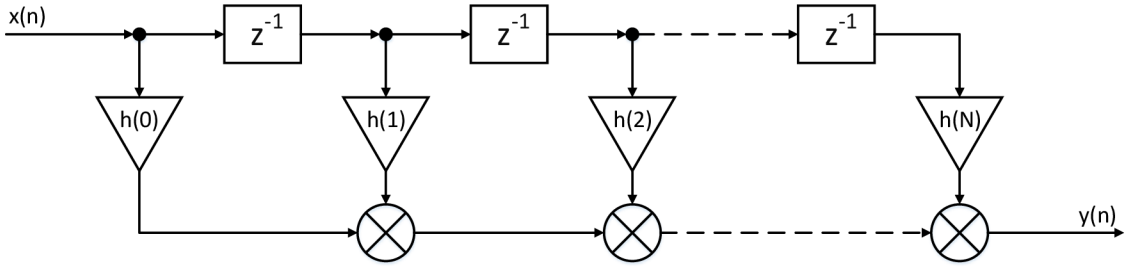
Obr. 21: Frekvenční charakteristiky CIC filtru [18]

Na Obr. 21 b) můžeme vidět vliv decimačního faktoru na tvar frekvenční charakteristiky pro totožné diferenciální zpoždění. Je zde vidět minimální vliv změny frekvenční charakteristiky, především strmosti a frekvence řezu na změnu decimačního faktoru. [12][13][17][18]

3.3.3.2 FIR filtr

FIR filtr v decimčním filtru slouží pro zvýšení strmosti frekvenční charakteristiky od frekvence řezu. Tím se odfiltruje zbytkový šum, narovná se frekvenční charakteristika v propustném pásu a zabrání se vzniku aliasingu. Návrh FIR filtru je obtížnější než CIC filtr. Obsahuje totiž koeficienty, kterými se násobí vstupní zpožďovaný signál, viz schéma zapojení Obr. 22. Koeficienty FIR filtr se musí před samotnou filtrací nejprve

početně stanovit. Řád filtru je daný jedna mínus počet koeficientů. Pro dosažení vysoké strmosti je zapotřebí vysoký řád systému, poté se ale stává neefektivní vůči výpočetnímu výkonu. Proto se opět musí volit kompromis mezi strmostí a výpočetním výkonem. Jeho výhodou je však to, že nemá zpětnou vazbu, proto nemůže být nestabilní. FIR filtr má konečnou impulsní charakteristiku. Decimace se provádí až na výstupním signálu *y (n)*. Faktor decimace je proti decimaci v CIC filtru daleko menší. Obvykle se decimační faktor volí dva nebo čtyři a vyšší decimační faktor je docílen sériovým spojením většího počtu FIR filtrů. [12][13]

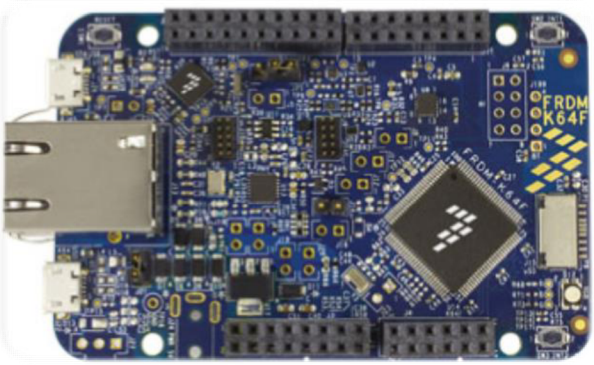


Obr. 22: Blokové schéma FIR filtru

4 VÝVOJOVÁ PLATFORMA FRDM-K64F

FRDM-K64F je výkonná vývojová platforma s vyhovujícími vlastnostmi pro můj systém. Hlavními parametry při výběru platformy byly vysoký výkon pro zpracování signálů a především Ethernet s podporou PTP hodin. Tato platforma požadavky splňovala, proto jsem systém realizoval na ni.

FRDM-K64F je velice levná vývojová platforma pro Kinetis K64, K63 a K24 MCU. Byla navržena firmou Freescale, dnes už *NXP*, ve spolupráci s mbed. Vývojová deska je vhodná pro vytváření všech různých prototypů, zejména těch, které vyžadují vysoký výkon. Ten dosahuje díky postavení na rodině ARM® Cortex-M4®. FRDM-K64F je založen na ARM architektuře s 32 *bit* Cortex-M4F jádrem běžícím na 120 *MHz*. Mezi jeho přednosti patři: výborný výpočetní výkon, díky koprocesoru pro výpočet s desetinou čárkou, vestavěné ethernetové rozhraní, velká paměť flash, snadné ovládání I/O, standardní konstrukce s možností rozšíření a vestavěný programovatelný OpenSDAv2 debug podporující CMSIS-DPA interface software. Nedílnou součástí je rozměrová kompatibilita a rozložení pinů pro Arduino R3. [2][3]



Obr. 23: FRDM-K64F[3]

4.1 Vývojářská komunita - mbed

FRDM-K64F je plně podporován mbed platformou, takže má přístup k bezplatným nástrojům a SDK, které poskytují zkušení embedded vývojáři. Její součástí je také C/C++ pro programování periferií na vysoké úrovni. Mbed poskytuje sdílení knihoven, programů a příkladů v mbed komunitě. Nalezneme zde i online kompilátor, kde lze bezplatně vytvářet programy. Programy se ukládají na server, takže se k nim dostanete odkudkoliv. Stačí se jen bezplatně zaregistrovat a vybrat typ desky. [2]

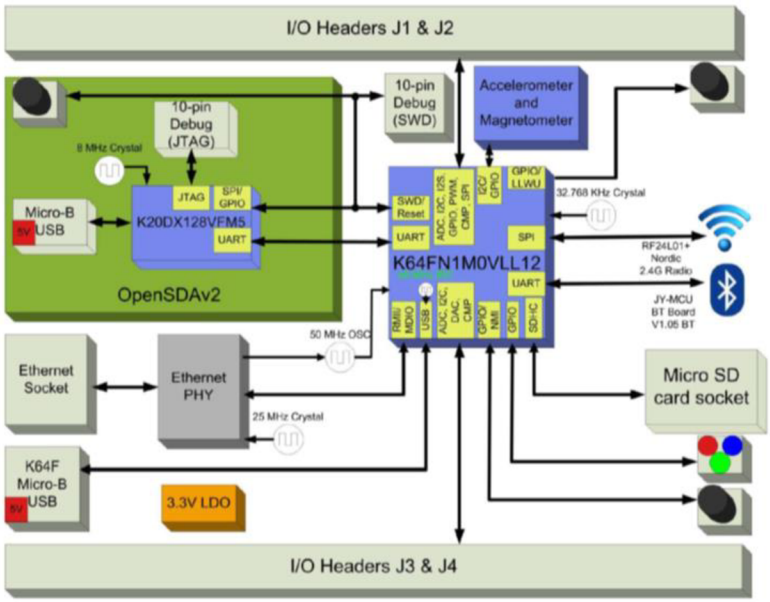
4.2 Vlastnosti [2][3]

- Freescale K64F Kinetis K64 MCU (MK64FFN1M0VLL12)
  - 32 bit ARM® Cortex®-M4+ Floating point jednotku a DSP
  - 120 MHz, 256 KB RAM, 1 MB FLASH

    - 3xSPI, 3xI2C, I2S, 5xUART, USB OTG/Host/Device, USB regulátor, PWM, 2xADC (16 bit), DAC (12 bit), GPIO, Komparátor, CAN
- Vnější periferie
    - FXOS8700CQ - 3-osý akcelerometr a magnetometr
    - Ethernet
    - SDHC
    - RGB LED
    - 2 tlačítka
- Konstrukční parametry
    - Rozměry: 81 mm x 53 mm
    - Napájení: 5V USB nebo 4,5-9 V
    - Built-in USB drag 'n' drop FLASH programátor
- Vývojářské webové stránky: mbed a SDK
    - Online Compiler
    - Vysoká úroveň C/C++ SDK
    - Návody s radami a příklady ostatních uživatelů
    - CMSIS-DPA
    - USB Serial port

4.3 Technické parametry[2][3]

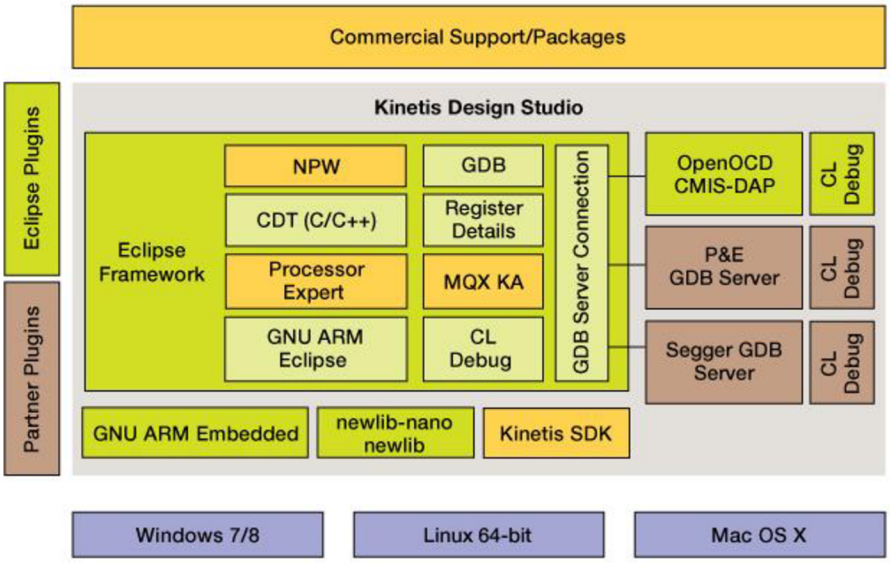
- Napájení z USB nebo připojení 5 – 9 *V* na externí napájení desky (VIN)
- Digitální IO 3,3 *V* a na jeden pin 4 *mA*, všechny piny maximálně 400 *mA*
- Externí napájení desky na pinu Vin (5-9 *V*, 100 *mA*)



Obr. 24: Blokový diagram FRDM-K64F [3]

4.4 Vývojové prostředí KDS (IDE)

Pro práci s vývojovou platformou jsem používal vývojové prostředí KDS (*Kinetis Design Studio*). KDS je bezplatné vývojové prostředí pro všechny Kinetis mikrokontroléry i pro jednotlivé vývojové desky. Umožňuje robustní editaci, kompilaci a ladění jakýchkoli návrhů. KDS je postaven na základu Eclipse, obsahuje open-source software, gcc, GNU Debugger (GDB) a jiné doplňující nástroje, které se dají volně přidávat. To vše vede k ulehčení návrhu bez omezení velikosti. K tomu všemu KDS ještě obsahuje nástroj nazývaný Processor Expert, ten ulehčuje konfiguraci jak vnitřních tak i vnějších periferií. Pomocí formulářového nastavování lze nakonfigurovat potřebné periferie a využívat i předem vytvořené funkce k jejich základnímu ovládání. V následujících odstavcích naleznete rozsáhlejší popis možností jednotlivých částí KDS.



Obr. 25: Blokové schéma vnitřní struktury KDS

KDS IDE je postavené na Eclipse Luna. Eclipse je univerzálním skeletem většiny vývojových prostředích, který obsahuje celou řadu důležitých nástrojů například: GNU ARM Embedded, OpenOCD DMIS-DAP, atd., viz Obr. 25.

Možnost přidání partnerských pluginů například GBD debugger podporuje následující ladící rozhraní: SEGGER J-Link, P&E Multilink, CMSIS/DPA.

KDS podporuje také Eclipse pluginy, ke kterým patří operační systémy pro MCU (MQX RTOS, Free-RTOS).

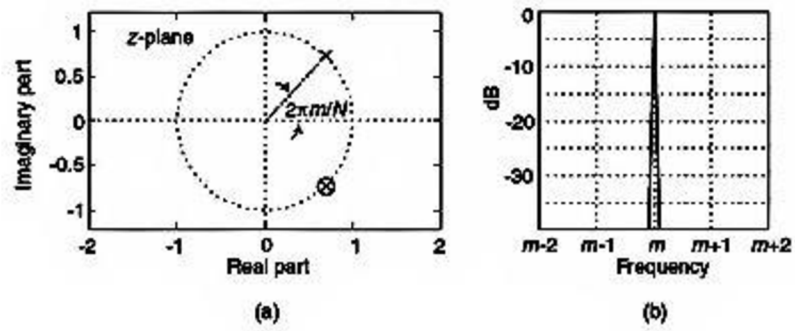
Pro vytvoření nových návrhů lze používat standardní knihovny, které obsahují základní ovládaní MCU, nebo dodatečně doinstalovat a používat rozsáhlejší knihovny KSDK (*Kinetis Software Development Kit*), které podporuje již zmíněný Procesor Expert.

5 ZPRACOVÁNÍ SIGNÁLU

V této kapitole zpracování signálu je popsána pouze jedna metoda, která převádí signál z časového spektra na spektrum frekvenční. Systém jsem se snažil navrhnout tak, aby byl co nejjednodušší. Tato metoda je v různých funkcích použita pro vytvoření příznaků pro klasifikační model.

5.1 Goertzelův algoritmus [26]

Goertzelův algoritmus (*Goertzel Algorithm*) je jednou z diskrétních metod na převedení signálu z časové oblasti do frekvenční. Za určitých podmínek je tato metoda výpočetně rychlejší než rychlá Fourierova transformace. Algoritmus je rekurzivní filtr, který se zaměřuje na předem stanovenou frekvenci ve spektru. Nejčastěji se využívá na detekci tónů a pro analýzu spektra radiových frekvencí.



Obr. 26: a) Rozložení pólů a nul filtru, b) Odezva frekvence ve frekvenčním spektru [26]

„Goertzelův algoritmus je realizován jako IIR filtr druhého řádu.“ [26] Zásadním pojmem algoritmu je nahradit obecnou funkci filtru za identifikaci zadané frekvenční spektrální složky signálu. Tento filtr počítá jednu spektrální komplexní hodnotu DFT definovanou vztahem (9).

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-jmk\frac{2\pi}{N}} \tag{9}$$

„Výstup filtru *y(n)* je roven výstupní frekvenci koeficientu DFT, *X(m)*, když index *n=N*, (…).“[26] „Pro přesný výsledek DFT musí index rezonanční frekvence *m* ležet v intervalu $0 \geq m \leq N-1$.“ [26] Převedením do z-roviny pomocí Z-transformace získáváme přenos Goertzelova filtru (10)

$$H_G(z) = \frac{Y(z)}{X(z)} = \frac{1 - e^{-j\frac{2\pi m}{N}} z^{-1}}{1 - 2\cos\left(\frac{2\pi m}{N}\right) z^{-1} + z^{-2}} \tag{10}$$

s jednou nulou $z = e^{-j\frac{2\pi m}{N}}$ a sdruženými póly $z = e^{\pm j\frac{2\pi m}{N}}$. Nula a pól se vykrátí a zbyde nám pouze jeden pól na mezi stability, protože se nachází na jednotkové kružnici z-

roviny, viz Obr. 26 a). Pro regulaci mez stability většinou znamená nestabilitu systému. U Goertzelova algoritmu to není tak nebezpečné z důvodu velkého počtu vzorků N, který obvykle bývá ve stovkách. Filtr zůstává stabilní pro tyto krátké časové sekvence, protože po uložení dat do interních registrů se na začátku nového bloku data resetují na nulu.

Na Obr. 26 b) je znázorněn frekvenční rozsah filtru s frekvenční odezvou v žádané frekvenci $\frac{2\pi m}{N}$, která odpovídá periodické frekvenci $\frac{mFs}{N}$ $[Hz]$. Z toho vyplývá, že žádanou rezonanční frekvenci m vypočítáme podle vztahu (11).

$$m = \frac{f \cdot N}{f_s} \tag{11}$$

Kde *f* je požadovaná frekvence, *N* počet vzorků a f_s vzorkovací frekvence.

Z důvodu realizace pomocí IIR filtru má filtr nekonečnou impulsní charakteristiku, a proto má úzkou frekvenční odezvu.

Diferenční rovnice v časové oblasti je vyjádřena vztahy (12)(13).

$$\mathrm{w}(n) = 2\cos\left(\frac{2\pi m}{N}\right) w(n-1) - w(n-2) + x(n) \tag{12}$$

$$y(n) = w(n) - e^{-j\frac{2\pi m}{N}} w(n-1) \tag{13}$$

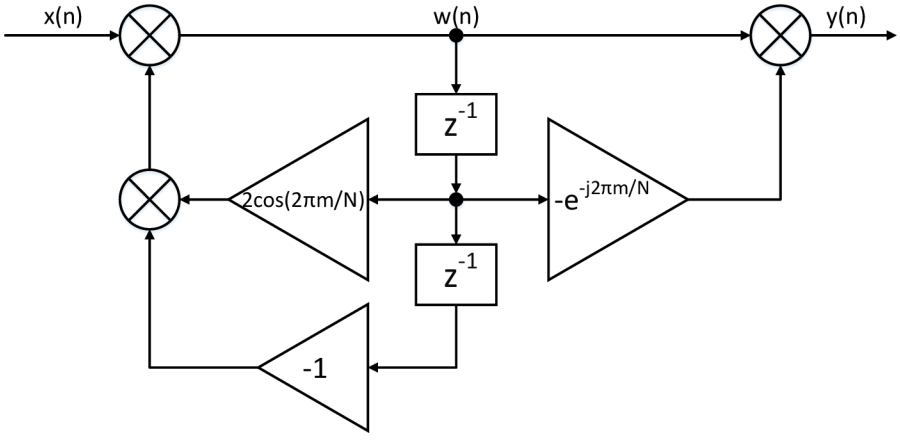
Z rovnic plyne *N+2* reálných násobení a *2N+1* reálných součtů pro výpočet jedné spektrální hodnoty DFT.

Počet prvků *N* může být libovolné číslo, čím bude větší, tím bude lepší frekvenční rozlišení a větší odolnost proti rušení. Zvýší se tím ale i počet výpočtů. Rezonanční frekvence *m* může být jakékoliv číslo v rozsahu *0* a *N-1*.

Výkonu jedné spektrální hodnoty DFT vypočítáme ze vztahu (14).

$$|X(m)|^2 = |X(N-1)|^2 =$$
$$= \mathrm{w}(N-1)^2 + \mathrm{w}(N-2)^2 - \mathrm{w}(N-1)\mathrm{w}(-2)\left[2\cos \mathrm{w}(\frac{2\pi m}{N})\right] \tag{14}$$

Skutečné blokové zapojení Goertzelova algoritmu je zobrazeno na Obr. 27, kde jsou i výpočty jednotlivých koeficientů pro softwarové začlenění.



Obr. 27: Blokové schéma zapojení Goertzelova algoritmu

6 STROJOVÉ UČENÍ

Strojové učení je jednou z oblastí umělé inteligence. Jeho konkrétní zařazení je velice obtížné, protože prolíná spoustou vědních oborů. Strojové učení (SU) se zabývá algoritmy, které lze na základě zkušeností učit, tzn. automaticky zdokonalovat jejich schopnosti. Výsledkem je model, na který lze nahlížet jako na funkci, která na základě vstupních dat predikuje výsledek. [28] [29]

6.1 Vstupní data

Ze vstupních dat, které jsou na počátku řetězce, chceme získat informaci a z informací znalost, která odpovídá funkci (modelu). Obecně se vstupní data dělí na dvě základní skupiny - kvalitativní a kvantitativní data.

Kvalitativní data jsou nečíselné hodnoty, které svůj popis vyjadřují slovně (např. barvu, pocit teploty, chuť, dosažené vzdělání). Z příkladu je vidět, že některá data se dají řadit a lze u nich použít logické operátory (vzdělání vysoké>nízké), takovým datům říkáme ordinérní. Ostatní data jsou nominální. O nich lze pouze říci, že jsou jiná nebo stejná (červená ≠ modrá).

Kvantitativní data jsou naopak číselná, která mají jednotku (např. procenta účasti, hodnota frekvence, velikost napětí). Ty se dělí na spojité a diskrétní. Spojité udávají hodnotu celku, například vzdálenost bodu A do bodu B. Diskrétní udávají počet, například počet lidí u lékaře.

Pří návrhu modelu se tyto typy dat dělí na data trénovací a testovací. Trénovací data (verifikační) slouží pro naučení modelu. Skládají se z N-tice záznamů obsahující (x,y), kde x je příznakový vektor o n příznacích (atributech) a y je výstupní třída (label). Testovací data jsou používána pro určení přesnosti modelu, skládají se také z N-tice (x,y). Většinou se ale k učení modelu používají všechna data a k určení přesnosti modelu se používá některá z metod odhadu chyby modelu, viz kapitola Odhad chyby modelu. Využitím celé množiny dat (trénovací i testovací) pro učení modelu můžeme v případě kvalitních dat dosáhnout lepšího naučení modelu a lepších výsledků predikce. [28]

6.2 Předzpracovaní dat

Předzpracování dat je v řetězci SU zařazeno proto, aby upravovalo vstupní data. Úpravou se myslí převedení na správný datový typ, normalizaci, eliminaci irelevantních příznaků, odstranění outliears a další úpravy vstupního signálu. Obecně je předzpracování dat operace, která upravuje vstupní data za účelem zlepšení klasifikačních schopností modelů, například odstraněním příznaků ze vstupních dat, které nenesou žádnou informaci nebo nesou pouze slabou nebo dokonce chybnou informaci o výstupní třídě. Záleží také na tom, pro jakou metodu modelování jsou data připravována. Každá metoda má svoje specifikace. [28]

6.2.1 Úprava dat

Jednou z prvních operací předzpracování je bezpochyby změna datového typu, například binární na kvalitativní. Datový typ se mění v závislosti na použité metodě. Další úpravou je normalizace příznaků. To znamená upravení hodnot příznaků tak, aby všechny patřily do určitého intervalu. Používají se normalizace - lineární tzv. min-max normalizace v intervalu $\langle 0; 1\rangle$, $\langle -1; 1\rangle$, střední hodnotou a rozptylem, nelineární tzv. soft-max normalizace logaritmickou funkcí v intervalu $(0; 1)$, $(-1; 1)$.

Další krokem předzpracování je odstranění nulových nebo odlehlých dat (outliers). Outliers jsou odlehlá data, která jsou vzdálenější než *n*-násobek rozptylu od průměru příznaků, obvykle se tento násobek volí 3. Pokud naopak data někde chybí, například jsou nulová, nebo chybí pouze jedna hodnota, data se nahradí průměrem okolních hodnot. Při velkém počtu dat je vhodnější celý příznakový vektor odstranit, aby nedošlo k narušení vzájemných závislostí. [29][28]

6.2.2 Selekce příznaků

Selekce příznaků je jednou z nejdůležitějších operací při vytváření nějakého klasifikačního systému. Selekce příznaků se provádí ze všech vstupních příznakových vektorů, z kterých se snažíme vybrat pouze ty příznaky, které nesou nejvíce informací o určité výstupní třídě. To znamená příznaky mající největší vzdálenost průměru mezi všemi výstupními třídami a nejmenší rozptyl mezi svojí výstupní třídou. Díky tomu se sníží celkový počet příznaků, který vede k menší výpočetní náročnosti, pokud se příznaky vytváří při běhu systému. Také se sníží počet stupňů volnosti. Metody se dělí na metody wrapper a filter.

Metody typu filtr jsou méně výpočetně náročné, a proto se používají pro velký počet vstupních dat. Obvykle je výsledkem těchto metod seřazení příznaků od nejlepšího k nejhoršímu. K těmto metodám patří například AUC, GiniIndex a T-test. Tyto metody nejsou tak efektivní jako wrapper vzhledem k tomu, že pro selekci nepoužívají konkrétní klasikační model.

Metody typu wrapper jsou mnohem efektivnější než metody filtr. Je to především díky tomu, že pracují přímo s konkrétně používaným klasifikačním modelem. Za to jsou ale výpočetně náročnější. K nejznámějším wrapper metodám patří dopředná selekce, zpětná selekce a brutal-force. Všechny metody využívají kritérium kvality predikce. Kritérium kvality se získá z některých metod pro odhad chyby predikce s konkrétním klasifikačním modelem, například Cross Validation. [28][32]

6.2.2.1 Dopředná selekce

U metody dopředná selekce (Forward Selection) se nejdříve určí kritérium kvality u všech dílčích příznaků. Do prázdné množiny se vybere nejlepší příznak a k němu se přidávají další příznaky z předchozího kroku a počítá se kritérium kvality, výsledkem je nejlepší dvojice. A takto se pokračuje až do zadaného počtu *k* nejlepších příznaků. [28]

6.2.2.2 Zpětná selekce

U zpětné selekce (Backward Selection) se nejdříve určí kritérium kvality všech příznaků. Z této množiny se v dalším kroku odebere vždy příznak, který působí nejmenší změnu kritéria kvality. Takto se pokračuje do té doby, než zbyde požadovaný počet *k* nejlepších příznaků. [28]

6.2.2.3 Brutal-force

Brutal-force (Brute Forse) počítá kombinace všech příznaků a vybírá příznaky s nejlepším kritériem kvality. [28]

Z těchto metod dokáže zaručit optimálního nalezení množiny příznaků pouze metoda brutal-force. Tato metoda je ale velice výpočetně náročná, protože zkouší všechny kombinace příznaků, odtud plyne její název. Dopředná ani zpětná selekce nezaručují nalezení optimální kombinace příznaků, ale jsou méně výpočetně náročné. [28]

6.3 Klasifikační modely

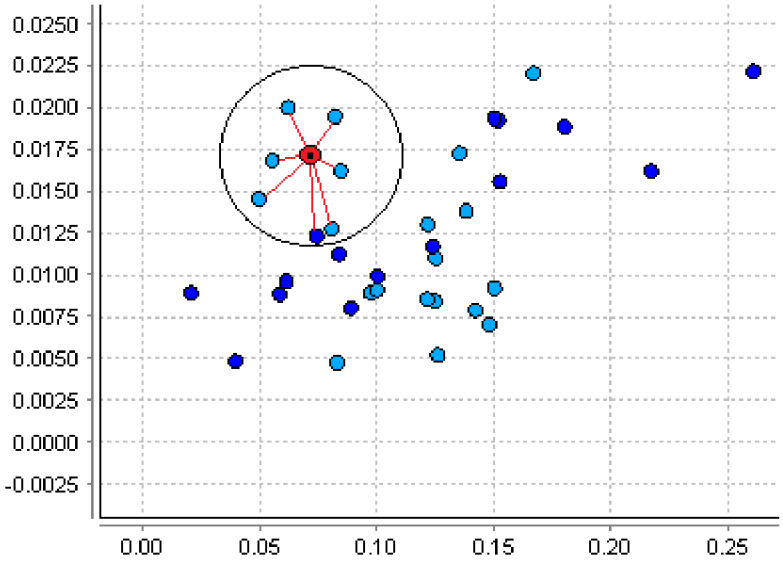
Klasifikační modely mají za úkol přiřadit objekty do některé předem určených výstupních tříd, díky znalostem získaných analyzováním trénovacích dat daných tříd. Po naučení modelu závisí predikce *y* pouze na hodnotách vstupních dat *x*, které jsou tvořeny příznaky (atributy). Jedním z nejznámějších příkladů klasifikace je rozpoznávání spamových emailů na základě obsahu textu ve zprávě. [30]

Obr. 28: Klasifikační model

6.3.1 k nejbližších sousedů (k-NN)

Metoda k-NN patří do skupiny učení založené na instancích (*IBL - Instance-based Learning*). To znamená vstupní instance *x*, u které chceme určit výstupní hodnotu *C*. K určení výstupní hodnoty *C* je potřeba mít uložené instance se známou výstupní hodnotou. Parametr *k* určuje počet nejbližších prvků mezi vstupní instancí *x* a uloženou známou instancí. Výstupní hodnota *C* a vstupní instance *x*, je dána počtem *k* nebližších prvků vůči známé uložené instanci. Pro výpočet výstupní hodnoty se používá například průměr nebo modus *k* nejbližších prvků. Při výpočtu se může taky uvažovat váha souseda podle jeho vzdálenosti, která vede k větší efektivnosti. Nejčastěji se pro určení vzdálenosti vstupní instance a uložené instance používají metriky, jako jsou euklidovská,

hammingova nebo kvantitativní vzdálenost. Pro standardizaci dat je nutné použít normalizaci, která je ale odvozena od trénovacích dat. [28] [29]



Obr. 29: Klasifikace výstupní třídy metodou k-NN pro k=7

6.3.2 Naive Bayes

Metoda Naive Bayes vychází z Bayesova vzorce, se kterým v druhé polovině 18. století přišel anglický duchovní Thomas Bayes, která je blízká lidskému uvažování. Naivní Bayesovský klasifikátor nebo také jednoduchá Baysovská klasifikace je založena na Bayesově vzorci (15), platný za podmínky P(x) > 0:

$$P(H_k|X) = \frac{P(X|H_k)P(H_k)}{P(X)} = \frac{P(X|H_k)P(H_k)}{\sum_{i=1}^{k} P(X|H_i)P(H_i)} \qquad (15)$$

,kde k je počet výstupních tříd, H_k je hypotéza výstupní třídy pro X, X je měření, $P(H_k|X)$ je podmíněná ((a)posteriorní) pravděpodobnost hypotézy H_k (pravděpodobnost hypotézy H_k, že X patří do výstupní třídy C), $P(H_k)$ (a)priorní pravděpodobnost H_k (pravděpodobnost výskytu výstupní třídy C), $P(X|H_k)$ je věrohodnost (vyplývá z ověřené zkušenosti), $P(X)$ pravděpodobnost nastolení měření X.

Výpočet $P(X|H_k)$ je dán součinem všech dílčích pravděpodobností (16).

$$P(X|C_j) = \prod_{i=1}^{n} P(x_i|C_j) \qquad (16)$$

,kde k je počet výstupních tříd, n je počet příznaků, C_j výstupní třída.

Výsledná klasifikace Naivního Bayesovského klasifikátoru je dána vztahem (17),

$$C = \arg\max_{j=1\ldots k} P(C_j) \cdot \frac{\prod_{i=1\ldots n} P(x_i|C_j)}{\sum_{i=1}^{k} P(x_i|C_i)P(C_i)} \qquad (17)$$

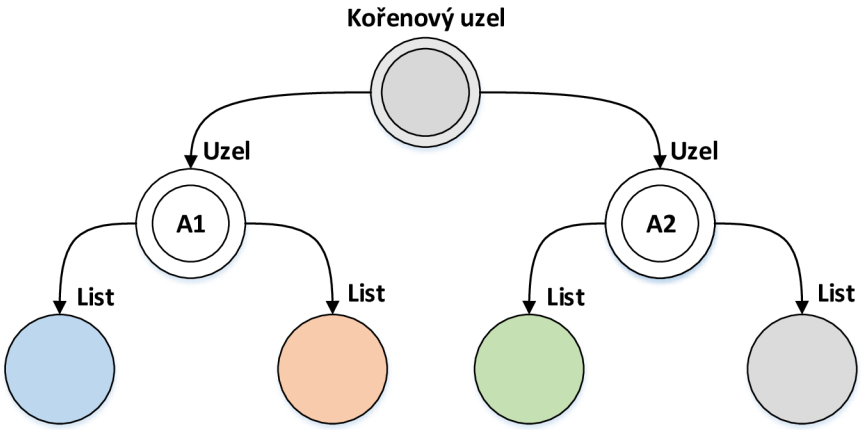
Za předpokladu, že jsou vstupní příznaky vzájemně nezávislé, může výsledný vztah klasifikace zjednodušit a získáme (18).

$$C = \arg\max_{j=1\ldots k} P(C_j) \cdot \prod_{i=1\ldots n} P(x_i|C_j) \qquad (18)$$

Nalezením argumentu maxima získáme číslo výstupní třídy. $P(C_j)$ lze zvolit jako $1/k$ a tím určit pravděpodobnost výskytu pro všechny výstupní třídy stejnou. [29]

6.3.3 Rozhodovací stromy

Rozhodovací stromy (RS) jsou v SU používány především pro svoji přehlednost, jednoduchost, nelinearitu, interpretovatelnost a rychlost. Používají se pro klasifikaci do výstupních tříd především na základě kvalitativních vstupních příznaků, ale lze je i použít na kvantitativní data. Rozhodovací strom se skládá z následujících částí. Na počátku je tzv. kořenový uzel, kterým začíná větvení RS. Dále následuje uzel, to je další stupeň RS po počátečním rozdělení v kořenovém uzlu, který se ještě větví. Spojnice mezi uzly se nazývá větev. Koncový člen, který se dále nevětví a udává výslednou výstupní třídu, se nazývá list. Hloubka stromu je určena počtem větví v přímé cestě stromem, viz Obr. 30. Základem správně pracujícího RS je vhodná volba a rozdělení vstupních příznaků. [29][28]



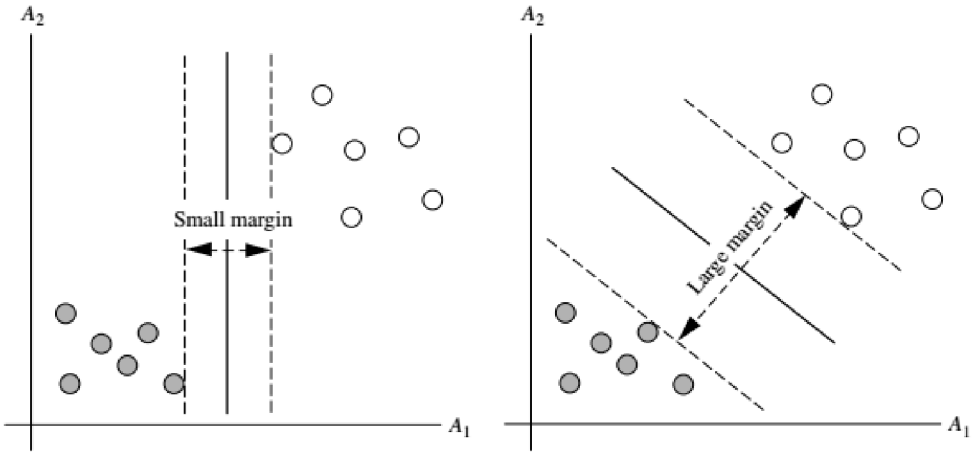
Obr. 30: Rozhodovací strom

Existují algoritmy, které tyto stromové strukturu automaticky vytvářejí, například CART, ID3, C4.5 a další. Základní funkcí těchto algoritmů je vhodný výběr příznaků a jejich vhodné rozdělení pro dosažení co největší efektivity klasifikace.

6.3.4 Support Vector Machine (SVM)

SVM (*Support Vector Machine*), je algoritmus který hledá nadrovinu, která nejlépe lineárně odděluje *N*-tici příznaků jedné výstupní třídy od druhé, tzv. lineárně separující nadrovina (rozhodovací hranice). Nejlepší je v tomto případě myšleno největší prostor meze krajními příznaky jednotlivých tříd. SVM jako i další metody klasifikace byla

vytvořena pro binární klasifikaci, ale je možné pomocí různých přístupů vytvořit multiclass. Například sdružení ostatních výstupních tříd do jedné a porovnávat jí proti hledané třídě. Na Obr. 31 je vidět příklad lineárně separovatelné nadroviny. Algoritmus tedy hledá nejlepší umístění nadroviny, aby co nejlépe oddělovaly výstupní třídy.

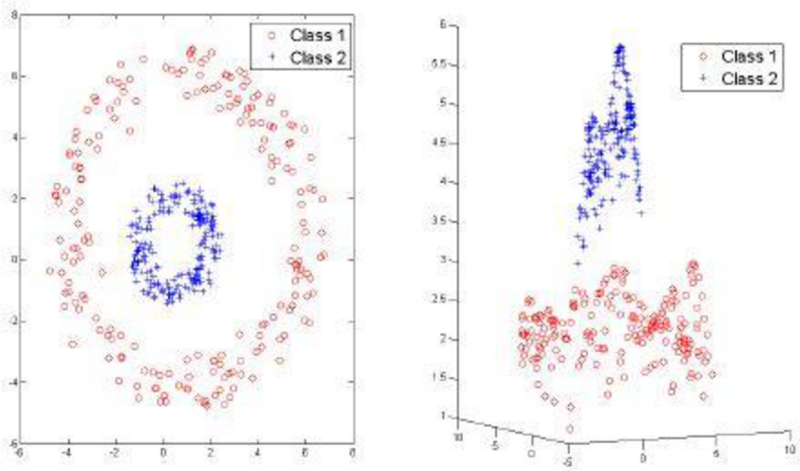


Obr. 31: Lineárně oddělitelné výstupní třídy [29]

Krajní body na čárkovaných přímkách jsou maximální krajní body, hodnoty příznaků *A1* a *A2*, které lze pro oddělení dvou tříd použít, nazývají se SV (Support Vectors). Od těchto bodů, přímek se počítá maximální mezera, která udává nejlepší umístění nadroviny. [29]

6.3.5 SVM – Kernel trik

SVM – Kernel trik je algoritmus, který nelineárním mapování transformuje trénovací příznaky do vyšší dimenze a tam hledá lineární nadrovinu, která nejlépe odděluje *N*-tici příznaků jedné výstupní třídy od druhé, tzv. separující nadrovina. Metoda se používá v případech, kdy není možné lineárně rozdělit trénovací množinu příznaků, viz Obr. 32. Na levém obrázku je vidět, že není možné lineárně rozdělit dvě výstupní třídy. Pomocí nelineárního mapování do vyšší dimenze o jeden řád, viz pravý obrázek, lze už tyto třídy lineárně oddělit.

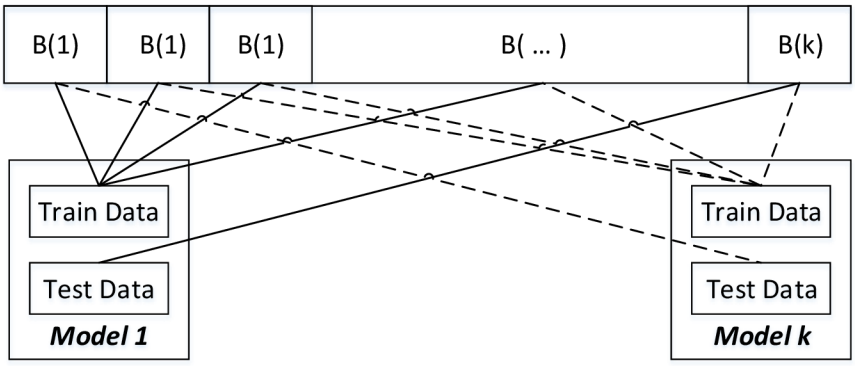


Obr. 32: Grafické znázornění nelineární transformace do jiné dimenze pomocí Gaussova jádra [31]

Mapování se provádí pomocí jádra (kernel), jeho typ je nutno vhodně zvolit. S přidaným jádrem je pak metoda používána stejně jako lineární SVM. Nejpoužívanějšími jádry jsou polynomiální, sigmoidové a gaussovo (*RDF – Radial Basis Function*) jádro. [29]

6.4 Odhad chyby modelu

Odhad chyby modelu určuje, jak přesná bude klasifikace modelu na testovacích datech. Výpočet chyby modelu se ale provádí na trénovacích datech. Jedna z metod určení této chyby je metoda křížová validace (*Cross Validation, x-validation*). Metoda pracuje tak, že se trénovací data rozdělí na testovací a trénovací, počet rozdělení je dán vstupním parametrem *k*, které se obvykle volí 10. Uvažujme rozdělení *k=10* a vytvoření deseti nejlépe stejně velkých bloků dat, ze kterých je vždy jeden blok určen jako testovací a ostatní bloky jsou sjednoceny a použity jako data trénovací. Takto se prostřídají křížem všechny bloky, viz Obr. 33. [28]



Obr. 33: Grafické zobrazení dělení Cross Validace

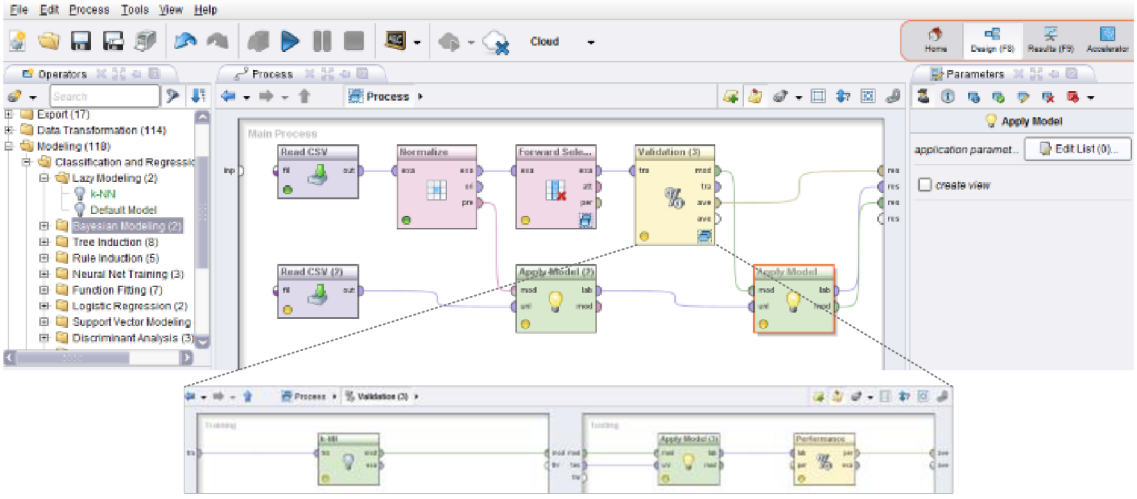
Výsledkem odhadu chyby modelu je průměrná hodnota chyby všech bloků, proto může mít výsledná hodnota velký rozptyl. Metoda se dá efektivně používat pouze pro větší počet vstupních dat ($N > 30$). V případech menšího počtu dat je vhodnější použít metodu bootstrap, která je ovšem výpočetně náročnější. [28]

6.5 RapidMiner

RapidMiner je počítačový SW, který se používá pro návrh systémů strojového učení. Software obsahuje velké množství funkcí pro načítání dat z různých zdrojů, předzpracování vstupních dat, filtraci, převody mezi datovými typy, uspořádání, detekci chybných dat, selekci příznaků, určení přesnosti modelu a samotné modely. Na výběr je velké množství modelů a jsou řazeny podle vnitřních struktur. Jsou tu například klasifikační modely líného modelování, bayesian modely, rozhodovací stromy, neuronové sítě, SVN, metamodely a další.

V RapidMineru se pracuje s blokovou strukturou návrhu, kde každá funkce má vlastní blok, který je možné dále nastavovat. Některé bloky je potřeba doplnit vnitřní strukturou,

například x-validation nebo loop, viz Obr. 34. Ke spojování bloků slouží spoj připomínající drát. Výstupy a vstupy bloků jsou označeny příslušnými zkratkami, aby se spojování ulehčilo. Alespoň jeden výstup musí být spojený s pravou stěnou vývojové plochy, aby se dal návrh spustit. Průběh návrhu plyne z levé strany do pravé. Výstupem je například přesnost modelu, nebo vstupní data, záleží na tom jaké a kolik bloků jsme spojily s pravou stěnou. Na Obr. 34 je vidět návrhové prostředí. V horní části prostředí je panel možností, kde jsou různé funkce ukládání a také tlačítka START, STOP a PAUSE. Na levé straně je strukturované menu s výběrem funkcí, uprostřed je hlavní prostor pro spojování bloků, kde i jsou vidět samotné bloky. Na pravé straně je nabídka s možnostmi nastavení označeného bloku.



Obr. 34: Návrhové prostředí RapidMiner

Na Obr. 34 je navržený systém, který načítá data z *csv* souboru pomocí bloku *Read csv*. V první řadě zatím následuje normalizace vstupních příznaků do intervalu -1 – 1. Dále selekce příznaků pomocí dopředné selekce, zde se jako parametr nastavuje počet výstupních příznaků. Dále x-validation s vnitřní strukturou model k-NN, blok pro použití modelu na testovacích datech a výpočet přesnosti. Druhý řádek slouží pro použití testovacích dat s modelem.

7 ETHERNET

Ethernet je nejrozšířenějším prostředkem (metodou) pro komunikaci v místních sítích (LAN). Názvem Ethernet jsou označovány pouze poslední dvě vrstvy referenčního ISO/OSI modelu, viz Tab. 1. Tyto dvě vrstvy byly standardizovány jako standard IEEE 802.3. Nejpoužívanějším modelem využívající Ethernet je TCP/IP model, také nazývaný jak TCP/IP Protocol Stack. Ten se skládá pouze ze čtyř vrstev OSI modelu, viz Tab. 1. [23][24]

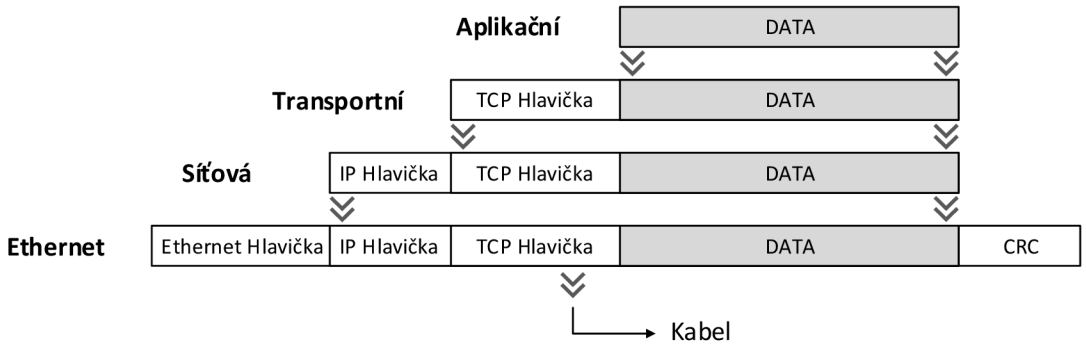
OSI model č.	OSI model	TCP/IP model	TCP/IP protokoly
7	Aplikační	Aplikace	Telnet,SMTP,POP3,HTTP
6	Prezentační		
5	Relační		
4	Transportní	Transportní	TCP,UDP
3	Síťová	Internet	IP, DHCP, ICMP
2	Linková	Linková a fyzická (Ethernet)	Ethernet, PPT, ADSL
1	Fyzická		

Tab. 1: Srovnání OSI a TCP/IP modelu a nejvíce používané protokoly

V mém systému využívám pro síťovou komunikaci model TCP/IP s transportní službou UDP a zároveň s podporovanou kompatibilitou protokolu PTP. Proto v následujících kapitolách popisuji pouze protokoly týkající se této komunikační sítě a protokolu PTP.

7.1 Ethernet TCP/IP

Jak již bylo zmíněno výše, Ethernet se skládá pouze z linkové a fyzické vrstvy referenčního modelu. Ty by však samy nedokázaly komunikovat, modulovat či zabezpečit spojení v síti. Proto se v ethernetových sítích společně s Ethernet protokolem používají další tři protokoly, souhrnně nazývány jako *Internet Protocol suite*. Komunikace mezi vrstvami jde vždy od shora dolu Obr. 35. To znamená, že nejvyšší vrstva neví o vrstvách pod sebou. Poslední čtvrtý protokol odpovídá aplikační vrstvě. Ta využívá protokoly, které zajišťují srozumitelnost mezi účastníky v síti a popisu se jí nebudu více věnovat.[24]



Obr. 35: Přehled vrstev sítě Ethernet TCP/IP

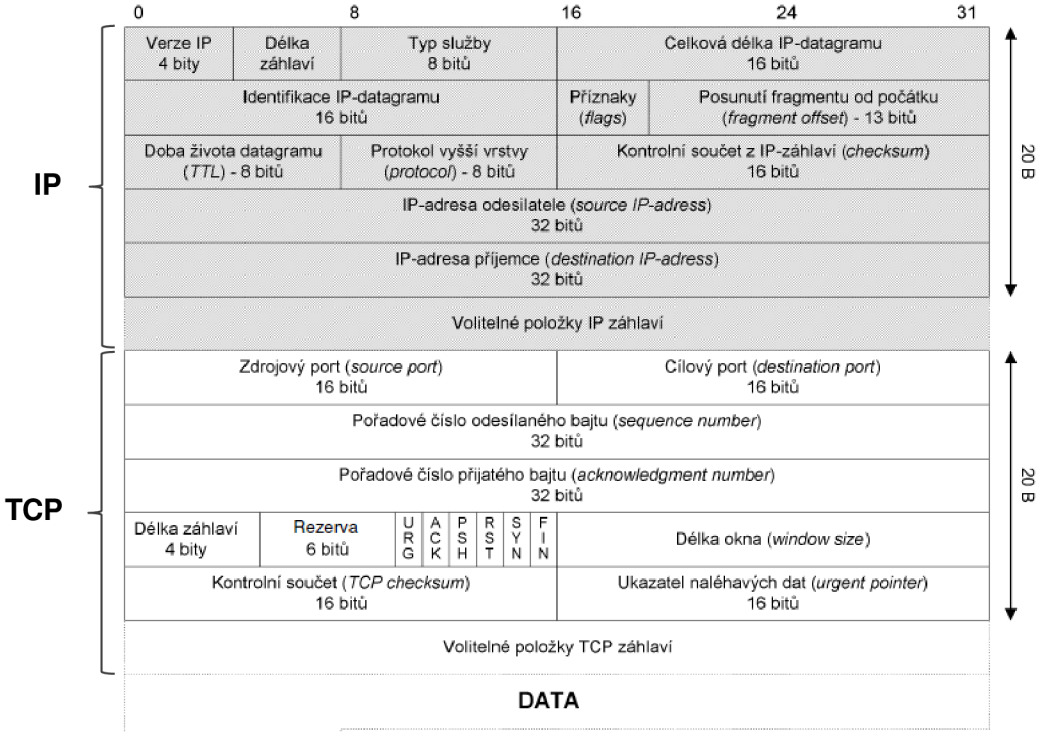
7.1.1 Protokol Ethernet

Protokol Ethernet se skládá z fyzické a linkové vrstvy ISO modelu. Tento protokol určuje nastavení spojené s fyzickým přenosem datového rámce modulovaného po bitech na elektrický signál. Přenos začíná zleva rámce od preambule a pokračuje postupně až nakonec rámce, viz Obr. 36. Poslední prvkem rámce je kontrolní součet CRC (*Cyclic redundanci check*). Kontrolní součet slouží ke kontrole přijímaného rámce. Pokud není rámec správný, je zahozen. Linková vrstva udává, jak budou datové pakety doručovány mezi zařízeními. Adresaci, mezi kterými zařízeními má komunikace probíhat, má na starosti MAC. MAC je podvrstva Ethertnet protokolu, která definuje fyzické adresy zařízení pomocí MAC adresy (48 *bit*). Komunikace je řízena principem CSMA/CD. [24]

Obr. 36: Formát rámce Ethernet protokolu B - bajty

7.1.2 Protokol IP

Síťová vrstva je v síti Ethernet TCP/IP reprezentována IP protokolem. Tato vrstva dovoluje komunikaci mezi více sítěmi, což protokol Ethernet neumožňoval. IP protokol může být složen ze dvou neznámějších protokolů IPv4 a IPv6. Jejich hlavní rozdíl je v adresovacím prostoru. IPv4 protokol má adresovací prostor 32 bitů a IPv6 má 128 bitů. Adresa nazývaná IP musí být v rámci jedné sítě jedinečná. Protokol IPv6 se od IPv4 liší ještě zjednodušenou hlavičkou, možností přidávat zprávám prioritu a umožňuje šifrování zprávy. Na Obr. 37 v horní části je zobrazeno záhlaví protokolu IPv4. [23][24]



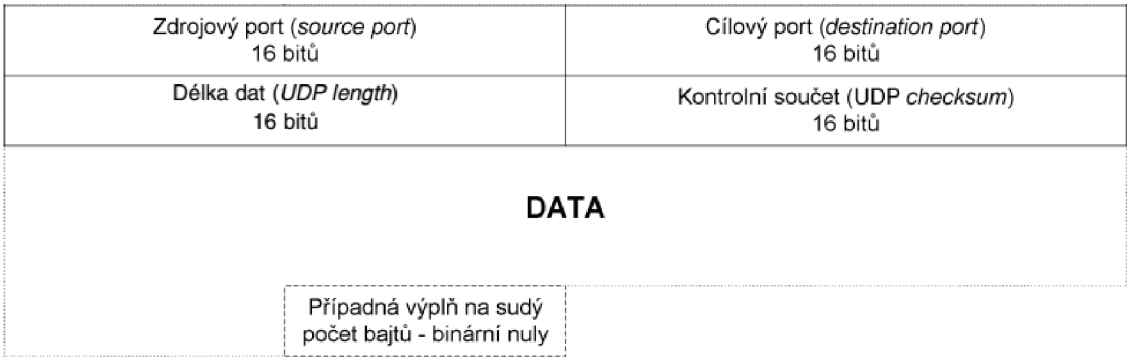
Obr. 37: Záhlaví (formát) paketu IP (IPv4), TCP [23]

7.1.3 Protokol TCP

Protokol TCP patří do transportní vrstvy. Jak je z názvu vrstvy patrné, tento protokol se stará o efektivní přenos paketů mezi sítěmi. Protokol TCP se především stará o navazování spojení mezi aplikačními programy a bezchybný přenos paketů a to potvrzením o přijetí od příjemce, proto se TCP protokol nazývá spojovanou službou. Pokud potvrzení nepřijde do daného času, paket se odešle znovu. Při navázání spojení se vytvoří dočasný oboustranný virtuální okruh, po kterém jsou oboustranně přenášeny informace. Na Obr. 37 je od středu směrem dolů zobrazeno záhlaví protokolu TCP. [23][24]

7.1.4 Protokol UDP

Jak protokol TCP tak i protokol UDP je protokolem transportní vrstvy. Na rozdíl od protokolu TCP je protokol UDP nepotvrzovaný. To znamená, že data, která se pošlou po síti, nejsou zpětně kontrolována o doručení. UDP protokol má volitelný kontrolní součet. Pokud je vypnutý může se stát, že když se data poškodí, příjemce se o tom nedozví, protože neví, jaké data byla na začátku. Ale na druhou stranu to přináší rychlejší komunikaci, menší režii i přípravu dat. Data se posílají pomocí tzv. UDP datagramů Obr. 38. [23]



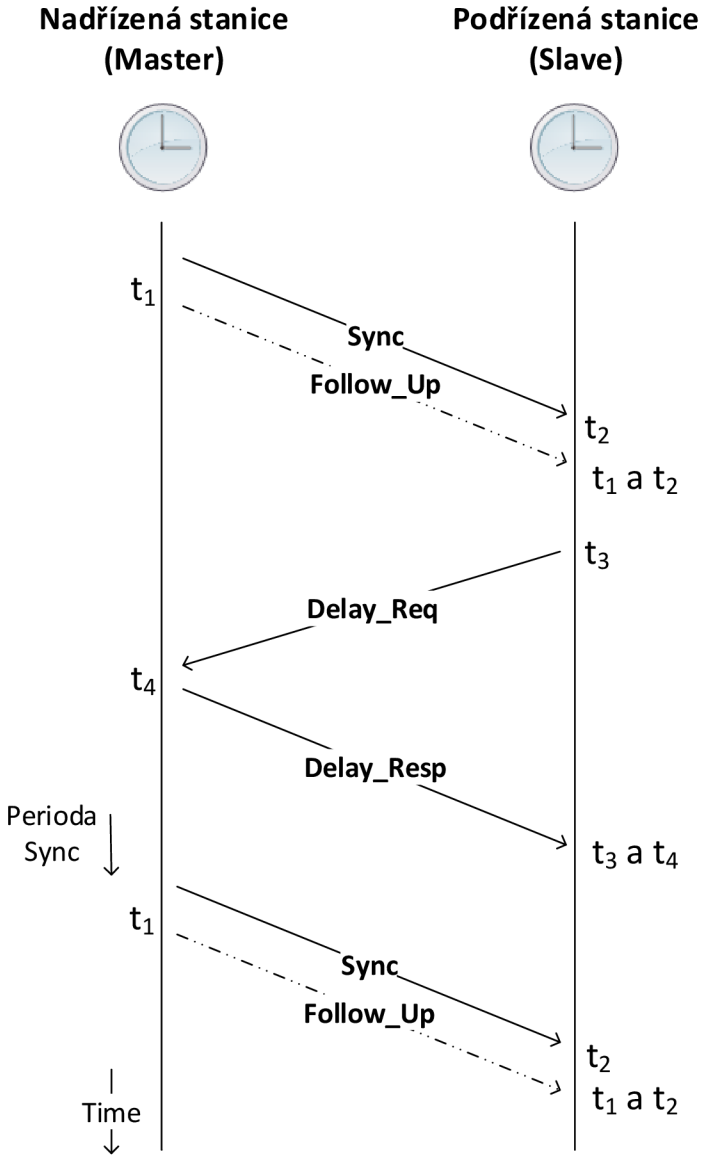
Obr. 38: UDP datagram [23]

7.2 PTP/IEEE-1588

U mnohých systémů reálného času je potřeba přesné načasování a synchronizace. K tomu slouží protokol PTP nebo také standard IEEE-1588, který pro synchronizaci využívá komunikační kanál. PTP protokol umožňuje obohatit standardní komunikaci, například Enthernet TCP/IP o vysoce přesnou synchronizaci mezi síťovými stanicemi. [25]

Princip synchronizace spočívá v posílání speciálních synchronizačních zpráv s časovými značkami, ze kterých se počítá zpoždění staničních hodin (slave) proti nadřazené stanici (master). Zprávy jsou posílány na stávající síťové komunikaci, není potřeba speciální spojení. Synchronizační zprávy jsou vysílány cyklicky s danou periodou od nadřazené stanice (master). Nadřazená stanice obsahuje hlavní hodiny, master clock, podřízená stanice slave clock. V PTP protokolu se používají tři druhy hodin.

Ordinery Clock (OC), jsou základní hodiny koncových zařízení. Boundary Clock (BC) omezují vliv nestálého zpoždění v síťových prvcích, například v routerech. Transparent Clock (TC) slouží pro kompenzaci zpoždění při přechodu v kaskádových sítích, je obsažen pouze ve verzi PTPv2. TC se dělí ještě podle typu kompenzace na Peer-to-Peer a End-to-End. Každé zařízení v síti se může stát nadřazenou stanicí a tedy master clock. K výběru se používá algoritmus Best Master Clock (BMC), který vybírá nejlepší zařízení do funkce nadřazené stanice, podle daných kritérií (přesnost, stabilita časové základny, atd.). K předáním výsledků BMC slouží zpráva *announce*, která informuje zařízení, které se mohou stát nadřazenou stanicí v síti o jejich funkci. [25]



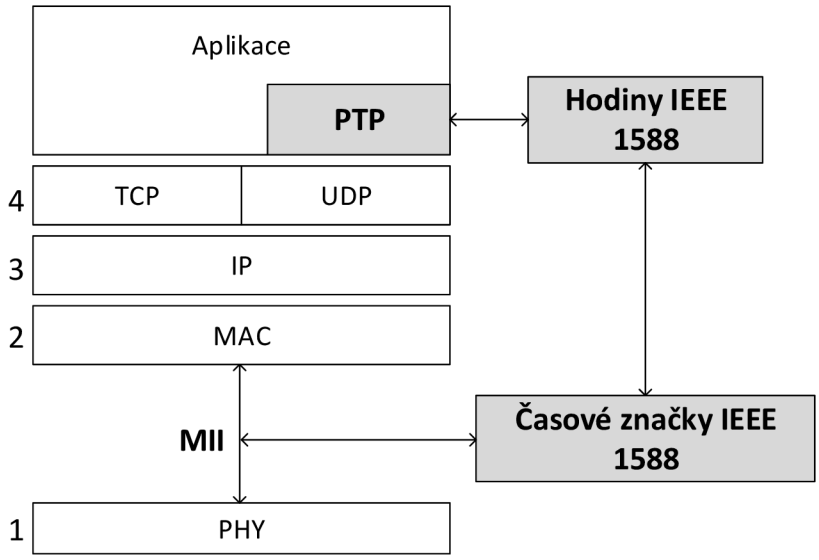
Obr. 39: Časová posloupnost zpráv předávaných v průběhu synchronizace času protokolu PTP[25]

Synchronizace hodin protokolu PTP začíná vysláním synchronizační zprávy *Sync* z nadřazené stanice na podřízené stanice. Pokud hardware neumožňuje odeslání přesného času přímo ve zprávě *Sync*, následuje od nadřazené stanice zpráva *Follow_Up* se změřeným časem odeslání zprávy *Sync*. Následuje žádost podřízené stanice

nadřazenou o velikost zpoždění způsobené přenosem. Podřízená stanice žádá zprávou *Delay_Req*, nadřazená stanice odpovídá na tuto žádost zprávou *Delay_Resp*, ve které udává čas přijetí žádosti. Ze znalosti těchto časů t_1,t_2,t_3 a t_4 lze dopočítat *offset*, který udává časový rozdíl mezi nadřazenou a podřízenou stanicí. Výsledkem synchronizace je doba značená jako *offset*, která se vypočítá podle vzorce (19).

$$offset = \frac{(t_2 - t_1) + (t_3 - t_4)}{2} \tag{19}$$

Hodnotou *offset* si opraví podřízená stanice (slave) svou aktuální hodnotu hodin. Synchronizace proběhne korektně za podmínek, když je posuv hodin konstantní v celém průběhu synchronizace, stejné zpoždění v obou směrech a přesné určení času ve stanicích. Průběh synchronizace je znázorněn na Obr. 39. Synchronizační zprávy z nadřazené stanice jsou posílány metodou multicast. [25]

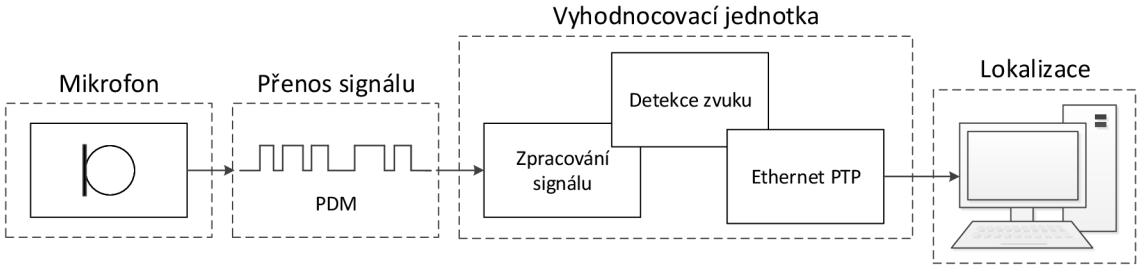


Obr. 40: Struktura Ethernetu s protokolem PTP [25]

Existují dva způsoby předávání časových značek. První je čistě softwarové, vkládání časových značek probíhá v aplikační vrstvě, kde je daleko nižší přesnost, která závisí na typu použitého OS. Druhý způsob je softwarově hardwarová, kdy software vykonává obsluhu protokolu a časové značky jsou vkládány mezi linkovou a fyzickou vrstvou v rozhraní zvané MII (*Media Independent Interface*), viz Obr. 40. Díky tomu lze dosáhnout vysoké přesnosti, protože nedochází k dalšímu zpoždění v ostatních vrstvách. [25]

8 PRAKTICKÉ ŘEŠENÍ

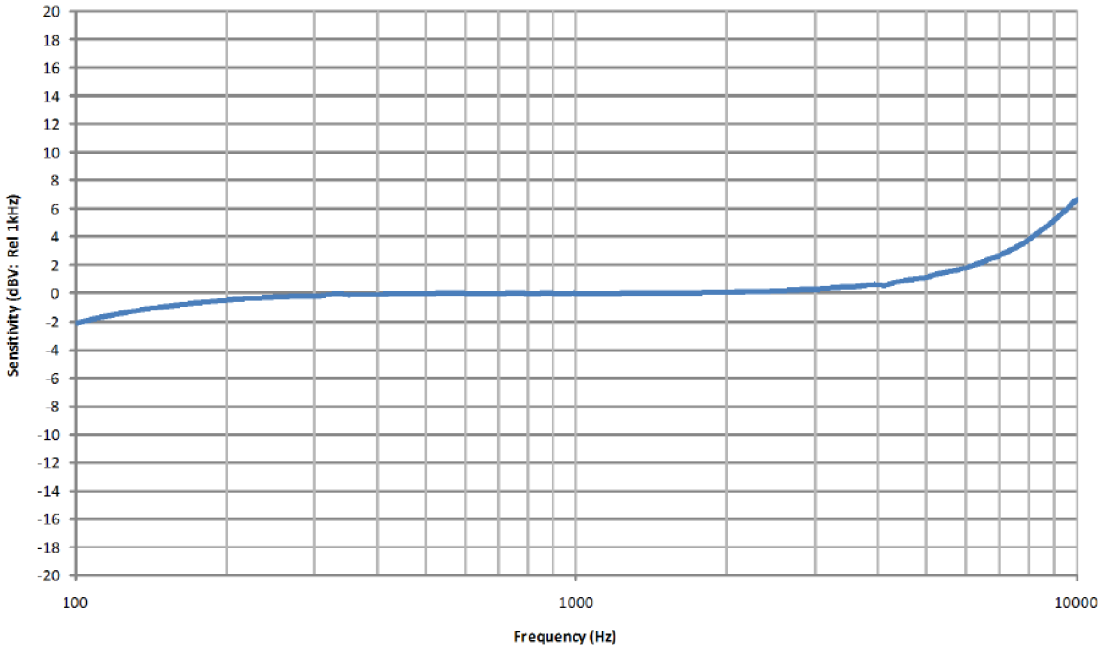
V kapitole praktické řešení popisuji postupy a úvahy, ke kterým jsem dospěl při praktické realizaci systému. Pořadí jednotlivých částí vychází z pořadí samotné realizace, které je znázorněn na Obr. 41.



Obr. 41: Blokové schéma systému

8.1 Mikrofon

Z výsledků rešerše jsem vybral mikrofon s digitálním PDM výstupem SPM0405HD4H-WB od firmy *KNOWLES*. Hlavními kritérii pro výběr byly citlivost, odstupu signál šum a frekvenční charakteristika, která však u MCM nebývá konstantní přes celé slyšitelné pásmo, Obr. 42. Také byla důležitá velikost napájecího napětí mikrofonu. Zde byl požadavek napájení 3,3 *V*, aby mohlo být použito přímo z vývojové platformy. To nebyl problém vzhledem k tomu, že je většina mikrofonů určena pro rozsah 1,6 – 3,6 *V*.



Obr. 42: Frekvenční charakteristika MCM SPM0405HD4H-WB, KNOWLES [27]

Dalším krokem bylo zapojení mikrofonu. Schéma digitálního PDM MCM s výstupními svorkami naleznete na Obr. 9. Na svorky napájení jsem tedy zapojil napětí 3,3 *V* z vývojové platformy FRDM-K64F. Dále bylo potřeba připojit hodiny (CLK), tedy

vzorkovací frekvence mikrofonu. Hodiny jsou použity z hodin (SCK) komunikace I^2S, více informací níže v kapitole I^2S, které ale musejí být v intervalu 1,00 – 3,25 *MHz*. Výběr kanálu byl volitelný vzhledem k tomu, že používám pouze jeden mikrofon. Proto jsem nastavení kanálu připojil na zem (GND) což znamená, že data jsou k dispozici na sestupné hraně hodinového signálu. Takto zapojený mikrofon posílá nepřetržitý proud dat ve formátu PDM signálu.

V této části jsem se především zaměřil na rešerši dostupných MEMS mikrofonů, uvedl jsem jednotlivé typy a vysvětlil jsem principy výroby i funkčnosti, viz teorie 2.1.

8.2 Nastavení vývojové platformy FRDM-K64F

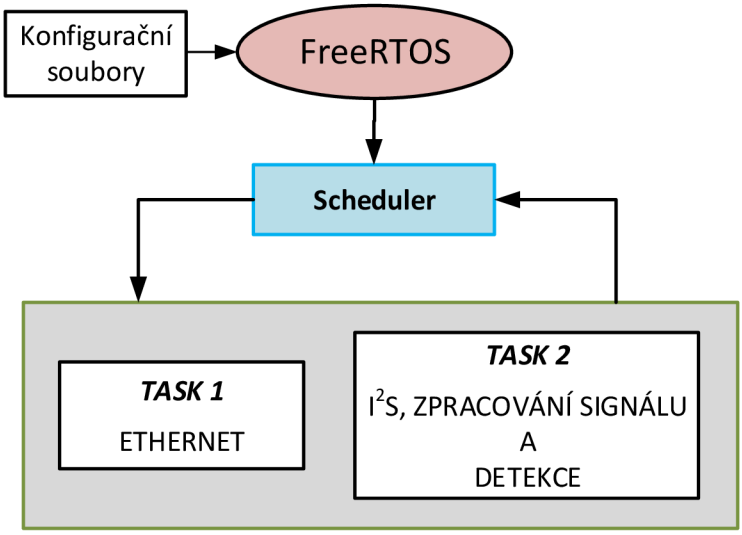
Vývojovou platformu jsem programoval ve vývojovém prostředí KDS IDE, které jsem musel nejdříve správně nastavit, například symboly preprocesoru, nastavení builderu, optimalizace, include a další. Pro jednodušší nastavení MCU jsem používal knihovny KSDK, které obsahují velké množství již vypracovaných funkcí pro obsluhu periferií na FRDM-K64F. Bohužel, možná vzhledem k nešťastné době pro firmu Freescale, kdy byla odkoupena NXP, byly veškeré pomocné nástroje jako je KSDK nebo PEx takřka nepoužitelné. Tyto nástroje samy pracovaly, ale při spojování s dalšími částmi nastaly problémy. Pro představu, nastavil jsem I2S pro MCM a vše fungovalo v pořádku. Po nastavení ethernetu a jeho zprovoznění nastal problém s posíláním dat od I2S do DMA a obráceně, samotný Ethernet pracoval v pořádku. Na odladění jsem strávil hodně času a při opakovaném selhávání jsem se rozhodl vytvořit si vlastní knihovny pro nastavení vnitřních periferií od I2S, DMA až po funkce nastavení portů, složené přímo z vnitřních registrů MCU.

V polovině února tohoto roku vydala firma NXP novou verzi KSDK v2, která je podle recenzí velice dobrá a odladěná. Také je k ní podrobnější dokumentace, která tolik chyběla u předchozí verze.

8.3 Řídící algoritmus

Nejdříve jsem pracoval bez OS pouze na „bare metal“ („holém železe“), super smyčka řízená pomocí přerušení, ale v okamžiku kdy jsem přidal ethernetový interface, bylo potřeba začít přemýšlet o OS. Řízení systému spočívá v rozdělení hlavních částí do vlastních vláken (tasků), viz blokové schéma řešení Obr. 43.

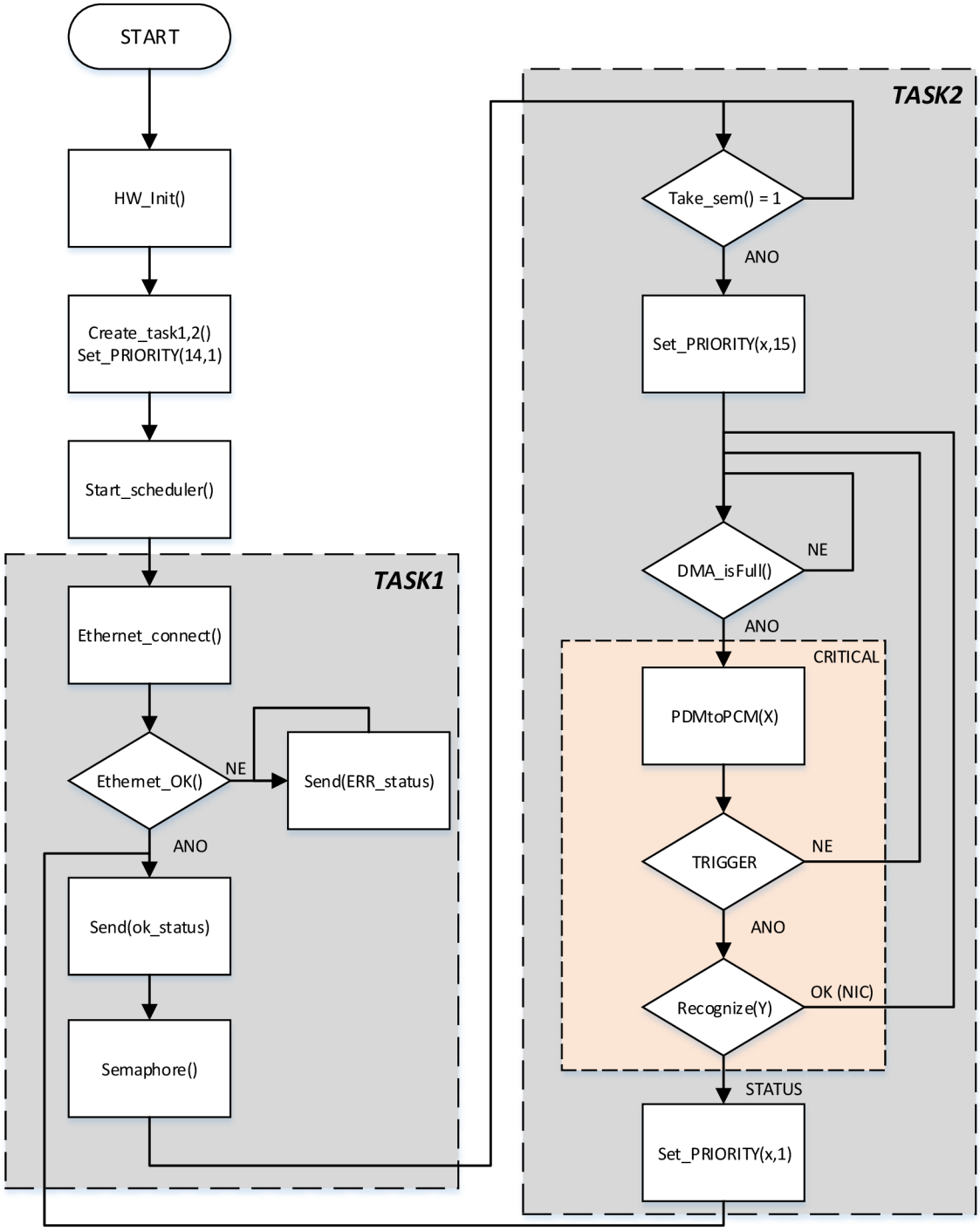
Prakticky jsem vyzkoušel dva operační systémy reálného času (RTOS). První RTOS byl přímo od mateřské firmy Freescale, s názvem Freescale (NXP) MQX RTOS. Druhý open-source RTOS, který jsem vyzkoušel, se nazývá FreeRTOS. Ten byl v nezávislých testech většiny dostupných RTOS pro MCU hodnocen lépe než ostatní dostupné. Po praktické zkušenosti se mi s FreeRTOS pracovalo taktéž lépe než s MQX RTOS a proto jsem ho zvolil pro realizaci.



Obr. 43: Blokové uspořádání řízení programu v MCU pomocí FreeRTOS

Jak je vidět na Obr. 43, systém je rozdělen do dvou tasků. O přepínání a řízení se stará scheduler. Druhotné přepínání mezi tasky je řízeno pomocí semaforu, protože dokud nenastane událost vhodná pro odeslání po Ethernetu, algoritmus je pouze v *tasku2*. Řídící algoritmus systému je zobrazen na Obr. 44.

Algoritmus začíná inicializací HW periferií MCU, což jsou hodiny (clock), porty, sériová komunikace, DMA, I2S a Ethernet. Dále jsou vytvořeny dva tasky s různou prioritou a spustí se scheduler. Priority jsou zde rozdílné proto, aby se po startu systému spustil *task1* (ethernetový task) a poslal počáteční stav vytvoření spojení. Pokud spojení neproběhne v pořádku, *task2* se nespustí. První zpráva po spuštění označuje stav o inicializaci interface. Když vše proběhne v pořádku, nastaví se binární semafor, který povoluje *task2*. Změní se priorita *task2* na prioritu vyšší než *task1* a tím je zaručeno, že se tasky nebudou mezi sebou přepínat. Semafor je zde proto, že není povoleno měnit prioritu tasku z jiného tasku. Od této chvíle je systém pouze ve smyčce *task2*, kde se provádějí operace se signálem do té doby, než je vyvolán pozitivní status od příchozího zvuku. Smyčka je navržena jako kontinuální, to vyplývá z postupu zpracování dat. Nejdříve se musí počkat, až se naplní DMA blok dat a až je plný, data se uloží do paměti a lze pokračovat ve zpracování. DMA blok dat už se opět plní od I2S sám na pozadí. Poté následuje převod z PDM na PCM signál pomocí decimačního filtru, následuje vytvoření příznaků pro klasifikační model, vyhodnocení příznaků klasifikačním modelem a přiřazení výstupu z modelu do statusu. Není-li status pozitivní, smyčka se vrátí zpět k předání dat z DMA. Pokud je ale status pozitivní, změní se priorita *task2* na nižší, tím scheduler přepne na *task1* a status se odešle po Ethernetu. Část *task2* je označena jako kritická část. Takto se označují ve FreeRTOSu části, kde je potřeba delší výpočetní čas. K tomu jsem ještě tuto část uzamkl a ostatní tasky jsem pozastavil, aby nedocházelo k nedovolenému přepnutí mezi tasky. Kdyby se tasky přepínaly, znamenalo by to ukládání velkého počtu dynamicky alokovaných dat na zásobník, které by způsobilo kolizi celého RTOS. Jednotlivé funkce algoritmu jsou podrobně rozebrány níže.

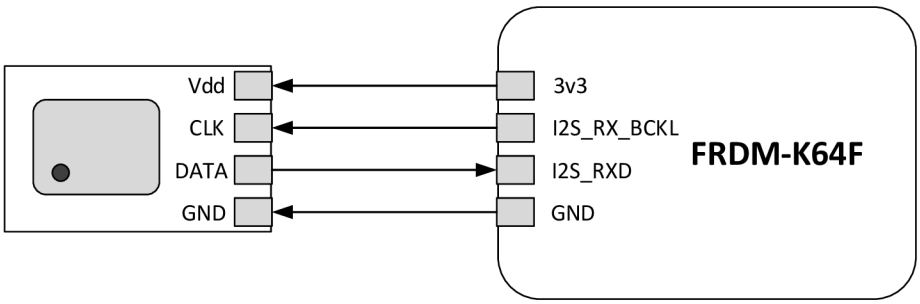


Obr. 44: Vývojový diagram systému

Ke správě FreeRTOSu jsem používal nástroj SAVERTOS, který se implementuje do vývojového prostředí (KDS IDE) a dokáže zobrazovat informace o stavu RTOSu. Například kolik je vytvořeno tasků, v jakém jsou stavu, kolik místa zabírají na zásobníku a další. Bohužel, i když je FreeRTOS zdarma, jak už název napovídá, většina podrobné dokumentace je placená, například referenční manuál stojí 30$.

8.4 I²S

Pro přenos signálu mezi mikrofonem a vývojovou platformou FRDM-K64F jsem použil I²S interface. Výhodou je, že FRDM-K64F pro tento interface podporuje přímý přístup do paměti přes DMA, které zajistí minimální zatížení procesoru při přijímání dat.



Obr. 45: Zapojení MCM s FRDM-K64F pomocí I2S interface

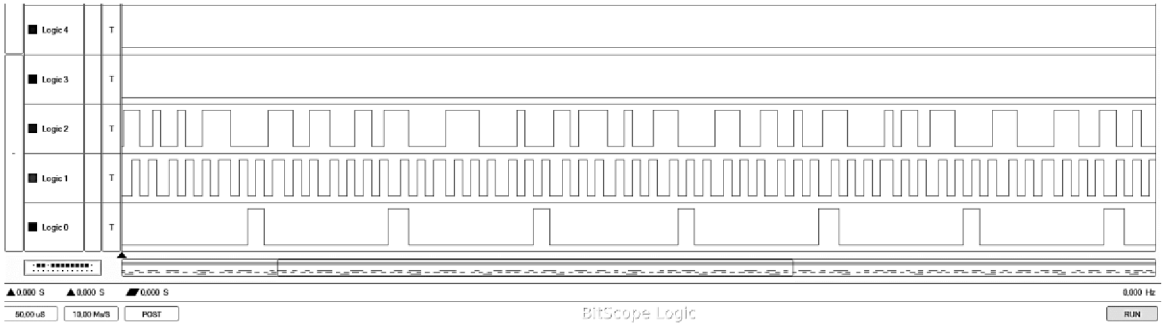
Bez této podpory by se interface nedal použít pro tento systém. Pro obsluhu I²S interface jsem vytvořil vlastní knihovnu, která pracuje s vnitřními registry MCU.

Nejprve jsem vybral a nastavil vhodný režim komunikace pro I²S interface. Vzhledem k nepřetržitému posílání dat z mikrofonu jsem vybral režim DPM. Jako zdroj obou hodin jsem zvolil hodiny z FRDM-K64F. MCM jsem s I²S interface ve FRDM-K64F zapojil podle zapojení na Obr. 45. Ovládací signál WS jsem nastavil na velikost 8 *bitů*, tento signál je využíván pouze pro vnitřní funkce I²S interface, takže není připojen k MCM. FRDM-K64F umožňuje oddělené nastavení hlavních hodin jak pro příjem, tak pro vysílání, proto název pinu hodin na platformě zahrnuje zkratku RX. Pro můj systém I²S interface ve FRDM-K64F využívám jen jako přijímač. Pro nastavení finální verze systému jsem I²S interface nastavoval následovně:

- frekvence hlavních hodin: 1,38 *MHz*
- velikost slova: 8 *bitů*
- významnost 0 bitu: LSB
- výstupní datový typ: unsigned 8 *bits*
- reakce na hranu hodin: sestupná
- Polarita WS signálu: 1
- velikost WS: 1 *bit*
- synchronizace WS: 1 *bit data*
- DMA kanál: 0

a mapování výstupních pinů, označených na Obr. 45. Frekvenci hlavních hodin jsem zvolil 1,38 *MHz*. Jak už jsem zmínil, frekvence vstupních hodin mikrofonu má určitý interval. Hodnotu jsem volil s ohledem na velikost snímaného okna, aby dokázalo snímat zvuky v širokém frekvenčním spektru, viz kapitola Decimační filtr. Obvykle se tato hodnota volí tak, aby byla násobkem některé z typických hodnot vzorkovací frekvence zvukových zařízeních. Tento fakt jsem však nebral při výběru v úvahu vzhledem k dostačující nižší

vzorkovací frekvenci, která je používána. Reakce na hranu hodin znamená vyčítání hodnoty z datového signálu podle reakce na nástupnou nebo sestupnou hranu hlavních hodin. Výstupní datový typ je typ, pod kterým je výstupní číslo z komunikace uloženo do paměti. To znamená, že v mém případě od I²S interface přijímám a do paměti ukládám 8bitové číslo, které je složeno z 8 *bitů* PDM signálu.

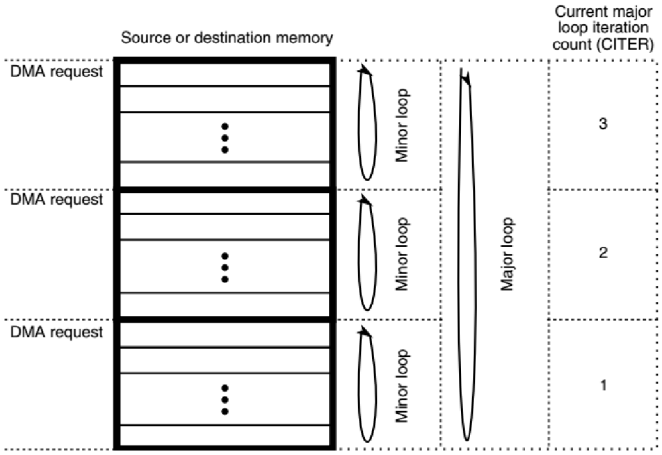


Obr. 46: Průběh I²S komunikace snímané logickým analyzátorem

Průběhy komunikace jsem zaznamenal pomocí logického analyzátoru a jsou zobrazeny na Obr. 46. Na průběhu kanálu 0 je vidět průběh nastavovacího signálu WS, který je nastaven na velikost 8 *bitů*. Na kanále 1 je průběh hodin SCK a na kanále 2 jsou vidět samotná data z mikrofonu.

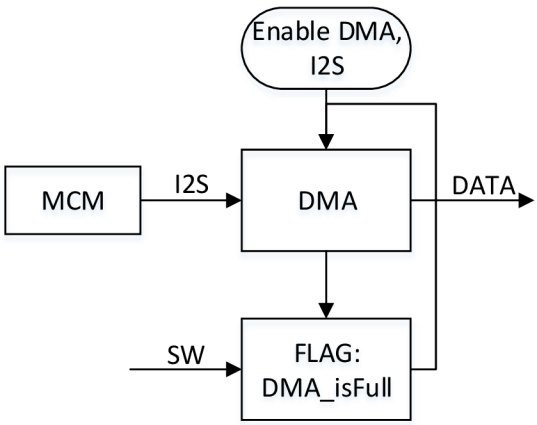
8.5 DMA

Pro nastavení DMA jsem vytvořil, jako u ostatních používaných periferií, vlastní knihovnu, která inicializuje a obsluhuje DMA. V inicializační rutině nastavuji DMAMUX, který určuje, z jakého zdroje mají být vyčítána data (RX data od I²S), povolení příslušného kanálu, nastavení velikosti posuvu adresy zdroje, velikost posuvu v datovém poli, velikost datového pole, počet operací v minor smyčky, velikost major smyčky a posun ukazatele na vstupní na počátek datového bloku po dokončení DMA.



Obr. 47: Znázornění průběhu řídících smyček ukládaní DMA [19]

Pro čtení a ukládaní dat z DMA slouží funkce *getData()*, ve které je umístěna funkce *DMAisFull()*, která vrací informaci o tom, že DMA dokončilo plnění dalšího bloku dat. Pokud je tato funkce zavolána, program v ní čeká do té doby, než jsou data uvolněna, tato vlastnost s vysvětlením je popsána v kapitole 8.3. Automatická obsluha příchozích dat po I^2S interface je zobrazena Obr. 48.



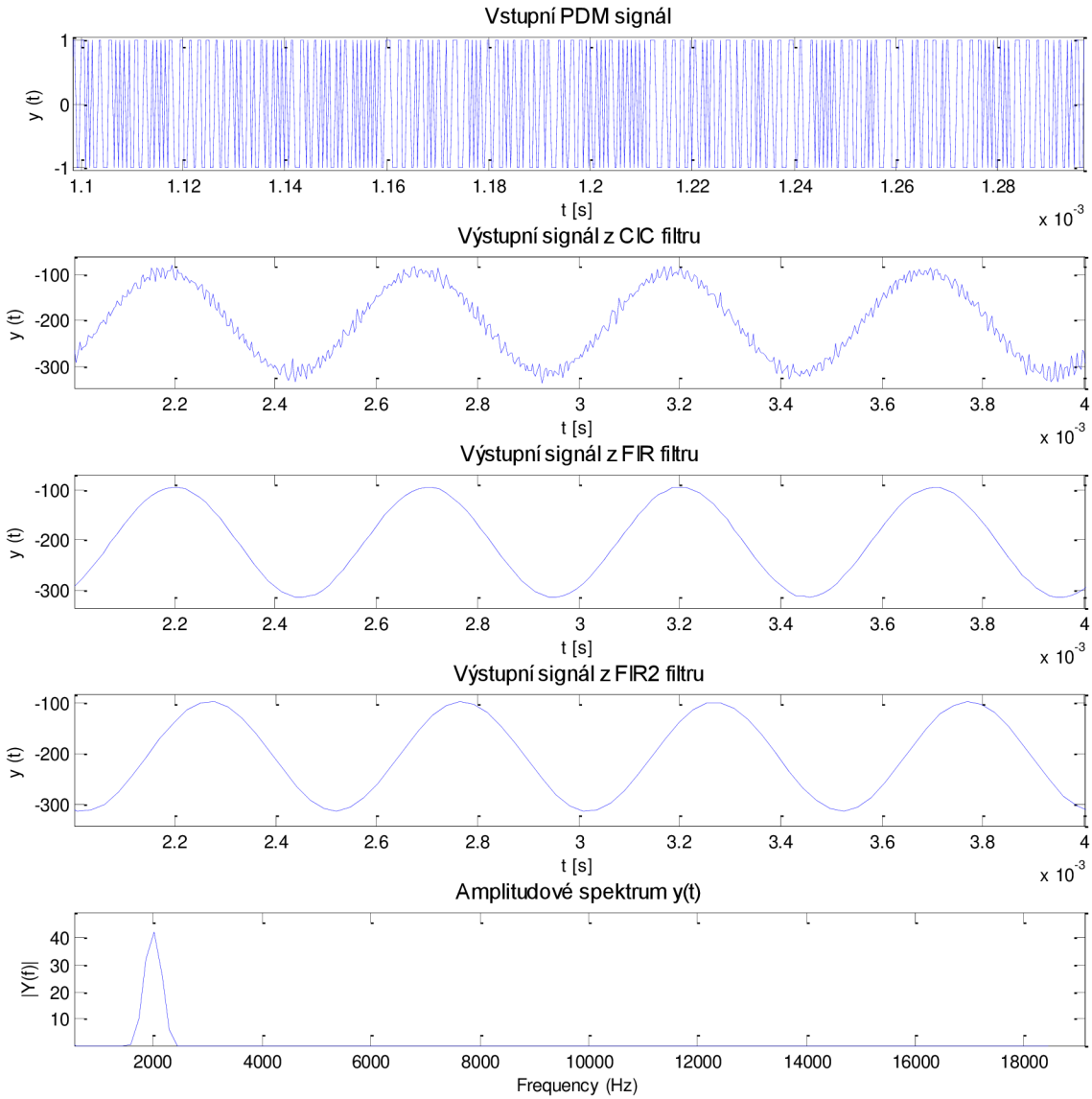
Obr. 48: Blokové schéma automatické obsluhy příchozích dat po I2S interface

Na Obr. 48 je vidět, že po inicializaci DMA a I2S je povolen řídící DMA kontroler, který řídí obsluhu příchozích dat po několika vstupních kanálech. Pokud je DMA blok dat naplněn, kontroler vygeneruje flag o dokončení a uvolní data pro čtení. SW představuje funkci *DMAisFull()*, která se dotazuje na stav DMA.

8.6 Decimační filtr

Decimační filtr jsem navrhoval v prostředí MATLAB z důvodu rychlé realizace a snadného zobrazení výsledků. Decimační filtr jsem navrhoval na reálných datech z MCM. Provedl jsem to tak, že data příchozí po I^2S interface jsem ukládal do paměti FRDM-K64F a ty jsem následně posílal po sériové komunikaci do počítače, kde jsem je zpracovával. Jelikož jsem měl data v datovém formátu unsigned 8 *bits* (nastavení I2S), hodnota čísla 0-255, musel jsem nejprve číslo převést zpět na PDM formát. To jsem prováděl bitovým posuvem a jednotlivé bity jsem ukládal do bitového pole.
Jako zdroj zvuku jsem používal akustický generátor signálu, abych si výsledek ověřil v amplitudovém spektru. Průběh PDM signálu o vzorkovací frekvenci 2,3 *MHz* je zobrazen na Obr. 49 v jiném měřítku proti ostatním průběhům. Vzhledem k velké hustotě zobrazených dat je zde graf pouze ilustrativní pro představu vstupního signálu.

Navržený decimační filtr jsem navrhoval tak, aby byl co nejefektivnější a co nejméně výpočetně náročný. Proto jsem navrhl pouze tříblokový decimační filtr. První blok je CIC filtr třetího řádu s decimačním faktorem 8 a diferenciální zpožděním 1. Druhý blok je FIR filtr 20 řádu s decimačním faktorem 4 a třetí blok je FIR filtr 20 řádu s decimačním faktorem 2.



Obr. 49: Výstupní signály z decimačního filtru navrženého v MATLABu (sinusový signál 2 kHz)

Na Obr. 49 jsou zobrazeny jednotlivé průběhy z částí navrženého decimačního filtru. Vstupním signálem je PDM signál z MCM o velikosti -1 a 1. Zdrojem zvuku je signál z akustického generátoru o frekvenci 2 *kHz*. Druhý signál výstupu z CIC filtru je stále zašuměný, vzorkovací frekvence tohoto signálu je 288 *kHz*. Třetí signál patří výstupu z prvního FIR filtru, který má decimační faktor 4, tento signál už je vyhlazený bez šumu, stále má vysokou vzorkovací frekvenci, takže na čtvrtém obrázku je výstupní signál z druhého FIR filtru s decimačním faktorem 2 a vzorkovací frekvencí 35,9 *kHz*. Výhoda těchto decimací je v tom, že signál obsahuje pouze frekvence o velikosti výsledku decimace. Takže ostatní vyšší frekvence jsou ze signálu odstraněny. Výstupní signál je v MCU ještě normalizován, zde je pro názornost výsledných hodnot nenormalizovaný.

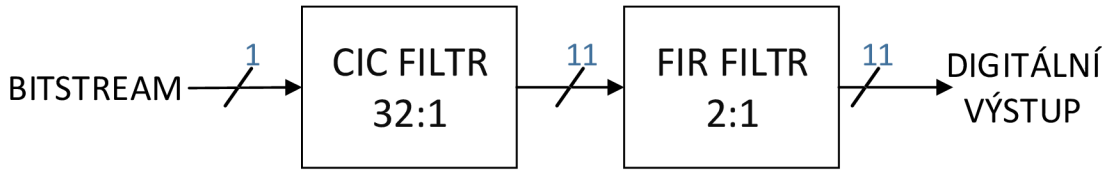
Výsledné rozlišení výstupního signálu je dáno vztahem (7) a pro takto navržený decimační filtr je výsledné rozlišení výstupního PCM signálu 19 *bitů*. Vzorkovací

frekvence je rovna podílu původní frekvence PDM signálu 2,3 *MHz* a celkovému decimačnímu faktoru 64, takže nová vzorkovací frekvence je 35,9 *kHz*.

Na posledním průběhu Obr. 49 je zobrazena amplitudová charakteristika výstupního signálu z decimačního filtru pro ověření věrohodnosti převodu z PDM do PCM formátu signálu. Z amplitudové charakteristiky je vidět, že konverze proběhla bez změny frekvence signálu. Amplitudová charakteristika je získána pomocí FFT, kde vstupní signál je výstupní signál z decimačního filtru upravený blackmanovým oknem. Nízké frekvence nejsou ve spektru zobrazeny, protože signál obsahuje stejnosměrnou složku. Ta je způsobena necelou periodicitou vstupního signálu PDM, který je v rozsahu (-1 - 1). Pro odstranění této složky je následně prováděna normalizace signálu v rozsahu (-1 - 1). Zesílení výstupní signálu vychází ze vztahu zesílením CIC filtru (8).

8.6.1 Optimalizace pro platformu

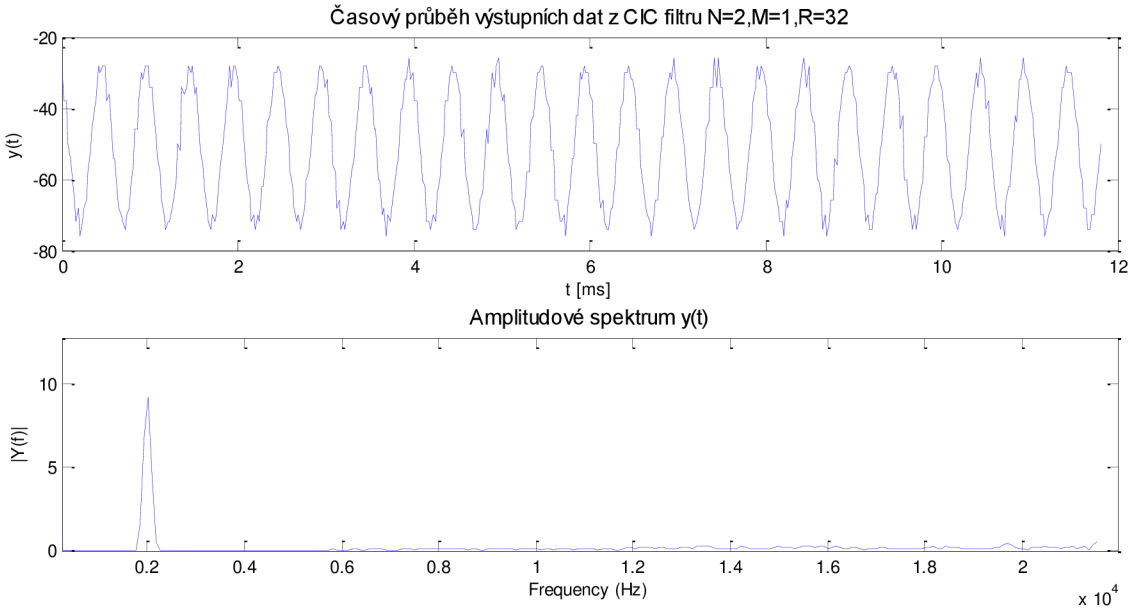
I přes dodržení efektivity a nenáročnosti výpočtu byl návrh CIC filtru v MATLABu pro platformu velice optimistický. Problém nastal ve velikosti dat, společně s operačním systémem, respektive velikost dat pro jeden task. Proto jsem musel přistoupit k optimalizaci původního návrhu decimačního filtru tak, abych zachoval jeho funkčnost. To znamená splnění hlavních dvou podmínek - co nejméně výpočetně náročný a co nejvíce funkčně výkonný.



Obr. 50: Blokové schéma navrženého decimačního filtru pro platformu

Při optimalizaci jsem postupoval následovně. Nejprve jsem původní návrh přeprogramoval do platformy, zde jsem zjistil, že velikost dat je moc velká. Postupně jsem tedy snižoval parametry a sledoval funkčnost filtru. Prvním problém nastal, když jsem nastavil velikost DMA na 2048 8bitových prvků, to je vlastně bistream zakódovaný na 8bitová čísla pro posílání po I2S interface. Při rozložení na bitstream vznikne pole dat o velikosti 2048x8 = 16384. Abych ušetřil paměť výsledky výpočtu integrálů, první část CIC filtru, jsem přepisoval zpátky do tohoto pole. Další problém souvisí s řádem používaného CIC filtru, protože s řádem roste počet sériově spojených integrátorů a tím roste velikost výsledku. Proto bych musel alokovat původní pole dat na 4 bytové hodnoty a v součtu by tato část filtru zabrala v paměti 65,5 kB. Tento fakt jsem vyřešil tak, že jsem bitstreamu hodnotě 0 přiřadil -1, tím se velikost výsledků zredukovala. Snížil jsem řád filtru na dva a efektivně jsem navrhl výpočet integrační části CIC filtru tak, aby se počítal pouze z aktuálních dat a neukládal veliké množství dat. Také jsem snížil vzorkovací frekvenci MCM, díky které jsem zvětšil časový interval snímaného úseku při stejném počtu dat. Decimaci jsem zvýšil na decimační faktor 32. Díky tomu jsem snížil velikost pole dat, které přenáším do FIR části, kde pracuji s datovým typem float. Diferenciální

zpoždění jsem zvolil jedna, to má vliv na výsledném bitové rozlišení (7), které je i tak dostačující.



Obr. 51: Časový průběh a amplitudové spektrum výstupních dat CIC filtru z platformy (sinusový signál 2 kHz)

V optimalizaci jsem pokračoval i v derivační části CIC filtru, zde jsem opět pracoval s původním polem a prováděl na něm dvojitou derivaci v jednom cyklu. Touto úpravou posouvání dat a výpočtem pouze v jednom cyklu přijdu o 2 vzorky (N*M), ale ušetří se opět paměť.

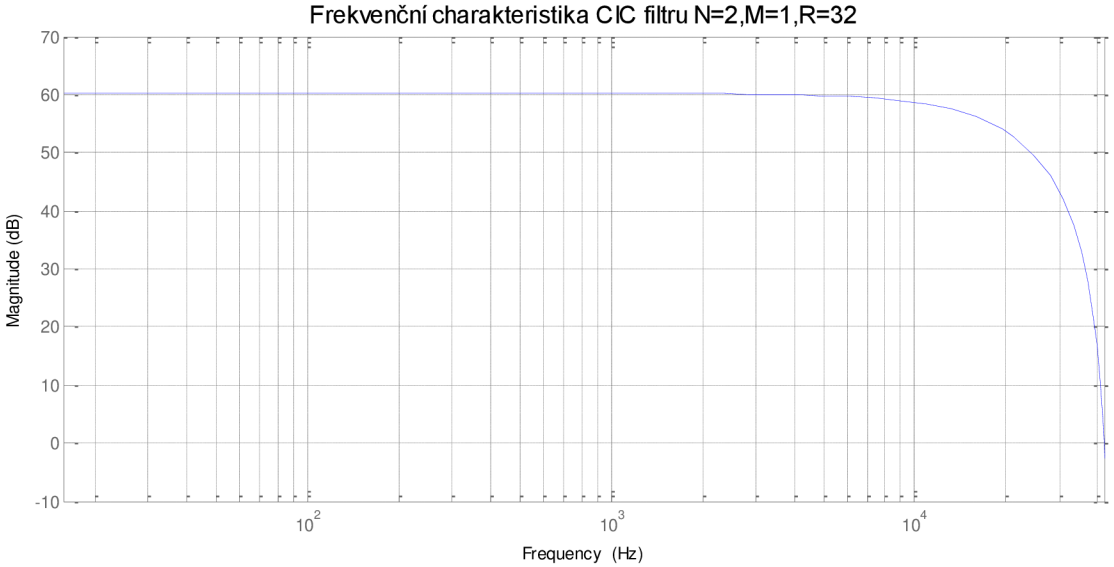
Pro zobrazení výsledků jsem opět využil MATLAB, podle kterých jsem provedl finální úpravy. Při výsledném návrhu CIC filtru jsem uvažoval v následovně.

Maximální frekvence snímaného zvuku se bude pohybovat v rozmezí od 200 *Hz* – 8000 *kHz*, jelikož půjde o varovné zvuky (alarmy, tříštění skla, rány a násilného dobývání). Alarmy jsou vždy konstruovány tak, aby měly co nejvýraznější a nejpronikavější zvuk pro lidské ucho. Tento interval tedy vychází z křivky slyšitelnosti lidského ucha, která pro tento interval dosahuje nejnižšího prahu slyšitelnosti. Nejcitlivější frekvence pro lidské ucho je interval mezi 3,5 – 4 *kHz* což je rezonanční kmitočet zvukovodu. Z těchto teoretických znalostí a později i z praktických jsem tedy vybral jako maximální snímanou frekvenci i rezervou na 8 *kHz*. Na hranici této frekvence už působí mírná nelinearita frekvenční charakteristiky samotného mikrofonu a je zde hodnota zesílena 4*dB*, viz Obr. 42. Rezervou myslím potlačení hraničních aliasingových vlastností.

Z takto určené maximální frekvence jsem vycházel pro stanovení celkového decimačního faktoru. Vzorkovací frekvenci signálu, které odpovídají hlavní hodiny I2S interface, jsem zvolil 1,38 *MHz*. Z frekvence vychází decimační faktor jako poměr vzorkovací frekvence PDM signálu a dvojnásobek maximální frekvence nebo vzorkovací frekvence výstupního signálu, viz rovnice (20).

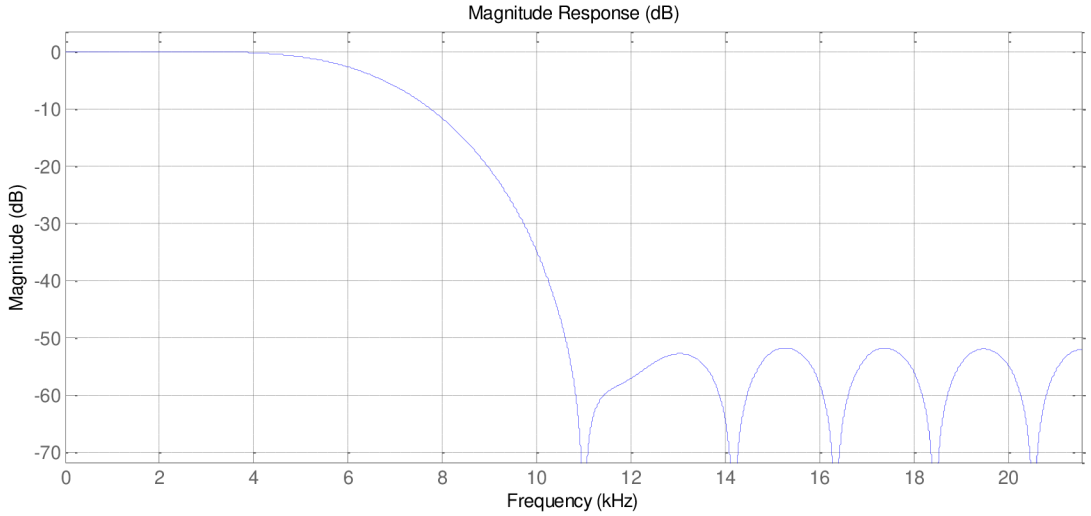
$$R = \frac{f_{s\_PDM}}{2 \cdot f_{max}} = \frac{1{,}38 \cdot 10^6}{2(8 \cdot 10^3)} \cong 86\ [-] \qquad (20)$$

S uvážením strmosti frekvenční charakteristiky následného FIR filtru jsem přidal další rezervu. Aby hodnota decimačního faktoru odpovídala mocnině dvou, zvolil jsem celkový decimační faktor R = 64. Tomuto faktoru odpovídá pro N = 2048 vzorků časový interval snímaného signálu 11 *ms*, viz Obr. 51. Podle decimačního faktoru vychází maximální snímaná frekvence 11 *kHz* a přitom splňuje vzorkovací teorém, který odpovídá vzorkovací frekvenci 22 *kHz*. Díky takto zvolené rezervě nemusí být výsledný FIR filtr tak vysokého řádu, vysoký počet koeficientů, s kterými roste výpočetní výkon, tedy velký sklon frekvenční charakteristiky v oblasti frekvence řezu.



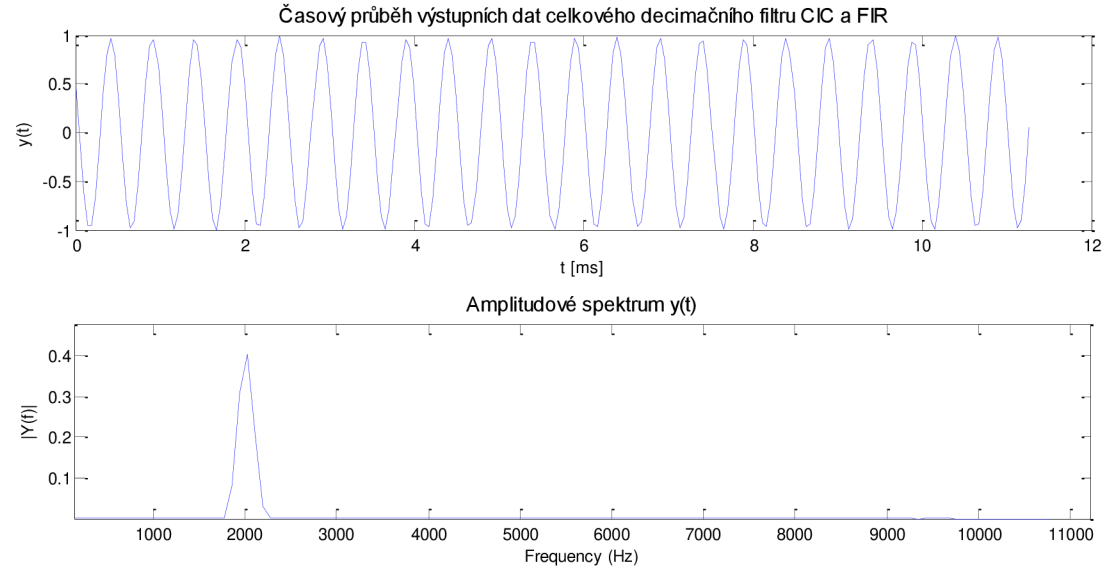
Obr. 52: Frekvenční charakteristika navrženého CIC filtru

Celkový decimační faktor jsem rozdělil na dva a signál jsem decimoval ve dvou částech decimačního filtru. První decimaci jsem provedl v CIC filtru, kde jsem zvolil decimační faktor 32. Tato volba je postavena na předchozí myšlence o řádu FIR filtru a na velikosti předávaného datového pole pro následující FIR filtr. Proto jsem zvolil tak vysoký decimační faktor. Velikost faktoru jsem ověřil i praktickým srovnáním. Pro zvýšení efektivity decimačního filtru jsem navrhl CIC filtr 2. řádu. Diferenční zpoždění filtru jsem podle výsledků zvolil 1, časový průběh takto navrženého CIC filtru je Obr. 51. Výsledné zesílení CIC filtru je podle vtahu (8) G = 1024 a bitové rozlišení převodu je 11 *bitů*. Frekvenční charakteristika filtru Obr. 52.



Obr. 53: Frekvenční charakteristika FIR filtru s frekvencí řezu 7 kHz

Při návrhu FIR filtru jsem postupoval podle teoretického popisu, viz 3.3.3.2. Frekvenci řezu jsem zvolil 7 *kHz*, i když je to o 1 *kHz* méně než moje stanovená maximální frekvence. Důvod je takový, že hodnota zesílení samotného mikrofonu je na frekvenci 8 *kHz* +4*dB* a dál roste. Abych dokázal vyšší frekvence dostatečně filtrovat, zvolil jsem frekvenci řezu na 7 *kHz*. Jak je následně vidět na frekvenční charakteristice filtru Obr. 53. Filtr je navržen tak aby byl výpočetně nenáročný ale účinný, proto jsem zvolil FIR filtr 20 řád. V bodě frekvence řezu je úbytek -6*dB* a v bodě 8 *kHz* je -11*dB*.



Obr. 54: Výstupní signál z optimalizovaného CIC filtru a FIR filtru v časové a amplitudové oblasti (2kHz)

Na rozdíl od CIC filtru, který nemá žádné koeficienty, respektive má koeficienty rovny jedné, při implementaci do vývojové platformy nepřenáší žádné konstanty. Konstanty se do platformy přenášejí u FIR filtru, který má koeficienty různé od jedné a je tedy nutné tyto koeficienty vypočítat.

Koeficienty jsem získal z návrhu filtru v MATLABu, v nástroji *vftools* a předal jsem je do platformy v podobě pole konstant uložených v paměti. Koeficienty jsou pro

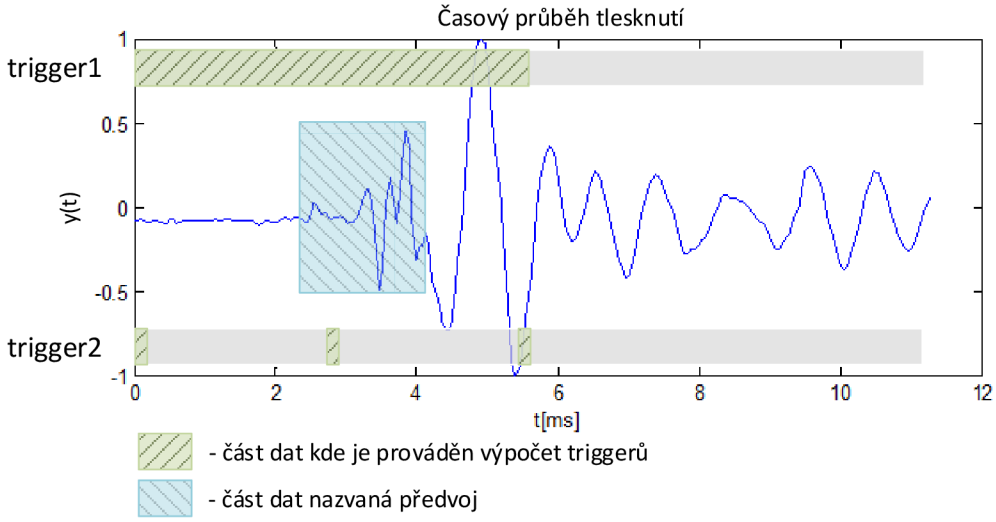
navržený filtr statické, proto je možné takto postupovat. Výpočet koeficientů je výpočetně náročný a bylo by neefektivní a zbytečné počítat koeficienty stále dokola až v platformě. Poté pomocí konvoluce koeficienty spojím s vstupním signálem a provedu tak filtraci. Decimace signálu decimačním faktorem 2 je provedena až na signále vyfiltrovaném FIR filtrem.

Tímto postupem jsem se řídil při návrhu optimalizovaného decimačního filtru. V testovacím programu vycházelo toto nastavení jako nejlepší volba mezi efektivitou a výpočetní náročností. Takto navržený decimační filtr jsem implementoval do vývojové platformy a otestoval jeho správnou funkčnost. Výsledný výstup z decimačního filtru je zobrazen na Obr. 54, kde vstupním signálem je generovaný akustický signál o frekvenci 2 *kHz*. Jak si můžete všimnout, zobrazený signál je již normalizovaný. Normalizace se provádí jako poslední krok části úpravy signálu. Takto převedený a filtrovaný signál už pokračuje do další části systému, jako je zpracování a tvorba příznaků pro model.

V MCU se o tento převod stará funkce *pdm2pcm()*, která má v sobě zahrnuty všechny výše popisované části. Kvůli minimalizaci výpočetních a paměťových nároků jsou dílčí funkce zařazeny do hlavního cyklu. Vstupním parametrem funkce je ukazatel na pole výstupních dat z DMA a funkce vrací ukazatel na pole převedených dat.

8.7 Realizace triggeru

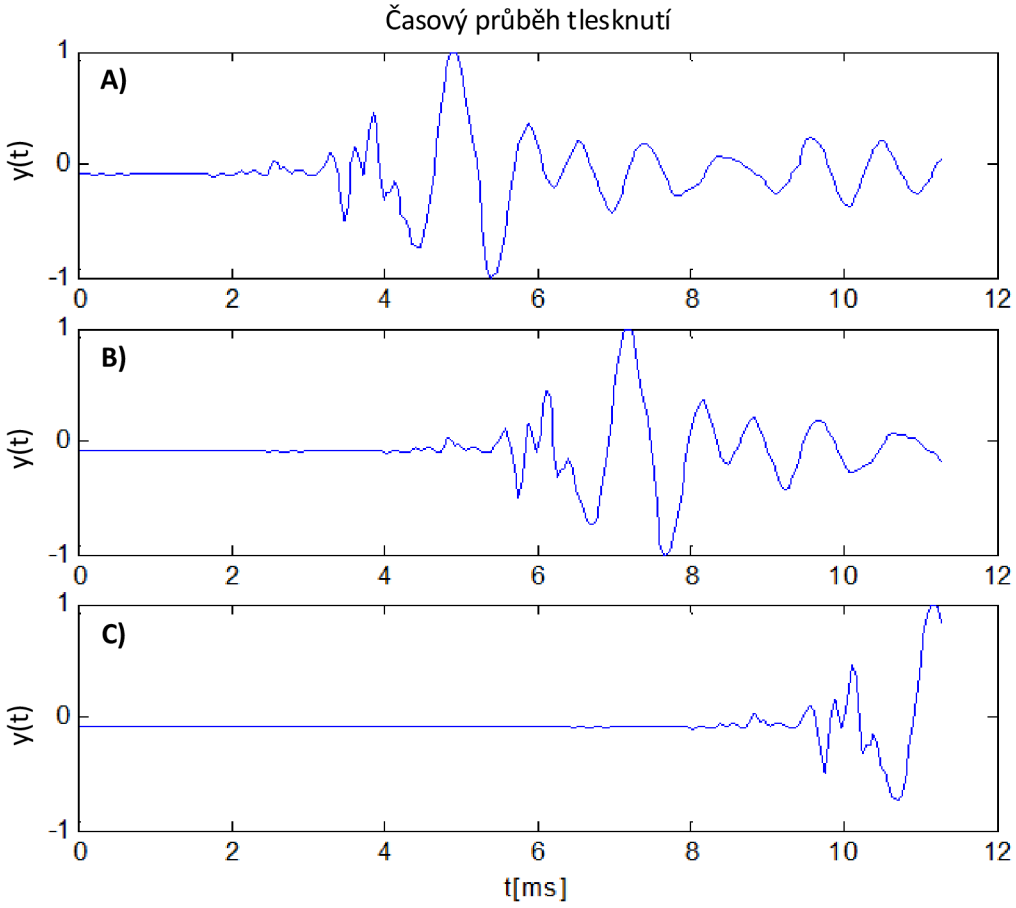
Trigger slouží pro spouštění dalších částí systému podle určité intenzity zvuku. Kvůli digitálnímu mikrofonu byla realizace triggeru složitější v porovnání s analogovým mikrofonem, kde lze sledovat úrovně signálu okamžitě vzorek po vzorku. U digitálního mikrofonu se trigger dá zjistit až po nasbírání bloku dat, viz DMA část. Tento blok dat se převádí z PDM signálu na PCM pomocí CIC filtru, kde signál nejprve integrován pak decimován a nakonec derivován. Do tohoto procesu jsem zařadil i výpočty obou triggerů.



Obr. 55: Časový průběh tlesknutí s označením částí využívaných pro triggery

První trigger je vložen do integrační části filtru, kde je druhý integrál PDM signálu v absolutní hodnotě znovu integrován, aby byla získána citlivější odezva na změnu. První

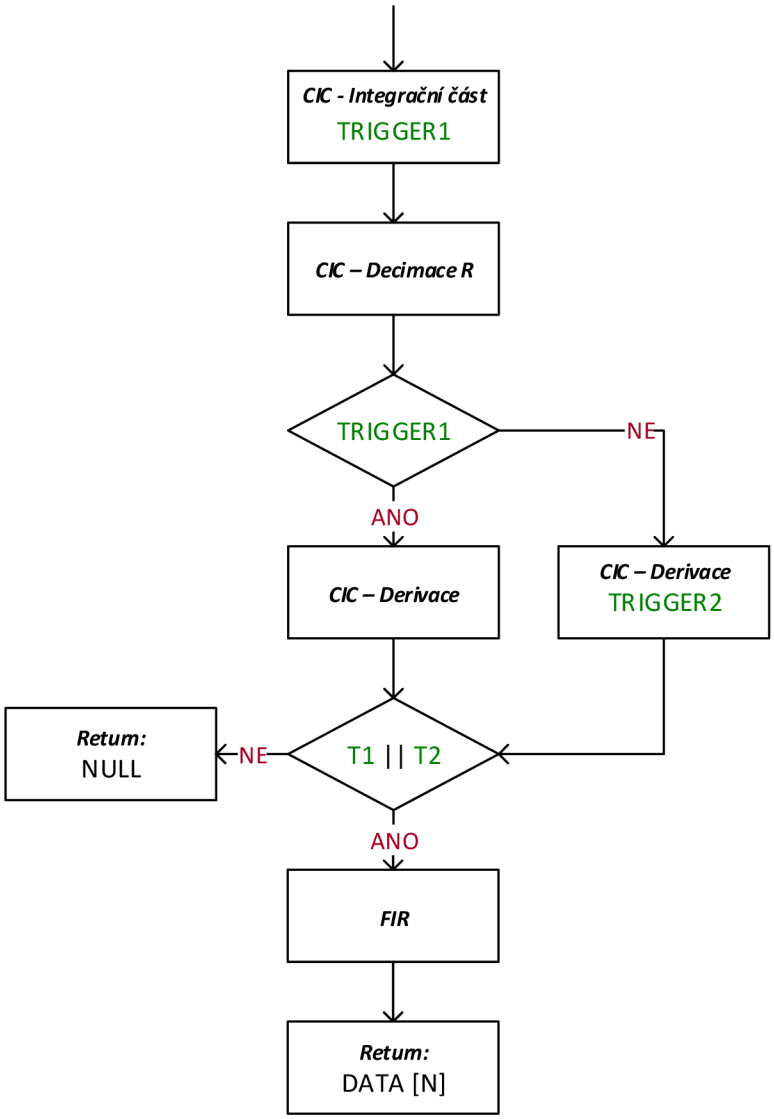
trigger je aktivován v závislosti dvou spouštěčů. První spouštěč porovnává výsledek integrálu z předchozího cyklu s aktuální hodnotou, pokud je rozdíl mimo dané meze, tak je trigger aktivován. Druhý spouštěč je absolutní, pokud aktuální hodnota integrálu překročí určenou hodnotu meze, trigger spustí. Spouštěče nejsou na sobě závislé, pokud je jeden z nich vyhodnocen, trigger je aktivován. Tento trigger je první ze dvou triggerů a je určen pro nízké frekvence o vysoké amplitudě, které jsou tvořeny v rázy a rány, podle toho jsem ho nazval rázový trigger. Jelikož je tento trigger na začátku celého zpracování signálu, je mu udělena priorita hlavního triggeru. Pokud spustí, výpočet druhého triggeru se už neprovádí. Výhoda tohoto triggru spočívá v tom, že je součástí CIC filtru a dvě třetiny výpočtu triggeru se provádí v něm.



Obr. 56: Časové průběhy s různým spouštěním triggerů: A) reakce ve středu poloviny datového bloku, B) reakce na konci datového bloku, C) ukázková reakce v druhé polovině datového bloku

Druhý trigger je určen pro vysoké frekvence a periodické signály způsobené například alarmy. Trigger funguje tak, že provádím tzv. sondy v první polovině bloku dat, viz Obr. 55. Sonda nasbírá pouze jednotky vzorků, ty integruje a násobí je konstantou v řádu 10 až 100, záleží na nastavení citlivosti. Výsledek sondy je porovnán s referenční hodnotou, která je dána směrnicí násobenou stejnou konstantou. Směrnice byla získaná z opakovaného měření klidového pozadí. Pokud je rozdíl mimo dané meze, trigger spustí. Tento trigger má tři druhy nastavení, které ovlivňuje citlivost, jako jsou meze, konstanta a počet vzorků. Nastavení citlivosti v mém systému je: 6 vzorků, 20 konstanta a meze od

500 do 1600. Druhý trigger, stejně jako první, využívá pro svůj výpočet cyklus CIC filtru. Tento trigger využívá derivační část filtru a jeho výpočet je zařazen do jeho části. To znamená, že se navíc neprovádí nový cyklus pro trigger, ale využívá se stávající.



Obr. 57: Blokové diagram uspořádání triggerů

Oba triggery nejsou počítány z celého nasbíraného bloku dat, to znamená, že triggery nemohou být spuštěny kdekoli z celého bloku dat. Kdyby byly počítány z celého bloku, přinášelo by to problém s nesourodostí dat v datového bloku na následné vyhodnocení signálu. Především, když je trigger spuštěn u konce bloku a využitelného signálu, zbyde na vyhodnocení pouze malá část užitečného signálu, viz Obr. 56 C). U prvního triggeru je problém vyřešen tak, že se integrace provádí pouze pro první polovinu bloku dat. Pokud by změna signálu, na kterou by měl trigger reagovat, přišla až za první polovinou, bude tato změna ignorována, viz Obr. 55. Je tomu tak, protože by s největší pravděpodobností nastala situace, viz Obr. 56 C) a užitečná data pro vyhodnocování by byly z moc krátkého úseku. Hranice je také dána malou změnou před samotným prvním nástupem tzv. předvoj, na který trigger zareaguje a tím se signál více posouvá za tuto hranici do druhé poloviny

bloku. V této situaci systém zareaguje hned na začátku dalšího bloku a má tím pádem daleko více korektních dat a bez hodnot počátečního nástupu. Tomuto řešení přispívá také fakt, že blok dat trvá 11,27 *ms* a připočítáním předvoje před výraznou změnou a pokud budeme uvažovat případ, že předvoj začíná přímo na hranici, tak přijdeme o 5,64 *ms* signálu, z nich část patří předvoji. Přijdeme také o tvar nástupu signálu, ale za cenu dostatku dat v dalším bloku, které jsou velice důležité pro periodické signály. Může se namítat, že se trigger může nastavit tak, aby nereagoval na předvoj, pak by se hranice triggeru mohla posunout více do pravé poloviny. Toto nastavení je ale dáno citlivostí. Pokud je zdroj signálu blízko modulu, intenzita předvoje bude srovnatelná se zdrojem signálu daleko, a proto trigger reaguje na předvoje těchto signálů a musí být brán v úvahu.

U druhého triggeru je řešení obdobné. Sondy se provádí pouze ze signálu z první poloviny bloku dat, viz Obr. 55, kde jsou ve spodní části zeleným šrafování znázorněny sondy druhého triggeru.

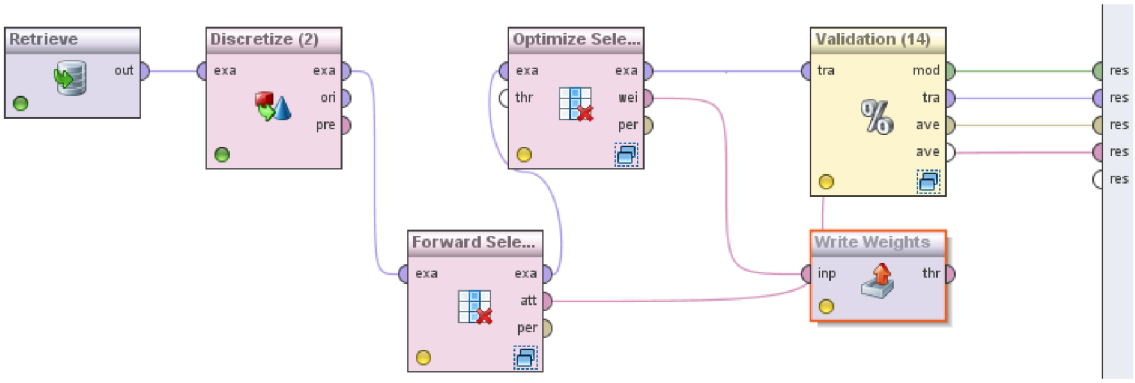
Jak už bylo zmíněno výše, triggery spouštějí další části systému. Jak je vidět na vývojovém diagramu Obr. 57, pokud je aktivován první trigger, výpočet druhého triggeru už se neprovádí, pouze se v cyklu CIC filtru vypočítá derivační část.

8.8 Rozpoznávání signálu

Nedílnou součástí systému je rozpoznávání akustického signálu. Jak jsem zmínil již v úvodu, systém je navržen jako univerzální senzor, kterému se dodají zvuky, na které má reagovat. Za pomoci nadřazeného softwaru v PC se na základě těchto dodaných zvuků navrhne (naučí) model, který se následně implementuje do MCU. Při návrhu modelu je ale potřeba brát v úvahu výpočetní výkon MCU a také paměťový prostor, takže nelze použít modely, které provádí výpočetně nebo paměťově náročné operace.

8.8.1 Trénovací data

Základním prvkem pro návrh modelu jsou trénovací data, která jsou použita pro učení modelu. Trénovací data pro model jsou vytvořena ze vstupních dat, které obsahují zaznamenané akustické signály od podmětů, které má rozpoznávat. Vstupní data jsem nahrával přímo v MCU a posílal jsem je pomocí sériové komunikace do PC, kde jsem je ukládal, podobně jako náhledy v části Decimační filtr. Data jsou spouštěna stejným triggerem a upravována stejně tak jako v MCU. Takže jsou stejná jako data, které bude systém zaznamenávat a vyhodnocovat, viz časové průběhy na Obr. 59. Snažil jsem se zachytit akustické signály z co nejvíce možných pozic a situací, a tím docílit přesnějšího a robustnějšího naučení modelu. Všechna vstupní data jsou označena hodnotou výstupní třídy (label), která udává, o jaký typ akustického zvuk se jedná. Jako vstupní data jsem nahrál 143 zvuků pro 7 typů (výstupní tříd), jejich názvy i popis naleznete v kapitole 8.8.3.



Obr. 58: Zapojení bloků při návrhu modelu v RapidMineru

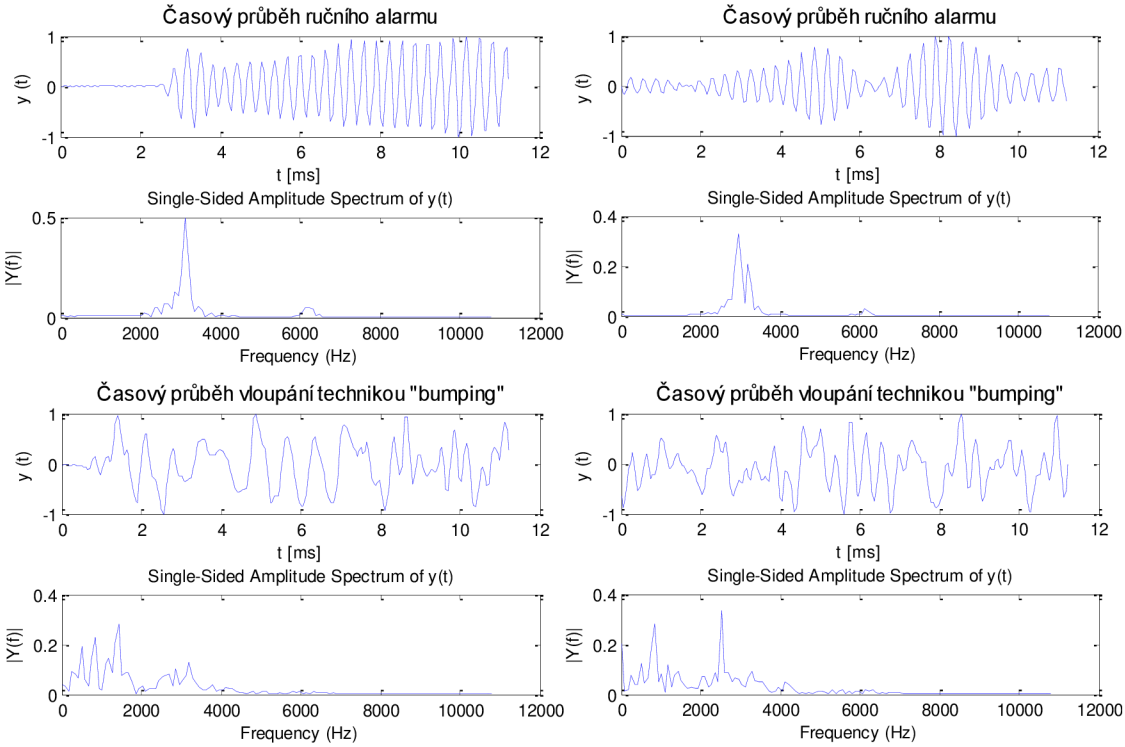
Nasnímané zvuky, na které má systém reagovat, nejsou ještě trénovací data pro učení modelu. Z dat se napřed musejí vytvořit příznaky, které popisují daný akustický signál. Jednotlivé příznaky jsou získány z různých metod zpracovaní signálu a z nich jsou vytvořeny příznakové vektory pro dané typy akustických zvuků. Jedná se o metody, které dokáží vystihnout rozdíly mezi jednotlivými zvuky. Proto je důležité rozumět daným akustickým signálům ve smyslu použít takovou metodu zpracování signálu, která dokáže najít v signálu něco jedinečného, co ho odlišuje od ostatních a jednoduše to převést do příznaků. Toto hledání je velice pracné a ve většině případech je obtížné až nemožné přijít jednoduchou dedukcí na všechny souvislosti. Proto jsem využil technik strojového učení, díky kterým lze z velkého množství dat vybrat příznaky, které jsou pro popis daného akustického signálu nejvýhodnější. To znamená příznaky, které jsou pro určitý počet výstupních tříd dostatečně odlišitelné.

8.8.2 Metody zpracování signálu

Metody zpracování signálu jsou v systému proto, aby vytvořily číselné hodnoty vlastností signálu, z kterých se vytvoří příznakový vektor. Metody musejí být stejně jako model výpočetně a paměťově nenáročné. Proto jsem mířil do oblastí dvou spekter signálu - časové a frekvenční. Metody v časovém spektru jsou výpočetně výhodnější, protože pracují rovnou se vstupním signál, nemusí se upravovat a převádět. Abych se vrátil k metodám, v časovém spektru jsem použil následující - integrál celého signálu, z kterého jsem ukládal hodnoty v určitých okamžicích, zerro-crossing, threshold-crossing s více úrovněmi, maxima a lokální maximum.

Metody vycházející z frekvenčního spektra jsou lepší, protože nejsou tolik závislé na stejné hodnotě spouštění triggeru, zato jsou výpočetně náročnější. Proto jsem pro převod do frekvenčního spektra nepoužíval rychlou Fourierovu transformaci, ale použil jsem jednodušší algoritmus na zjištění frekvenčního spektra a tím je Goertzel algoritmus. Ten na rozdíl od FFT nepočítá celé frekvenční spektrum, ale pouze frekvenci, na kterou je potřeba se podívat. Přesněji Goertzel algoritmus vrací výkon pro danou frekvenci. Oproti FFT má méně výpočtů, nezabírá takový paměťový prostor, nemá nároky na počet vstupních dat, nemusí se dělat bitová inverze dat a pro určitý počet dat dosahuje větší přesnosti. Jako metody pro frekvenční spektrum jsem použil - maximální frekvenci,

lokální maximální frekvence, poslední tři dominantní frekvence, lokální minima, trend frekvence v časových oknech signálu, vlastnosti frekvence v časových oknech a součet celkové energie v daném frekvenčním rozsahu. Abych dodržel výpočetní nenáročnost, postupoval jsem tak, že jsem vypočítal frekvence přes celé frekvenční spektrum s krokem 1 *kHz* a pokud byly splněné podmínky, rozebíral jsem signál dále.



Obr. 59: Časové průběhy akustického signálu ručního alarmu a techniky bumping

Při výběru těchto metod jsem postupoval tak, že jsem si nejdříve zobrazil všechny zvuky v časové i frekvenční oblasti pomocí MATLABu. Zde jsem hledal viditelné příznaky, které odlišují jednotlivé typy zvuků a snažil se metody připravit tak, aby je dokázali vyhodnotit a číselně popsat. Nebylo to jednoduché, protože trénovací data jsou naměřena z různých míst a i pro stejnou výstupní třídu se některé průběhy už na první pohled dost liší, viz Obr. 59, kde je vidět i vliv triggeru. Taky zde má vliv délka snímaného časové okna, která je například u zobrazovaného ručního alarmu znatelná, protože zvuk alarmu se mění v čase. Některé metody si jsou podobné a může se zdát, že už jsou zbytečné, ale chtěl jsem vytvořit co nejvíce příznaků a nechat druhotně nadřazený SW prohledat a prozkoumat všechny další možné spojitosti, které jsem pouhou zrakovou dedukcí nenašel.

Všechny metody jsem si nejdříve naprogramoval v MATLABu s co největší podobností s programovacím jazykem C, aby konverze mezi MATLABem a MCU nebyla složitá a byla vůbec reálná. Dohromady všechny takto připravené funkce jsou výpočetně náročné, ale vycházím z předpokladu, že z náročných operací budou vybrány pouze dílčí hodnoty, které samy o sobě nebudou složité. Nedříve se ale musí celý příznakový prostor důkladně prohledat, aby byly tyto dílčí hodnoty objeveny.

Z takto připravených funkcí jsem vytvořil program, který postupně načítá jednotlivé akustické signály uložené v centrální souboru a vytváří matici příznaků. Ve sloupcích jsou ukládány příznaky a v řádcích všechny vstupní zvuky. V posledních dvou sloupcích jsou uložena čísla výstupní třídy a daného zvuku. Výslednou matici příznaků program převede na soubor typu *csv*, se kterým dále pracuji v programu RapidMiner. Výsledkem tohoto procesu je soubor o velikosti 123 příznaků pro 143 trénovacích zvuků.

8.8.3 Příznakový vektor

Příznakový vektor obsahuje číselně vyjádřené vlastnosti akustického signálu, které ho popisují. Příznakových vektorů je *M*, kde *M* je počet výstupních tříd. Navrhovaný systém má sedm výstupních tříd, které jsou rozděleny na tři skupiny. První skupina jsou zvuky od alarmů a unikajícího vysokotlakého plynného média, druhá je tříštění skla a třetí skupina jsou rány od vrat a zvuk násilného dobývání při použití techniky bumping, viz Tab. 2.

Označení výst. tříd	**Popis**
HandAlarm	Ruční alarm („vajíčko“) proti krádežím
FICO	Kouřové a CO bezpečnostní čidlo
FallGlassIntoShards	Padající kus skla do hromady střepů
FallGlassOnGlass	Padající kus skla na druhý kus skla, který se nerozbije
CompressedAir	Unik vysokotlakého vzduchu
BlowDoors	Rána s velkými plechovo-dřevěnými vraty
HitIntoDoor	Rány při udeření do zámku dveří (technika bumping)

Tab. 2: Vysvětlení názvů výstupních tříd a jejich popis

Prvním kritériem výběru příznaků byla výpočetní náročnost, výpočet příznaků nesmí být paměťově ani výpočetně náročný. Jednak kvůli případné optimalizaci na slabší MCU s bateriovým napájením a také kvůli časové odezvě celého signálu.

Row No.	Label	A11	A61	A73	A75	A77	A84	A87	A94	A100	A109
1	HandAlarm	34	0.340	3.746	0.444	2995	0.021	0.015	0.458	0.143	0.160
2	HandAlarm	34	0.269	3.476	0.306	3215	0.023	0.018	0.279	0.217	0.205
3	HandAlarm	21	0.247	1.296	0.109	3155	0.017	0.009	0.425	0.088	0.070
4	HandAlarm	26	0.202	1.805	0.177	3135	0.011	0.006	0.838	0.068	0.049
5	HandAlarm	33	0.161	5.440	0.438	3035	0.027	0.020	0.475	0.156	0.139
6	HandAlarm	23	0.182	1.500	0.158	3155	0.013	0.011	1.161	0.043	0.014
7	HandAlarm	31	0.273	4.104	0.337	3115	0.019	0.016	0.413	0.185	0.176
8	HandAlarm	34	0.086	2.552	0.321	3075	0.016	0.013	1.217	0.101	0.086
9	HandAlarm	20	0.211	1.773	0.149	3155	0.020	0.010	0.631	0.154	0.098
10	HandAlarm	24	0.052	3.003	0.350	2995	0.008	0.006	0.292	0.052	0.062
11	HandAlarm	32	0.203	4.653	0.364	3035	0.007	0.006	0.796	0.100	0.064
12	HandAlarm	31	0.181	2.817	0.320	3095	0.017	0.012	0.892	0.120	0.105
13	HandAlarm	33	0.339	3.420	0.426	2995	0.016	0.014	0.565	0.269	0.254
14	HandAlarm	34	0.173	3.731	0.452	2995	0.011	0.007	0.348	0.054	0.055
15	HandAlarm	32	0.222	4.683	0.357	3075	0.031	0.022	0.215	0.178	0.150
16	FICO	24	0.153	0.811	0.068	3255	0.021	0.014	0.211	0.147	0.106

Obr. 60: Část selektovaných příznakových vektorů z RapidMineru pro výsledný model

8.8.4 Předzpracování dat

S takto připraveným souborem příznaků, stále se však jedná o trénovací data vyjádřená číselnou hodnotou, jsem přistoupil už k zmíněnému programu RapidMiner. Nejdříve jsem navrhl základní schéma zapojení bloků, které obsahuje načítání příznaků, normalizaci, cross-validaci s modelem a výpočtem přesnosti predikce. Jako klasifikační model jsem zvolil jeden ze základních modelů z kategorie líného učení, k nejbližších sousedů (k-NN). Výsledek byl nad očekávání dobrý, celková přesnost predikce 84%, kde pro učení modelu byly použity všechny příznaky. Bylo potřeba zredukovat počet příznaků a vybrat pouze ty nejvýhodnější, ty které nesou nejvíce informací, k tomu jsem používal metody selekce. Používal jsem metody z kategorie wrapper. Wrapper metody pracují přímo s modelem a vybírají nejlepší příznaky pro které má model největší přesnost. Tyto metody jsou výpočetně náročné, to ale v tomto případě nevadí, protože model je učen v PC a ne v MCU, naopak to přinese zjednodušení pro výsledný model v MCU.

accuracy: 84.50% +/- 32.94% (mikro: 84.89%)

	true HandAlarm	true FICO	true FallGlassInt	true FallGlassOn	true Compresse	true BlowDoors	true HitIntoDoor	class precision
pred. HandAlarm	14	0	0	0	0	0	0	100.00%
pred. FICO	1	20	0	0	0	0	0	95.24%
pred. FallGlassIn	0	0	15	4	1	0	0	75.00%
pred. FallGlassO	0	0	7	17	0	0	0	70.83%
pred. Compresse	0	0	3	1	17	0	0	80.95%
pred. BlowDoors	0	0	0	0	0	14	0	100.00%
pred. HitIntoDoor	0	0	1	0	0	3	21	84.00%
class recall	93.33%	100.00%	57.69%	77.27%	94.44%	82.35%	100.00%	

Obr. 61: Tabulka s přesností predikce jednotlivých výstupních tříd, bez selekce příznaků, k-NN

Použil jsem dva zástupce z této kategorie a to dopřednou selekci, kterou jsem používal na hrubou selekci příznaků, přesně pro 30 nejlepších a poté jsem použil, už na menší počet příznaků, přesnější metodu brutal-force, která garantuje výběr optimální kombinace příznaků. Když jsem použil metodu brutal-force pro všechny příznaky, nestačil mi pro výpočetní výkon počítače (výpočet si zabral 7GB RAM a přesto mu to nestačilo), proto jsem použil pro hrubou selekci metodu dopředná selekce, zapojení v RapidMineru Obr. 58.

První úprava, která ale patří do sekce předzpracování dat, je normalizace příznaků. Tu jsem ale neprováděl, protože normalizace pro k-NN je od -1 – 1, ale maximální a minimální hodnota se získává z trénovačích dat. To ale způsobuje problémy u dat, na které se model neučil a maximální nebo minimální hodnoty mohou být větší a překročit tak nastavený interval. Zvlášť je to znát u frekvenčních maxim, kdy hodnoty dosahují tisíce a při lineární normalizaci by ostatní příznaky přišly o informace vlivem zaokrouhlování, proto jsem normalizaci neprováděl.

Další úpravou v předzpracování dat je vyřazení tzv. outliers. Tato metoda odstraní příznaky, které nějakým přičiněním vystupují z celkového uspořádaní. Příčiny jsou různé, v mém případě jsou trénovací data naměřena všechna, tak jak budou v reálných situacích, proto by odstranění těchto příznaků omezilo robustnost predikce. Takže jsem odstranění outleirs nepoužíval.

8.8.5 Klasifikační modely

Další částí byl výběr klasifikačního modelu. Na první predikci jsem použil model k-NN, bylo ale nutné vybrat model, který je dostatečně přesný, jednoduše implementovatelný a výpočetně nenáročný pro MCU. Do metod selekce příznaků z předchozí kapitoly jsem vkládal aktuální modely. Testování přesnosti jsem prováděl pomocí Cross-Validace.

8.8.5.1 k – nejbližších sousedů (k-NN)

Klasifikační model k-NN dosáhl přesnosti predikce 90%, pro 12 příznaků vybraných pro tuto metodu předzpracováním signálu. Problém nastal hned ve dvou případech, s implementací do MCU a ignorací ostatních zvuků, které se budou vyskytovat v prostředí snímače. Problém s ostatními zvuky ale není pouze u k-NN, ale i u ostatních metod klasifikace. Problém s implementací byl takový, že metoda k-NN hledá *k* nejbližších sousedů, to by ale znamenalo, že v MCU bych musel mít uloženo minimálně počet *k* příznaků, s kterými bych aktuálně vypočítané příznaky porovnával. Na výběru těchto vzorů by pak závisela robustnost predikce.

accuracy: 90.00% +/- 9.15% (mikro: 89.93%)

	true HandAlarm	true FICO	true FallGlassInto	true FallGlassOn(	true Compressed	true BlowDoors	true HitIntoDoor	class precision
pred. HandAlarm	15	0	0	0	0	0	0	100.00%
pred. FICO	0	20	0	0	0	0	0	100.00%
pred. FallGlassInt	0	0	21	3	0	1	0	84.00%
pred. FallGlassOr	0	0	2	18	0	0	0	90.00%
pred. Compresse	0	0	1	0	18	0	0	94.74%
pred. BlowDoors	0	0	2	1	0	13	1	76.47%
pred. HitIntoDoor	0	0	0	0	0	3	20	86.96%
class recall	100.00%	100.00%	80.77%	81.82%	100.00%	76.47%	95.24%	

Obr. 62: Tabulka s přesností predikce jednotlivých výstupních tříd klasifikačního modelu k-NN s 12 příznaky

Proto jsem zvolil jednodušší řešení, které vytvoří ze všech trénovacích příznaků jeden střed, ke kterému se bude vztahovat vyhodnocení. Tuto úpravu však RapidMiner nepodporuje, takže jsem celý algoritmus přeprogramoval do MATLABu. Jako střed příznaků jsem zvolil medián trénovacích příznaků. Aby byl klasifikátor robustnější, přidal jsem do algoritmu hledání druhé nebližší hodnoty, které jsem ve výsledném součtu přiřazoval poloviční váhu. Druhý problém, ostatní zvuky, jsem vyřešil maximální vzdáleností od vzorových příznaků. Vzdálenost jsem v algoritmu určil z maximální hodnoty trénovacích příznaků plus rezervu. Takto navržený algoritmus měl celkovou přesnost predikce 75%. Algoritmus bohužel selhával pro výstupní třídu pád skla na sklo, kde měl přesnost pouze 45%. S těmito výsledky jsem tuto metodu opustil a přešel jsem k použití jiné.

8.8.5.2 Naive Bayes

Další klasifikační metodou, kterou jsem použil, byl Naive Bayes. Zde vycházela celková přesnost bez selekce příznaků na 87%. Pro selekci příznaků jsem použil již popisovaný postup v předzpracování a přidal jsem ještě dopřednou váhovou optimalizaci příznaků a zvolil jsem maximální počet 10 příznaků. Před část selekce příznaků jsem z realizačních důvodů přidal blok, který příznaky diskretizuje do intervalů. Bez tohoto kroku bych

nemohl efektivně model implementovat. Zvolil jsem rozdělení pouze do dvou intervalů pro snazší implementaci. S těmito úpravami a omezeními jsem přesto získal 90% přesnost predikce, viz tabulka přesností predikce modelu pro všechny výstupní třídy, vztažené k chybám 1. a 2. druhu Obr. 63.

accuracy: 90.48% +/- 9.49% (mikro: 90.51%)

	true HandAlarm	true FICO	true FallGlassInto	true FallGlassOn(	true Compressed	true BlowDoors	true HitIntoDoor	class precision
pred. HandAlarm	15	0	0	0	0	0	0	100.00%
pred. FICO	0	19	0	0	0	0	0	100.00%
pred. FallGlassInt	0	0	20	1	1	1	0	86.96%
pred. FallGlassOr	0	0	3	19	0	0	1	82.61%
pred. Compresse	0	0	2	1	17	0	0	85.00%
pred. BlowDoors	0	0	1	0	0	15	1	88.24%
pred. HitIntoDoor	0	0	0	0	0	1	19	95.00%
class recall	100.00%	100.00%	76.92%	90.48%	94.44%	88.24%	90.48%	

Obr. 63: Tabulka s přesností predikce jednotlivých výstupních tříd klasifikačního modelu Naive Bayes s 10 příznaky

Selekcí příznaků vyšlo jako nejvíce informativně přínosných 10 příznaků. Kromě jednoho byly všechny příznaky vybrány z frekvenční oblasti. Jediný zástupce příznaku z časové oblasti byl threshold-crossing s hodnotou prahu -0,2. Ostatní příznaky z frekvenční oblasti patřily metodě frekvenční vlastnosti ve frekvenčních oknech s krokem 500 Hz, kde byly parametry dominantní frekvence, její hodnota a součet všech frekvencí v daném okně. Část tabulky selektovaných příznaků je zobrazena na Obr. 60

Ostatní zvuky u tohoto modelu filtruji tak, že pokud je predikovaná přesnost pro daný zvuk menší než referenční hodnota, tak je zvuk označen jako ostatní a ignoruje se. Referenční hodnota je pro každou výstupní třídu nastavená na jinou hodnotu. V tomto místě by se dalo provádět druhotné vyhodnocení s časovými intervaly a čítači událostí a ošetřit tak tyto stavy. Ve výstupní modelu není tato nadstavba zařazena.

Attribute	Parameter	HandAlarm	FICO	FallGlassInt...	FallGlassO...	Compresse...	BlowDoors	HitIntoDoor
A11	value=range1 [-∞ - 28.500]	0.333	0.150	0.538	0.682	0.056	0.588	1
A11	value=range2 [28.500 - ∞]	0.667	0.850	0.462	0.318	0.944	0.412	0
A11	value=unknown	0	0	0	0	0	0	0
A61	value=range1 [-∞ - 0.693]	1	1	0.346	0.318	1	0	0
A61	value=range2 [0.693 - ∞]	0	0	0.654	0.682	0	1	1
A61	value=unknown	0	0	0	0	0	0	0
A73	value=range1 [-∞ - 1.085]	0	0.950	0.500	0.955	0.500	0.353	0.048
A73	value=range2 [1.085 - ∞]	1	0.050	0.500	0.045	0.500	0.647	0.952
A73	value=unknown	0	0	0	0	0	0	0
A75	value=range1 [-∞ - 0.088]	0	0.800	0.538	1	0.500	0.294	0.143
A75	value=range2 [0.088 - ∞]	1	0.200	0.462	0	0.500	0.706	0.857
A75	value=unknown	0	0	0	0	0	0	0
A77	value=range1 [-∞ - 3245]	1	0.350	0.385	0.273	0.389	0.471	0.381

Obr. 64: Část tabulky dílčích pravděpodobností P(X|Cj)pro klasifikační model Naive Bayes

Prvním plánem, jak řešit ostatní zvuky, bylo řešení s nasnímáním okolních zvuků a nechat model naučit s touto novou výstupní třídou, ale výsledek dopadl velice špatně. Hlavním problém byl velký počet příznaků (16), které dosáhli přesnosti pouze 83% a velké chyby v predikci. Tím myslím chyby druhu, ruční alarm predikován jako bouchnutí dveřmi, když si všimnete na Obr. 63, většina chybných predikcí je v rámci jedné skupiny zvuků. Například zvuk střepů je predikován jako zvuk sklo dopadající na sklo. Proto jsem použil předchozí popisovanou metodu.

Výstupem klasifikačního modelu je tabulka dílčích pravděpodobností pro dané intervaly jednotlivých výstupních tříd, viz Obr. 64. Lépe řečeno, tabulka s jakou pravděpodobností patří daný příznak do výstupních tříd (věrohodnost).

Tato verze klasifikačního modelu s 10 příznaky jsem implementoval do MCU, viz následující podkapitola. Po implementaci předchozího modelu jsem ještě navrhl jednu optimalizovanou verzi, která má pouze 5 příznaků a disponuje přesností predikce 85%. Takto navržená verze modelu může sloužit jako výchozí model pro optimalizovanou verzi systému.

8.8.5.3 Implementace modelu do MCU

Při implementaci do MCU jsem postupoval následovně. Abych mohl ověřit správnost predikce na trénovacích datech, tak jsem nejdříve výpočetní algoritmus klasikátoru vytvořil v MATLABu. Tento postup je složitější a časově náročnější, ale díky tomu můžu ověřit funkčnost lépe než v MCU. Bohužel RapidMiner nemá v základní verzi výstup modelování, například generátor kódu nebo podobné uživatelské funkce. Výpočet vychází z teorie, viz 6.3.2.

Nejdříve jsem si zkopíroval tabulku s dílčími pravděpodobnostmi $P(x_i|Cj)$ a uložil jsem ji do paměti jako dvourozměrné pole. Vytvořil jsem `for` smyčku s počtem opakování výstupních tříd a počítal sdruženou pravděpodobnost $P(X|Cj)$ pro danou výstupní třídu. Hodnota dílčí pravděpodobnosti je dána hodnotou aktuálního příznaku, to znamená, do kterého intervalu spadá a číslem dané výstupní třídy, s těmito indexy se odkazuji na uložené pole. Rozdělení příznaků do intervalů jsem provedl pomocí `if` a `else`, podobě jako metoda rozhodovací stromy s jednou úrovní. Sdruženou pravděpodobnost $P(X|Cj)$ pro jednotlivé výstupní třídy je dána součinem všech dílčích pravděpodobností příznaků (16). Celková pravděpodobnost dané výstupní třídy je určena sdruženou pravděpodobností pro danou třídu, krát apriorní pravděpodobnost lomeno součet všech sdružených pravděpodobností, viz (17). Apriorní pravděpodobnost jsem zvolil jako 1/počet výstupních tříd, protože předpokládám stejnou pravděpodobnost výskytu všech výstupních třid. Nevolil jsem zjednodušený vztah (18), protože nemůžu zaručit nezávislost příznaků. Výslednou klasifikaci modelu určuji jednoduchým výpočtem maxima.

Takto připravený a otestovaný algoritmus už jen stačilo přenést do MCU, vytvořit výstupní hodnoty a zabudovat do stávajícího algoritmu. Pro detekci jsem v MCU vytvořil funkci *recognize()*, která vytváří příznaky pro klasifikační model pomocí funkce *goetrzel()* a vyhodnocuje příznaky klasifikačním modelem pomocí funkce *NBayes()*. Vstupním parametrem této funkce jsou převedená data na PCM datový formát. Funkce vrací status, který odpovídá klasifikaci modelu. Stejně jako u předchozích funkcí jsou dílčí procesy zařazeny do hlavního cyklu, aby celková funkce byla co nejúspornější.

8.8.5.4 Výběr klasifikačního modelu

K výběru modelu pro MCU vedlo více aspektů. Prvním aspektem byla reálná implementace modelu do MCU, výpočetně nenáročná realizace modelu, jednoduché filtrování ostatních zvuků a efektivita modelu. Všechny tyto aspekty Naivní Bayesovský klasifikátor splňoval, proto jsem ho vybral jako finální metodu klasifikace a implementoval ho do MCU.

8.9 Komunikace

Komunikace s vnějším světem probíhá dvěma způsoby. První je ladící kanál pomocí sériové komunikace, který byl zachován vzhledem k tomu, že zprávy jsou malých velikostí a nezdrží celkový algoritmus. Ve zprávách jsou po spuštění posílány informace o nadřazené komunikaci (číslo portu, IP adresa) a alarmové zprávy během činnosti systému. Nadřazená komunikace používá UDP protokol pro komunikaci v síťovém režimu. Systém je nastaven jako klient a posílá zprávy na definovaném portu. V tomto nastavení se zprávy posílají typu broadcast, takže stav je posílán i na ostatní systémy v síti.

Vývojová platforma FRDM-K64F podporuje ethernetové rozhraní s vlastním Ethernet MAC kontrolerem. Pro vytvoření vyšších ISO OSI vrstev jsem použil lwIP TCP/IP Stack. Pro komunikaci jsem zvolil UDP protokol s pevnou IP adresou. Řešení je postaveno na komunikačním protokolu UDP, v dalších verzích bych kvůli bezpečnosti zpráv přešel na protokol TCP, který vychází ze stejné vrstvy, ale je potvrzovaný.

Pro ověření komunikace mezi FRDM-K64F a PC jsem používal program Wireshark. V programu Herkules jsem si vytvořil server se stejným číslem portu a nechal jsem si zprávy zobrazovat na konzoly.

Dalším prvkem komunikace jsou časové značky PTP protokolu. Zde jsem bohužel narazil na realizační problém ze strany firmy Freescale (NXP). Realizační problém představuje samotná platforma. Platforma obsahuje hardware pro časové značení ethernetové komunikace, které bylo i jedním parametrem pro kritéria výběru této platformy. Bohužel pro platformy Freescale nejsou zatím vytvořeny open-sourcové knihovny (PTP stack), které by řešily rutiny tohoto protokolu (řízení synchronizace hodin a vytváření normou definované datové rámce a další). Stejný jako je pro vytvoření vyšších vrstev ethernetový komunikace lwIP. Jediná softwarová podpora, kterou tato platforma zatím nabízí, je placený stack od softwarové průmyslové firmy IXXAT. Ta nabízí stack běžící na operačním systému MQX přímo pro dané zařízení. Nabízejí k vyzkoušení demo program, ale pouze binární soubor, ne source code. Řešení zadarmo existuje, ale pro jiné platformy například od firmy STM, na kterých je podporován PTPd stack.

Na začátku této práce jsem počítal s realizací PTP protokolu a obohatit tak výstup systému o časový údaj události. Proto jak už jsem zmínil, byla platforma vybírána s podporou PTP protokolu. Ve všech propagačních materiálech byla tato vlastnost popisována. Bohužel mě na začátku nenapadlo podívat se až k podpoře kódům, kde nastal problém. Když jsem po obrovské časové ztrátě vyřešil zmiňované problémem s KSDK a

vytvořil vlastní knihovny pro obsluhu a na řadu přišly časové značky, došel jsem k tomuto problému. Měl jsem i plány přejít k platformě od STM, ale nebylo by v mých silách za polovinu času vše předělat na nové platformě s novým vývojovým prostředím a přitom stihnout další části systému - predikci a zpracování signálu. Také jsem se pokoušel přelinkovat alespoň jádro PTPd stacku, ale po několika dnech nikam nepostupující práce jsem od tohoto kroku ustoupil. Kód je psaný ve vyšším jazyce a zkombinovat všechny časti, tak aby je dokázal překladač přeložit, je velice obtížné.

Tato možnost je popsána teoreticky, tak jak by se postupovalo při její praktické realizaci. Kód je na přítomnost PTP protokolu téměř připraven, pokud by se objevila podpora PTPd stacku, jednoduše by se implementoval do systému a vzhledem k tomu, že synchronizace probíhá sama, když je volná komunikace a časové značky nejsou vkládány softwarově, ale v nejnižší vrstvě Ethernetu, tak by tato úprava nebyla nijak náročná. V inicializaci jsem připravil i zdroj hodin časových značek a částečně i potřebné registry pro tyto periferie.

8.9.1 Časová identifikace

Časová identifikace slouží v systému pro určení polohy zdroje zvuku. Návrh systému koresponduje s reálnou časovou nejistotou synchronizace v síti v řádu mikrosekund, pro PTPv2 dokonce pod mikrosekundu. Tyto časy platí, pokud mají zařízení HW podporu PTP. Pro výpočet mezní vzdálenosti modulů volím časovou nejistotu 1 μs, protože platforma podporuje HW PTP (MII). Podle základního vzorce (21) lze vypočítat vzájemnou vzdálenost modulů v distribuovaném systému, aby byl určitelný směr zdroje akustického signálu.

$$s = v \cdot t = 343{,}71 * 2 \cdot 10^{-6} = 687 \cdot 10^{-6}\ [m] \qquad (21)$$

kde v je rychlost zvuku ve $20°C$, t časové rozlišení synchronizace a s je mezní vzdálenost modulů.

Pomocí rovnice (21) jsem vypočítal mezní vzájemnou vzdálenost modulů, která je 687 μm. Aby bylo možné určit směr zdroje akustického signálu v jedné rovině, musí být moduly od sebe vzdáleny více jak tato mezní hodnota. Výpočet platí pro prostředí s teplotou $20°C$. Teplota tady však nepředstavuje velký problém, protože mezní vzdálenost se v rozsahu $\pm 20°C$ změní o $\pm 12\ \mu m$. Při rozměrech MCM typicky 4x5 mm není zásadní problém. Pokud by byla vzdálenost menší než 687 μm, nebylo by možné určit směr. Počet určení možných směrů takto sestaveného distribuovaného systému je závislý na počtu připojených modulů v síti. Při správném rozmístění modulů je minimální počet pro určení polohy zdroje zvuku v rovině tři moduly.

Spojením této schopnosti lokalizace a schopnosti systému identifikovat akustické zdroje zvuku by bylo možné vytvořit síť velice sofistikovaných zařízení. Takovýto distribuovaný systém by mohl být využíván pro otevřené prostory, například pro těžební průmysl, sledování a vytváření zvukové mapy. Ale díky velice přesné časové synchronizaci lze dosáhnout i malých vzdáleností v uzavřených prostorách, například

sledování pohybu, především osob v pokročilém věku, nebo pro jiné zvukovo-orientační potřeby. Díky přesnosti menší než 1 *mm* lze takovýto distribuovaný systém používat i pro laboratorní účely. Nadřazený systém bude v síti představovat, například PC. Ten bude také zaujímat funkci master clock, který bude řídit synchronizaci. Následný lokalizace z časových hodnot posílaných po síti se provádí až v tomto nadřazeném systému, například PC SW.

Další možností je vytvoření distribuovaného systému pouze z modulů. Zde by funkci serveru a master clock převzal jeden z modulů, což platforma podporuje. Následné vyhodnocení informací příchozích do nadřazené stanice by mohlo být šířeno do vnějších sítí pomocí bezdrátové komunikace využívající sítí, které pokrývají mobilní operátoři (Sigfox, LoRa). Tímto řešením by se stal systém nezávislý na monitorování výsledků a používal by vnitřních rutin nadřazených sítí k vygenerování varovných zpráv, které by l být v podobě volání, SMS, emailu a dalších. Tento systém by mohl být umístěn v oblastech, kde není dostupné připojení k vnější síti, například v již zmiňovaných lomech. Tato realizace by se také hodila pro moduly s bateriovým napájením vzhledem k podpoře lowpower režimů platformy. V tomto případě by se však zhoršila časová nejistota synchronizace z důvodu velké periody synchronizace. Takovýto systém bude možné uplatnit i v jiných odvětvích díky svoji flexibilitě.

9 TESTOVÁNÍ IDENTIFIKACE

Testování jsem rozdělil pro každou výstupní třídu. Do firmwaru MCU jsem pro tento účel přidal počítadla událostí a spustil testování. Průběh měření byl takový, že jsem vzal podmět první výstupní třídy a zkoušel jsem z různých pozic a situací 30x reakci systému. Po třiceti měřeních jsem opsal hodnoty čítačů. Takto jsem prováděl pro všechny třídy s výjimkou ostatních zvuků (other). Testování ostatních zvuků probíhalo v přirozeném prostředí se zvýšenou intenzitou hluku, kde bylo kontinuálně snímáno v průběhu 5 hodin. Výsledky měření jsou zaznamenány v první Tab. 3. Bohužel výstupní třída D, atypický zvuk plechovo-dřevěných vrat, jsem z důvodů rekonstrukce a následného odstranění vrat nemohl provést testování, a jelikož se tak stalo u konce této práce, nebyl čas trénovat a přeučovat celý model od znovu.

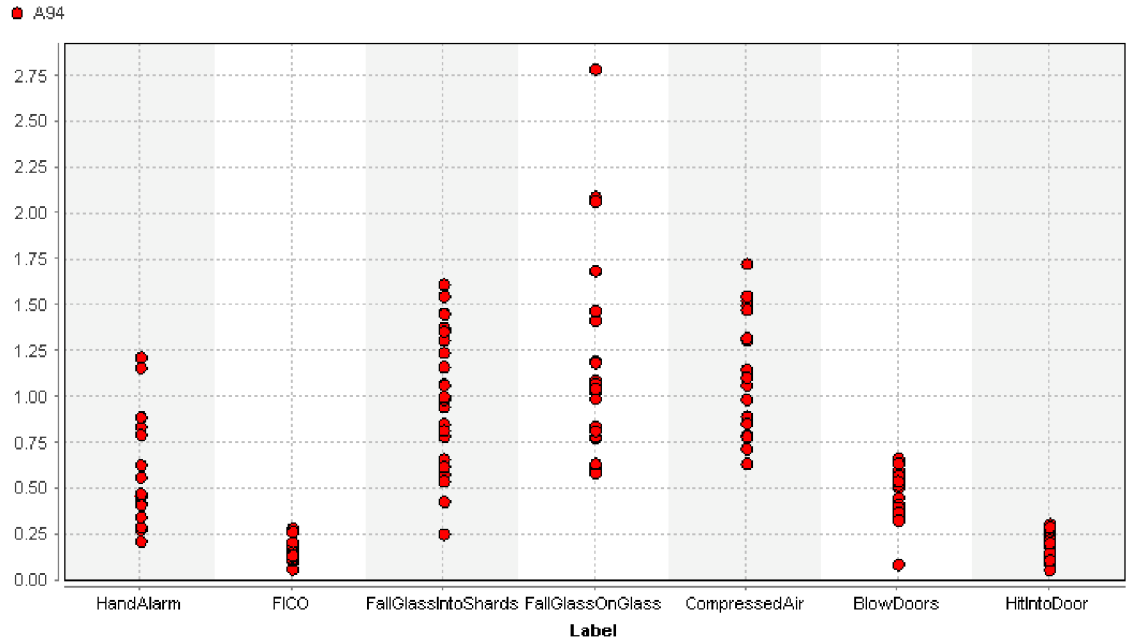
C\true	*A_H*	*A_F*	*S*	*G*	*Air*	*D*	*HD*	*Other*	Pt %
A_H	29	0	0	0	0	0	0	4	**87,88**
A_F	0	30	0	0	0	0	0	0	**100,00**
S	0	0	14	11	6	0	0	0	**45,16**
G	0	0	4	7	0	0	0	0	**63,64**
Air	0	0	6	5	24	0	0	0	**68,57**
D	0	0	0	0	0	0	0	0	
HD	0	0	0	0	0	0	27	8	**77,14**
Other	1	0	6	7	0	0	3	530	**96,89**
Pc %	**96,67**	**100,00**	**46,67**	**23,33**	**80,00**		**90,00**	**97,79**	

C\true	*A_H*	*A_F*	*S*	*G*	*Air*	*D*	*HD*	*Other*	Pt %
A_H	29	0	0		0	0	0	4	**87,88**
A_F	0	30	0		0	0	0	0	**100,00**
S	0	0	36		6	0	0	0	**85,71**
G									
Air	0	0	11		24	0	0	0	**68,57**
D	0	0	0		0	0	0	0	
HD	0	0	0		0	0	27	8	**77,14**
Other	1	0	13		0	0	3	530	**96,89**
Pc %	**96,67**	**100,00**	**60,00**		**80,00**		**90,00**	**97,79**	

Tab. 3: Výsledky predikce na testovací podměty na reálném systému s naučeným modelem

Druhý problém nastal při predikci kategorie zvuků skla. Příznaky této kategorie jsou v příznakovém prostoru tak blízko u sebe i přes sebe, že nemohou být pro rozmanitost zvuků z této kategorie vyhodnoceny, viz Obr. 66. V tomto případě nešlo o reakce modelu, ale o umění napodobit naučené zvuky. Je velice těžké z hromady střepů dostávat stále podobné zvuky. Zde záleží na velikosti střepů, na počtu a dalších faktorech, proto jsem tyto dvě výstupní třídy (náraz sklo o sklo *G* a pád skla do střepů *S*) sjednotil do jedné

výstupní třídy, viz druhá tabulka Tab. 3. Tříštění skla je samo o sobě náročné predikovat, například komerčně prodávané detektory používají kombinaci zvukové predikce a tlakového senzoru. Kdyby bylo pro trénování modelu poskytnuty stejné podmínky pro všechny trénovací zvuky, tzn. tabule skel rozbíjené různými předměty, tak věřím, že predikce bude o několik procent lepší. I přesto je tato kategorie predikována s velkou chybou a je zde zaměňována s výstupní třídou *Air* (tlakový vzduch). Celková přesnost klasifikace modelu při praktických zkouškách je 86,7%.



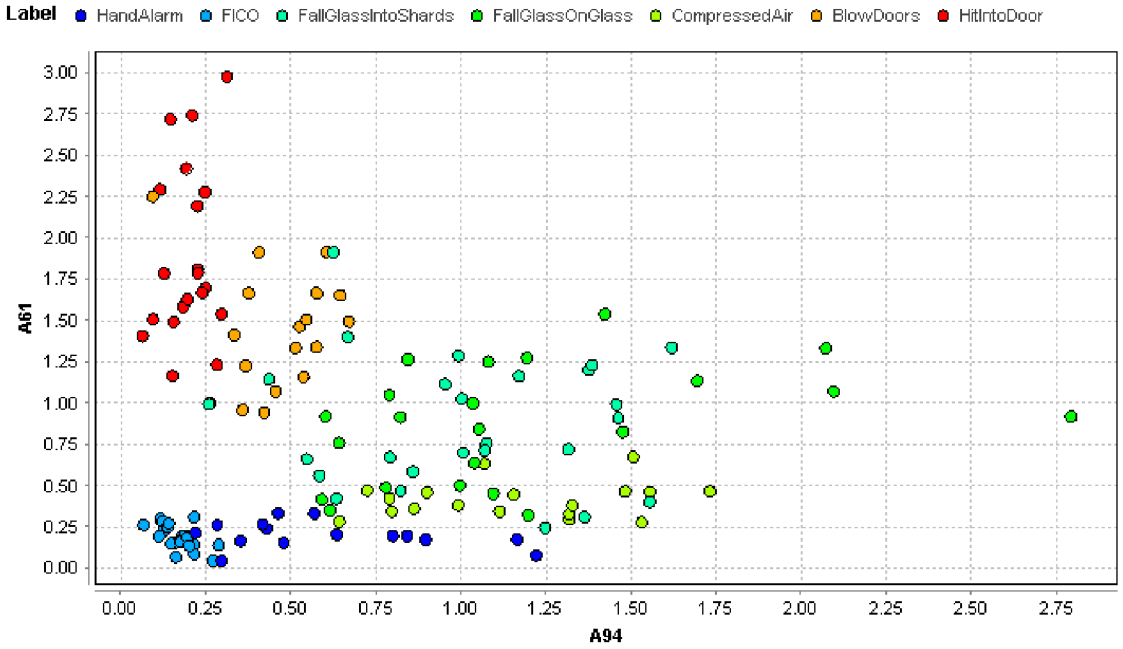
Obr. 65: Rozptyl hodnot příznaku A94 (výkon maximální frekvence) pro jednotlivé výstupní třídy

První možnou příčinou této chybné predikce může být již zmiňovaná trénovací množina příznaků. Tato množina se skládá z různých tónů skla a rozptyl hodnot příznaků je obrovský, viz Obr. 65. Dokonce zasahují do středů ostatních tříd a příznaky nevytvářejí pro tuto výstupní třídu celistvý shluk, viz Obr. 66. Druhá příčina tak velké chyby je predikování ostatních zvuků. To znamená určení hranice pravděpodobnosti, že se jedná o daný zvuk. V 50% případů chybné predikce, byla predikce samotného bayesovského klasifikátoru správná, ale pod hranicí druhotné predikce pro ostatní zvuky. Tuto hranici jsem pro tyto výstupní třídy hledal ručně, ale oddělení všech situací nebylo pro tyto zvuky možné.

Zlepšení predikce těchto výstupních tříd by určitě přinesla důkladnější příprava příznaků, například pomocí časově-frekvenční analýzy, která v menší míře byla zahrnuta ke zpracování všech příznaků, ale nebyla vybrána selekcí. Signál rozdělený časově-frekvenční analýzou by dokázal oddělit zvuky s rezonančním doběhem od nerezonančního. Pak by zvuky, jako je únik plynu a rozbití skla byly v příznakovém prostoru odlišitelné. Na tuto analýzu bude ale potřeba větší výpočetní výkon.

Další možná úprava pro zlepšení přesnosti predikce je rozdělení příznaků do sekcí a každou sekci vyhodnocovat zvlášť jinou metodou a na konci výsledky klasifikace váhově spojit. To by pomohlo faktu, že všechny příznaky jsou společné pro všechny třídy,

protože při selekci byly vybírány kombinace příznaků s největší přesností predikce pro všechny výstupní třídy. Kdyby se selekce vybírala a prováděla pouze pro kategorii zvuků se stejnými spojitostmi, mohla by být predikce citlivější na menší rozdíly příznaků. Výsledkem by mohla být výsledná klasifikace pro všechny třídy účinnější.



Obr. 66: Rozmístění hodnot příznaků A61-A94 v prostoru pro jednotlivé výstupní třídy

10 ZÁVĚR

Úkolem mé diplomové práce bylo navržení a realizování modulu pro akustickou detekci. Modul identifikuje zvuky ze čtyř vybraných kategorií - alarmové zvuky, zvuk unikajícího plynného media, zvuky tříštění skla a zvuky násilného dobývání. Z těchto kategorií jsem vybral sedm zvuků, na které jsem naučil klasifikační model. Jedná se o zvuky - ručního alarmu, kouřového čidla, unikajícího vysokotlakého vzduchu, padajícího skla na sklo, padající sklo do střepů, rána plechovo-dřevěných vrat a zvuk vloupání techniky bumping. Pro detekci jsem využíval předem naučeného klasifikačního modelu. Díky tomu je možné množinu trénovacích zvuků měnit a získat tak univerzální detekční modul. Pro návrh modelu jsem používal techniky strojového učení, které jsou v teoretické části práce popsány. Výsledný systém jsem implementoval do firmwaru mikrokontroléru.

Pro snímání zvuku jsem použil digitální MEMS mikrofon, jeho výběr je podložený rešerší. Signál z mikrofonu předávám do MCU pomocí I2S interface a využívám přímého přístupu do paměti přes DMA. Bez těchto periferií by nebyl přenos dat o frekvenci 1,38 *MHz* možný. Díky nim lze dosáhnout i kontinuálního snímání. Protože je výstupní signál z digitálního mikrofonu ve formátu PDM, musí být pro další zpracování převeden na signál PCM. Pro převod jsem navrhl a realizoval optimalizovaný decimační filtr, s ohledem na výkon MCU, skládající se z CIC a FIR filtru. Pro obsluhu vnitřních periferií MCU jsem vytvořil vlastní knihovny složené z vnitřních registrů.

Klasifikační model pro detekci zvuků je taktéž realizován s ohledem na výpočetní výkon. Navrhl jsem více variant modelů lišící se počtem využívajících příznaků i typem modelu. K vytvoření příznaků jsem použil výpočetně nenáročné metody zpracování signálu. Všechny příznaky pocházejí z časového nebo frekvenčního spektra. Pro získání dat z frekvenční oblasti jsem z důvodu výpočetní nenáročnosti používal Goertzelův algoritmus. Pro výběr informačně nejpřínosnějších příznaků jsem prováděl selekci příznaků vycházející z teorie strojového učení. Pro modul jsem vybral model vytvořený z bayesovského klasifikátoru, který měl z testovaných modelů nejlepší výsledky. S tímto modelem jsem dosáhl celkové přesnosti klasifikace 90 % pro trénovací data. Přesnosti pro jednotlivé výstupní třídy naleznete na Obr. 63. Přesnost klasifikace modelu na trénovacích datech byla určena pomocí metody CrossValidation. Celková přesnost klasifikace modelu při praktických zkouškách byla 86,7 %. Zde je ale nutné podotknout, že výsledná přesnost byla vypočítána pro sjednocené výstupní třídy kategorie tříštění skla a z důvodu přítomnosti okolních zvuků byla vytvořena nová výstupní třída ostatní zvuky.

Aby modul mohl varovat nadřazený systém o aktuálním stavu, vytvořil jsem ethernetovou síť s komunikačním protokolem UDP. Vzhledem k náročnosti řízení celého řešení jsem systém obohatil o operační systém reálného času, přesněji FreeRTOS, který zajišťuje plynulé přechody mezi jednotlivými funkcemi. V případě více modulů připojených do sítě se vytvoří distribuovaný systém, pro který je navržena přesná časová synchronizace pomocí PTP protokolu definovaného normou IEEE-1588.

Při realizaci celého sytému jsem se řídil kritériem co nejméně výpočetně a paměťově náročný, aby bylo možné systém implementovat i na méně výkonné zařízení.

Výsledná práce se skládá z teoretického i praktického popisu při řešení problematiky spojené s tímto tématem. V textu jsou také uvedeny nápady a návrhy na zlepšení. Také jsou v práci popsány postupy pro další pokračování této práce a možné uplatnění modulů v praxi. Při řešení se mi podařilo splnit všechny body zadání.

Literatura

[1] JAWED, Syed Arsalan. CMOS READOUT INTERFACES FOR MEMS CAPACITIVE MICROPHONES. Trento, 2009. Dostupné také z: http://eprints-phd.biblio.unitn.it/82/1/thesis_mems_microphone_readout.pdf. Disertační práce. DIT - University of Trento. Vedoucí práce Massimo Gottardi.

[2] Mbed: FRDM-K64F [online]. 2005 [cit. 2015-12-30]. Dostupné z: https://developer.mbed.org/platforms/FRDM-K64F/

[3] FRDM-K64F: Freedom Development Platform for Kinetis K64, K63, and K24 MCUs. NXP [online]. 2014 [cit. 2015-12-21]. Dostupné z: http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F?

[4] Akustica Products: Akustica AKU240 Family & AKU440 Family. Akustica [online]. 2014 [cit. 2015-12-23]. Dostupné z: http://www.akustica.com/Digital%20HD%20voice%20mic.asp

[5] MEMS MICROPHONES: EMERGING TECHNOLOGY AND APPLICATION TRENDS. MEMS JOURNAL [online]. 2015 [cit. 2015-12-23]. Dostupné z: http://www.memsjournal.com/2015/07/mems-microphones-emerging-technology-and-application-trends.html

[6] MEMS Microphone Model Presented at ASA 166 in San Francisco. COMSOL [online]. 2014 [cit. 2015-12-23]. Dostupné z: https://www.comsol.com/blogs/mems-microphone-model-presented-asa-166-san-francisco/

[7] MEMS Mics Taking Over. EETimes [online]. 2014 [cit. 2015-12-23]. Dostupné z: http://www.eetimes.com/document.asp?doc_id=1324827

[8] Technology-Vesper MEMS. Vesper MEMS [online]. 2015 [cit. 2015-12-23]. Dostupné z: http://vespermems.com/technology/

[9] LEWIS, Jerad. Analog and Digital MEMS Microphone Design Considerations. In: Industrial automation products [online]. Norwood, MA: Analog Devices [cit. 2015-12-23]. ISBN 02062-9106. ISSN 02062-9106. Dostupné z: http://www.analog.com/media/en/technical-documentation/technical-articles/Analog-and-Digital-MEMS-Microphone-Design-Considerations-MS-2472.pdf

[10] RUMSEY, Francis a Tim MCCORMICK. Sound and recording. 6th ed. Oxford: Focal Press, 2009, s. 48-53. ISBN 978-0-240-52163-3.

[11] GLEN M. BALLOU, Glen Meditor. Handbook for sound engineers. 4th ed. Oxford: Focal, 2008. ISBN 02-408-0969-6.

[12] LYONS, G. Richard. Understanding digital signal processing. Vyd. 2. New Jersey: Prentice-Hall, 2004, 665 s. ISBN 0131089897.

[13] PARK, Sangil. MOTOROLA. Motorola Digital Signal Processors: Principles of Sigma-Delta Modulation for Analog-to- Digital Converters. 1. Dostupné také z: http://www.numerix-dsp.com/appsnotes/APR8-sigma-delta.pdf

[14] JARMAN, David. INTERSIL. A Brief Introduction to Sigma Delta Conversion. -, 1995. Dostupné také z: http://www.intersil.com/content/dam/Intersil/documents/an95/an9504.pdf

[15] KITE, Thmas. AUDIO PRECISION. Understanding PDM Digital Audio. Beaverton, 2012. Dostupné také z: http://users.ece.utexas.edu/~bevans/courses/rtdsp/lectures/10_Data_Conversion/AP_Understanding_PDM_Digital_Audio.pdf

[16] Digilentinc. Refmanual [online]. 2015 [cit. 2015-12-31]. Dostupné z: https://reference.digilentinc.com/nexys4-ddr:refmanual

[17] Engineer-to-Engineer Note: Seamlessly Interfacing MEMS Microphones with Blackfin® Processors. 2010. Dostupné také z: http://www.analog.com/media/en/technical-documentation/application-notes/EE-350rev1.pdf

[18] LYONS, Richard. CIC FILTERS: Understanding cascaded integratorcomb filters. In: EE Times-India [online]. 2005 [cit. 2015-12-31]. Dostupné z: http://www.eetindia.co.in/STATIC/PDF/200504/EEIOL_2005APR01_EMS_RFD_SIG_TA.pdf?SOURCES=DOWNLOAD

[19] FREESCALE SEMICONDUCTOR. K64 Sub-Family Reference Manual [online]. 2. 2014 [cit. 2016-01-02]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/ref_manual/K64P144M120SF5RM.pdf

[20] LEWIS, Jerad. INVENSENSE. Inter-IC Digital Audio Interfaces. San Jose, 2013. Dostupné také z: https://www.digikey.com/Web%20Export/Supplier%20Content/invensense-1428/pdf/invensense-inter-ic-digital-audio-interfaces.pdf?redirected=1

[21] TEXAS INSTRUMENTS. TLV320AIC3101: Low-Power Stereo Audio Codec for Portable Audio/Telephony. Dallas, 2014. Dostupné také z: http://www.ti.com/lit/ds/symlink/tlv320aic3101.pdf

[22] PHILIPS SEMICONDUCTORS. I2S bus specification. 1996. Dostupné také z: https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf

[23] DOSTÁLEK, Libor a Alena KABELOVÁ. Velký průvodce protokoly TCP/IP a systémem DNS. 2. aktualiz. vyd. Praha: Computer Press, 2000, 426 s. Komunikace. ISBN 80-722-6323-4.

[24] ZEZULKA, František a Ondřej HYNČICA. Průmyslový Ethernet II: Referenční model ISO/OSI.*AUTOMA*. 2007, (3): 86-90.

[25] Sysnchronizace v distribuovaných řídicích systémech: Precision Time Protocol (PTP) podle IEEE1588. *AUTOMA*. 2010, (2): 17-19.

[26] LYONS, Richard. Single tone detection with the Goertzel alhorithm. In: Embedded: cracking the code to systems development [online]. 19. 11. 2012 [cit. 2014-05-19]. Dostupné také z: http://www.embedded.com/design/real-world-applications/4401754/Single-tone-detection-with-the-Goertzel-algorithm

[27] KNOWLES ACOUSTICS. *Digital "Mini" SiSonic TM Microphone Specification - Halogen Free: SPM0405HD4H-WB*. Itasca, 2009. Dostupné také z: http://www.daxia.com/bibis/upload/SPM0405HD4H-WB.468.pdf

[28] PETR HONZÍK, Petr. *Strojové ucení*. 1. Brno: FEKT Vysokého ucení technického v Brne, 2006.

[29] HAN, Jiawei, Micheline KAMBER a Jian PEI. *Data mining: concepts and techniques*. 3rd ed. Boston: Elsevier, c2012. Morgan Kaufmann series in data management systems. ISBN 978-0-12-381479-1.

[30] TAN, Pang-Ning, Michael STEINBACH a Vipin KUMAR. *Introduction to data mining*. 1. Boston: Pearson Addison Wesley, c2006. ISBN 0321321367.

[31] Nonlinear Support Vector Machine. *Bindichen* [online]. 2012 [cit. 2016-05-04]. Dostupné z: http://www.bindichen.co.uk/post/AI/Nonlinear-Support-Vector-Machines.html

[32] KOHAVI, Ron a George H. JOHN. Wrappers for feature subset selection. *Artificial Intelligence* [online]. 1997, **97**(1), 273-324 [cit. 2016-05-07]. Dostupné z: http://ai.stanford.edu/~ronnyk/wrappersPrint.pdf

Seznam zkratek

MEMS	Mikro-Elektro-Mechanický Systém
MCM	MEMS Capacitive Microphone (MEMS kapacitní mikrofon)
ASIC	Application-Specific Integrated Circuit (specifický integrovaný obvod)
ADC	Analog to Digital Converter (analogově digitální převodník)
SoP	System-on-a-Packed (systém na jednom základu)
SoC	System-on-a-Chip (systém na jednom čipu)
DC	stejnosměrné napětí
M	pohyblivá elektroda (membrána)
PE	pevná elektroda
IC	Integrated Circuit (integrovaný obvod)
AC	střídavé napětí
RI	Readout Interface (integrovaný obvod)
ΔΣ	sigma delta
A/D	analogově na digitální
PDM	Pulse-Density Modulation (modulace relativní hustoty pulsů)
PCM	Puls-Code Modulation (pulzně modulovaná modulace)
FPGA	Field Programmable Gate Array (programovatelná hradlová pole)
I^2S	Integrated Interchip Sound (sériová zvuková komunikace)
MCU	Microcontroller Unit (mikrokontrolér)
DMA	Direct Memory Access (přímý přístup do paměti)
CPU	Central Processing Unit (centrální procesorová jednotka)
RAM	Random Access Memory
DSD	Direct-Stream Digital (další název pro PDM)
CIC	Cascaded Integrator Comb filtr (kaskádový hřebenový integrační filtr)
FIR	Finite Impulse Response (konečná impulsní odezva)
ARM	označení architektury procesorů
KDS	Kinetis Design Studio (vývojové prostředí)
RTOS	Real Time Operating Systém (operační systém reálného času)
MAC	Media Access Control (podvrstva linkové vrstvy zajišťující adresování)
CRC	Cyclic Redundancy Check (cyklický redundantní součet)
TCP	Transmission Control Protocol (protokol transportní vrstvy)
IP	Internet Protocol (protokol síťové vrstvy)
SU	Strojové učení, ang. *Machine Learning*

RS	Rozhodovací stromy
SVM	Supprot Vector Machine (metoda používaná v SU pro klasifikaci)
RDF	Radial Basis Function (Gaussovo jádro)
OC	Ordinery Clock (typ hodin PTP protokolu)
BC	Boundary Clock (typ hodin PTP protokolu)
TC	Transparent Clock (typ hodin PTP protokolu)
BMC	Best Master Clock (algoritmus pro vývěr nejlepších hodin PTP)
MII	Media Independent Interface (rozhraní v Ethernetu)
PEx	Processor Expert Software
RX	označení pinu pro příjem
FFT	Fast Fourier Transform (rychlá Fourierova transformace)
PTPv2	Precision Time Protocol (druhá verze PTP protokolu)

Seznam obrázků

Seznam tabulek