VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

KOMPRESE SIGNÁLU EKG

ECG DATA COMPRESSION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PATRIK NÉMETH**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. ZDENĚK VAŠÍČEK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

147093

Ústav:	Ústav počítačových systémů (UPSY)
Student:	**Németh Patrik, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Kybernetická bezpečnost
Název:	**Komprese signálu EKG**
Kategorie:	Zpracování signálů
Akademický rok:	2022/23

Zadání:

1. Seznamte se s problematikou měření srdeční aktivity za pomocí elektrokardiografie (EKG) a algoritmy používanými v této oblasti pro bezetrátovou a ztrátovou kompresi EKG signálů. Seznamte se s instrukční sadou mikrokontrolerů ARM řady Cortex M3 a M4F.
2. Na základě rešerše literatury navrhněte algoritmus pro kompresi EKG signálu vhodný pro mikrokontrolery na bázi ARM.
3. Zpracujte studii pokrývající bod 1 a 2 zadání.
4. Navržený algoritmus implementujte formou prototypu v jazyce Python a na vhodné datové sadě ověřte jeho korektní funkci.
5. Implementujte algoritmus v jazyce C a na zvolené platformě využívající jádro ARM bez a s FPU jednotkou vyhodnoťte parametry.
6. Ověřte funkčnost navrženého řešení. Diskutujte parametry navrženého řešení a případná rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodu 1 až 3 zadání.

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

Vedoucí práce:	**Vašíček Zdeněk, doc. Ing., Ph.D.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	31.10.2022

Abstrakt

Práca sa zaoberá tématikou kompresie EKG signálu na mikrokontroléroch. Rozoberá sa tématika fyziológie srdca a zachytávania srdcovej aktivity pomocou EKG. Uvádza sa prehľad rôznych kompresných metód s ohľadom na možnosť využitia týchto metód na systémoch s obmedzeným výkonom a pamäťou. Práca načŕta prehľad architektúry a inštrukčnej sady procesorov ARM a následne navrhuje a implementuje kompresnú metódu na základe existujúcich prístupov. Prototyp v jazyku Python a implementácia pre mikrokontrolér v jazyku C sú vyhodnotené a výsledky diskutované.

Abstract

This work tackles the subject of ECG data compression on microcontrollers. First, the physiology of the heart and recording of the heart's activity is explored. Then the work provides an overview on various compression methods with a focus on their usability on systems with limitied computing power and memory. This work also outlines the architecture and instruction set of ARM processors and then proposes and implements a compression algorithm based on existing research. A Python prototype and a C implementation for microcontrollers are both evaluated and the results are then discussed.

Kľúčové slová

EKG, kompresia, ARM, Cortex-M, mikrokontrolér, lineárna predikcia, C, Python

Keywords

ECG, compression, ARM, Cortex-M, microcontroller, linear prediction, C, Python

Citácia

NÉMETH, Patrik. *Komprese signálu EKG*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Zdeněk Vašíček, Ph.D.

Komprese signálu EKG

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Zdeňka Vašíčka. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Patrik Németh
10. mája 2023

Poďakovanie

Rád by som poďakoval môjmu vedúcemu práce doc. Ing. Zdeňkovi Vašíčkovi Ph.D. za jeho výborné rady a podporu počas tvorby diplomovej práce.

Obsah

Kapitola 1

Úvod

Srdcové ochorenia sú jednými z najčastejších zdravotných komplikácii moderného sveta. Či už sa jedná o vrodené vady alebo o komplikácie spôsobené rôznymi okolnosťami, kardiológovia musia využiť každý možný zdroj informácií o fungovaní srdca pacienta pre určenie čo najpresnejšej diagnózy. Elektrokardiogramy sa pre účely sledovania fungovania srdca používajú už vyše jedného storočia a prístupy k ich zaznamenávaniu sa časom vyvíjajú. Pôvodný prístup zistenia okamžitého stavu srdca zo záznamu EKG sa časom vyvinul a rozvetvil do dlhodobého monitorovania pomocou Holter zariadení. Oba z týchto prístupov majú svoje výhody, avšak priebežné preventívne Holter monitorovanie sa rýchlo rozširuje s rozmachom kompaktných bezdrôtových zariadení. Takéto zariadenia sú schopné bez akéhokoľvek zásadného obmedzenia aktivít nositeľa zaznamenávať potenciálne životne dôležité dáta o fungovaní srdca.

Záznamy EKG zaberajú značný úložný priestor nezávisle od spôsobu získania signálu a tématike kompresie týchto signálov sa je venované už dlhú dobu. Kompresia má veľký význam aj zrovna pri uvedených dlhodobých monitorovacích Holter zariadeniach, ktoré majú vysoko obmedzený úložný priestor, prípadne rýchlosť spojenia so vzdialeným úložným zariadením, na ktoré svoje dáta odosielajú. Je preto vhodné signál okamžite po jeho nasnímaní komprimovať. Holter zariadenia sú však obmedzené aj v iných ohľadoch. Vzhľadom na kompaktnosť takýchto zariadení sú výrazne obmedzené výpočtové a pamäťové možnosti, a teda je nutné navrhovať metódy kompresie, ktoré sú schopné získaný signál komprimovať v dostatočne krátkom čase a s využitím čo najmenšieho množstva pamäte. Táto práca sa zaoberá návrhom a implementáciou zrovna takého algoritmu, pričom cieľová platforma pre navrhnutý algoritmus sú procesory série ARM Cortex-M.

V práci je najskôr rozobraná základná fyziológia srdca, ktorá je využívaná pri získavaní EKG signálu. Uvedené sú aj vlastnosti EKG signálu a jeho štruktúra. V kapitole 3 je následne rozoberaná problematika kompresie. Sú tu uvedené rôzne objektívne metriky kvality, prípadne efektívnosti kompresie a základné prístupy ku kompresií. Následne je v kapitole 4 uvedený prehľad konkrétnych algoritmov využívaných pre kompresiu EKG. Kapitola 5 uvádza prehľad cieľovej architektúry ARM Cortex-M a následne v kapitole 6 je načrtnutý navrhovaný algoritmus. Ďalej je v kapitole 7 popísaná implementácia návrhu vo forme Python prototypu a funkčnej C implementácie pre mikrokontrolér a nakoniec je v kapitole 8 uvedené vyhodnotenie týchto implementácií s diskusiou o navrhnutom riešení.

Kapitola 2

Srdce a elektrokardiogram

Táto práca sa venuje spracovaniu a manipulácii signálu elektrokardiogramu (EKG), a preto je vhodné poskytnúť úvod do ním spojenej tématiky. Táto kapitola je venovaná fyziológii ľudského srdca (sekcia 2.1) a snímaniu elektrických signálov generovaných srdcom a prístrojom, ktoré ich zaznamenávajú (sekcia 2.2). Ak nie je uvedené inak, táto kapitola čerpá zo zdroja [23].

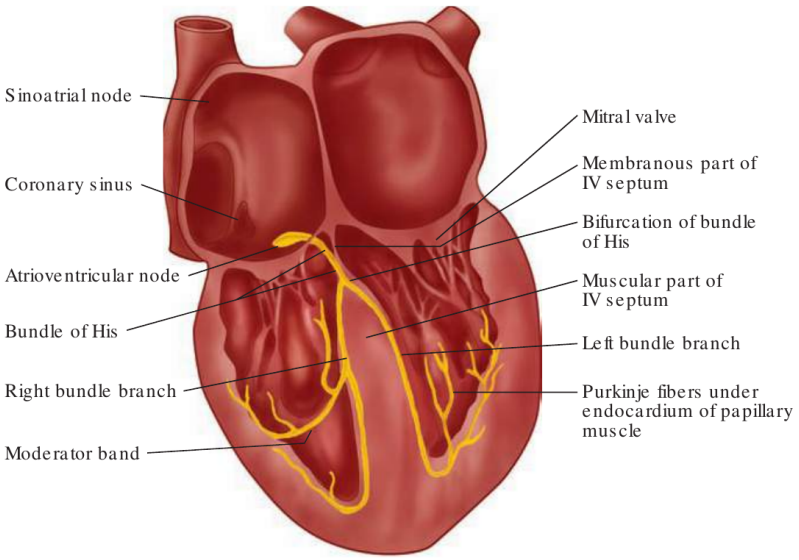
2.1 Fyziológia srdca

Srdce je dutý svalový orgán tvaru obráteného kužeľa nachádzajúci sa v hrudníkovej dutine, typicky uložený medzi pľúcami a bližšie k ľavej strane tela. Obsahuje štyri oddelené dutiny: *pravú a ľavú predsieň* a *pravú a ľavú komoru*. Ľavé a pravé dutiny sú od seba oddelené svalovou priečkou (ďalej *septum*). Predsiene a komory nachádzajúce sa na rovnakej strane sú prepojené chlopňami. Chlopne sú uzatvárané papilárnymi svalmi, ktorých účelom je udržať chlopne v uzavretej polohe počas *systoly*. Systola označuje fázu búšenia srdca keď sa srdcový sval stiahne a vytláča krv zo svojich komôr.

2.1.1 Prevodový systém srdca

Srdce disponuje tzv. *prevodovým systémom srdca* (angl. cardiac conduction system alebo impulse-conducting system), ktorý je zobrazený na obrázku 2.1. Tento systém je zložený zo špecializovaných buniek, ktoré inicializujú a koordinujú kontrakcie svalov jednotlivých častí srdca, čím riadia jeho búšenie. Za typických podmienok je elektrický impulz pre kontrakciu inicializovaný *sinoatriálnym uzlom* (SA uzol), ktorý sa nachádza vo vrchnej časti pravej predsiene. V spodnej časti medzipredsieňového septa sa nachádza *atrioventrikulárny uzol* (AV uzol). V tomto uzle sa vyslaný impulz v zdravom srdci oneskorí zhruba o 8 stotín sekundy.

Nižšie pod AV uzlom sa nachádza *Hisov zväzok*, ktorý perforuje medzikomorové septum a rozdvojuje sa do dvoch vetiev – ľavej a pravej. Tieto vetvy sa naďalej rozdeľujú do *Purkyňových vlákien* distribuujúcich elektrické signály do papilárnych a komorových svalov. Elektrické impulzy distribuované *His-Puryňovým systémom* sa najprv privádzajú do papilárnych svalov a až následne do komorových svalov. Takto sa zabezpečí, že sa chlopne pomocou papilárnych svalov uzavrú pred započatím systoly a neumožnia spätný prúd krvi do predsiení srdca počas systoly.



Obr. 2.1: Hlavné komponenty prevodového systému srdca. Prevzaté z [23].

2.1.2 Elektrofyziológia

Kontrakcie srdca sú závislé na organizovanej propagácii elektrických impulzov pozdĺž prevodového systému. Markantom elektrickej stimulácie je tzv. *akčný potenciál*, ktorý je vytváraný tokom iónov do a zo srdcových buniek. Srdce obsahuje tri typy buniek schopných elektrickej stimulácie:

- kardiostimulačné bunky (SA a AV uzly),
- špecializované vysokovodivé tkanivá (Purkyňove vlákna),
- svalové bunky predsiení a komôr.

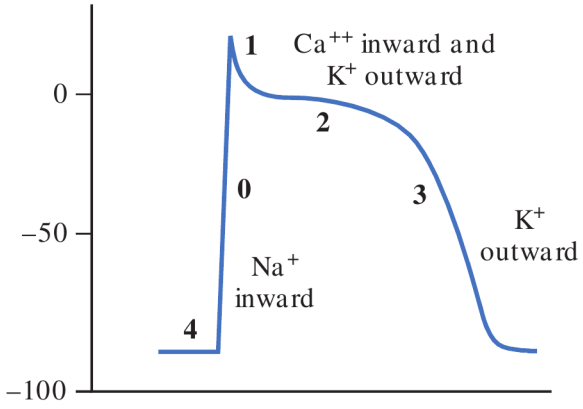
Sarkolema[1] všetkých z týchto buniek je nepriepustná pre ióny. Presun iónov naprieč bunkovou stenou je však zabezpečený bielkovinami roztrúsenými v sarkoleme, ktoré vytvárajú tzv. *iónové kanály*. Tok iónov cez bunkovú stenu vytvára procesy *depolarizácie* a *repolarizácie* buniek v srdci a zmeny v napätí spôsobené týmito procesmi je možné zachytiť elektrokardiografiou.

Depolarizácia je proces elektrického výboja bunky, spôsobujúci zmenu v elektrickom náboji naprieč jej membránou. Vďaka tejto zmene náboja sa umožní prepúšťanie iónov sodíka a vápnika cez membránu bunky (pomocou iónových kanálov). Vápnik sa následne začne viazať na bielkoviny v bunke, čo spôsobí kontrakciu svalovej bunky.

Repolarizácia je naopak proces návratu bunky do kľudového stavu, v ktorom je vnútro bunky nabité negatívne oproti vonkajšku. Do kľudového stavu sa bunka dostane vypudením pozitívnych iónov vstrebaných počas depolarizácie. Proces repolarizácie je pomalší, než depolarizácia a vytvára menší rozdiel potenciálov. Tieto dva procesy vytvárajú akčný potenciál, ktorý je možné definovať ako sekvenciu rýchlych zmien v napätí naprieč membránou bunky [15].

Akčný potenciál má 5 fáz číslovaných od 0 do 4 a jeho priebeh je znázornený na obrázku 2.2. Fáza 4 reprezentuje kľudový stav nestimulovanej bunky. V tomto stave je mem-

[1] Sarkolema je membrána na povrchu svalových vlákien.



Obr. 2.2: Priebeh akčného potenciálu v čase. Označené sú fázy 0 až 4 a smer toku iónov do a z bunky v jednotlivých fázach. Prevzaté z [23].

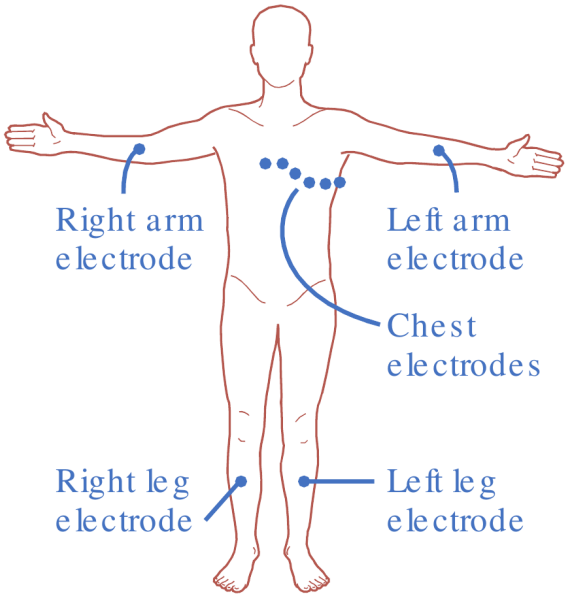
bránové napätie približne −90 mV a sodíkové a vápnikové iónové kanály sú uzavreté. V prípade, že sa membránový potenciál z akéhokoľvek dôvodu stane menej negatívnym, tak sa sodíkové kanály začnú otvárať a zaháji sa depolarizácia. Otvorením iónových kanálov sa umožní pozitívnym sodíkovým iónom (Na^+) rýchlo vnikať do negatívne nabitého vnútra bunky. Zvyšujúcou sa koncentráciou Na^+ sa membránový potenciál ďalej stáva menej negatívnym, čo podporuje ďalší vnik Na^+ do bunky. Behom tejto fázy, označovanej fáza 0, sa membránové napätie prudko zvýši a preklopí na pozitívne pri približne 20 mV. V tomto stave sa koncentračný gradient[2] Na^+ začne redukovať a iónové kanály sa začnú uzatvárať, čím sa začína proces repolarizácie, a teda vráteniu sa ku kľudovému napätiu [37]. Počas fázy 1 sa membránový potenciál vráti na 0 mV. Táto mierna zmena napätia je tvorená tokom iónov draslíka von z bunky cez dočasne otvorené draslíkové kanály. Vo fáze 2, tiež často označovanej ako plató pre jej plochý priebeh, sa návrat napätia ku kľudovej úrovni výrazne spomalí. Zdrojom tohto ustálenia membránového potenciálu je súčasný tok pozitívnych iónov draslíka von z bunky a pozitívnych iónov vápnika do bunky. Výmena pozitívnych iónov medzi exteriérom a interiérom bunky udržiava rozdiel potenciálov naprieč membránou takmer nulový. Ku koncu tejto fázy sa kanály pre prenos vápnika začnú postupne uzatvárať a úbytok iónov draslíka začne prevyšovať vstrebávanie iónov vápnika. Rozdiel potenciálov sa teda opäť začne zvyšovať. Nasleduje fáza 3, ktorá činí poslednú fázu pred návratom napätia na kľudovú hodnotu. Úbytok draslíka výrazne prevýši vstrebávanie iných katiónov, a teda sa membránový potenciál úmerne zníži až na kľudovú hodnotu −90 mV. Počas kľudového stavu bunky sa úrovne koncentrácií katiónov v rámci a mimo bunky vrátia na úrovne vyžadované pre opakovanie cyklu akčného potenciálu.

Je zrejmé, že procesy zmien elektrickej polarity buniek a kontrakcií srdca sú so sebou úzko spojené. Preto je detekciou elektrických impulzov, ktoré sú generované akčným potenciálom, možné fungovanie srdca študovať. Tieto procesy depolarizácie a repolarizácie buniek nie sú okamžité a ich priebeh naprieč srdcom je možné zaznamenať pomocou elektrokardiogramu.

[2]Gradient medzi oblasťami s vysokou a nízkou koncentráciou častíc – v tomto prípade sa jedná o koncentráciu iónov sodíka mimo bunky a v rámci nej. Častice majú tendenciu sa pasívne rozptyľovať dole koncentračným gradientom, teda z oblasti vysokej koncentrácie do oblasti nízkej koncentrácie.

2.2 Elektrokardiogram

Ako bolo spomenuté v predošlej sekcii, srdcové kontrakcie závisia na organizovanom toku elektrických impulzov naprieč srdcovým svalom. Tieto impulzy spôsobujú depolarizáciu a následnú repolarizáciu svalových buniek, pričom kumuláciu týchto zmien v polarite buniek srdca je možné detegovať na povrchu pokožky pomocou elektrokardiogramu (EKG).



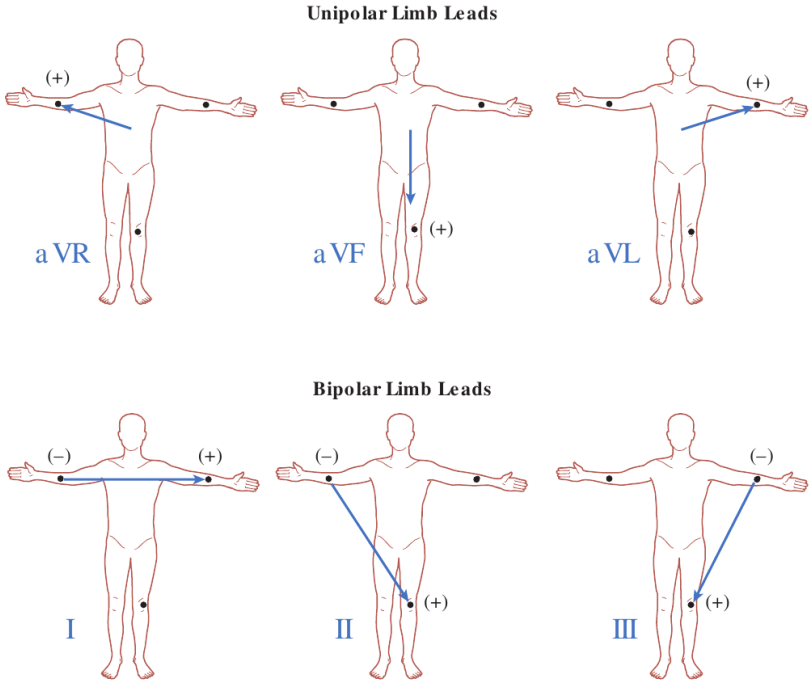
Obr. 2.3: Zobrazenie umiestnení jednotlivých elektród elektrokardiogramu na ľudskom tele. Prevzaté z [23].

2.2.1 Zvody EKG

V kľudovom stave srdca je povrch svalových buniek homogénne pozitívne nabitý v porovnaní s vnútrom bunky. Vďaka tomuto ekvilibriu náboja naprieč bunkami nie je v srdci žiadny rozdiel potenciálov, ktoré by EKG dokázalo zachytiť. Počas systoly sa však cez srdce preženie niekoľko vĺn depolarizácií a repolarizácií, ktoré toto ekvilibrium narušia. Tým sa vytvorí rozdiel potenciálov detegovateľný v rôznych osiach pomocou EKG. Tieto osi sa nazývajú *zvody* a sú tvorené desiatimi elektródami (rozmiestnenie elektród je zobrazené na obrázku 2.3). Pri úplnom EKG je zvodov 12 - šesť vo *frontálnej rovine* a šesť v *transverzálnej rovine* (tiež nazývaná horizontálna)[3]. Zvody reprezentujú rozdiel potenciálov medzi dvoma referenčnými bodmi. Štandardne sa zaznamenáva aktivita na troch *bipolárnych končatinových zvodoch* (označované **I**, **II** a **III**), šiestich *unipolárnych hrudných zvodoch* ($\mathbf{V_1}$ až $\mathbf{V_6}$) a troch *augmentovaných končatinových zvodoch* (**aVR**, **aVF** a **aVL**) [37].

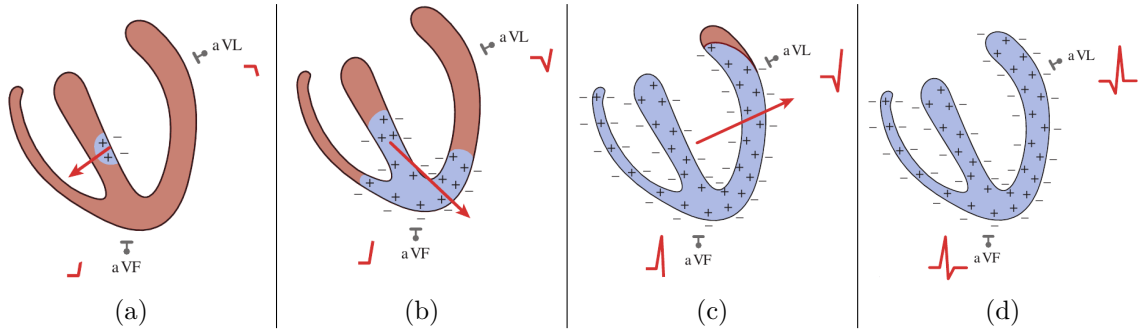
Bipolárne zvody zachytávajú rozdiel potenciálov medzi dvoma elektródami, kde jeden z nich pôsobí ako pozitívny terminál a druhý ako negatívny. Bipolárnymi zvodmi sa meria srdcová aktivita iba vo frontálnej rovine a ich osi merania je možné vidieť na obrázku 2.4. Tieto zvody vytvárajú tzv. *Einthovenov trojuholník*. Unipolárne hrudné zvody využívajú *Wilsonov centrálny terminál* (WCT), ktorý slúži ako referenčná negatívna elektróda. Tento

[3] Jedná sa o dve z troch anatomických rovín: frontálna rozdeľuje telo na prednú a zadnú polovicu, transverzálna na hornú a dolnú polovicu. Pre úplnosť ešte existuje sagitálna rovina rozdeľujúca telo na ľavú a pravú polovicu.



Obr. 2.4: Zobrazenie končatinových zvodov. Prevzaté z [23].

terminál je definovaný ako priemer potenciálov na oboch rukách a ľavej nohe. Všetkých šesť hrudných unipolárnych zvodov využíva tento referenčný terminál. Upravená verzia WCT, tzv. *Goldbergerov terminál* (GT), je využitý pri unipolárnych končatinových zvodoch, ktoré majú prívlastok augmentované (alebo predĺžené). Augmentovanými sa nazývajú z dôvodu vyššej senzitivity na napätie, čím vytvárajú zreteľnejšie výchylky v signáli EKG. Tieto zvody využívajú rovnaké elektródy ako WCT, avšak iba dva z nich sú využité k vytvoreniu referenčného negatívneho terminálu a tretí slúži ako pozitívny (viď zobrazenie unipolárnych končatinových zvodov na obrázku 2.4).



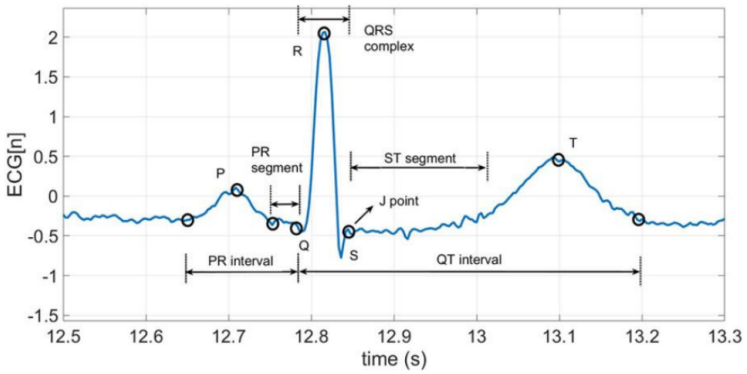
Obr. 2.5: Znázornenie dopadu rovnobežnosti smeru šírenia vlny depolarizácie s osami zvodov na zachytávaný signál EKG. Prevzaté z [23].

2.2.2 Signál elektrokardiogramu

Záznam EKG je tvorený krivkou, ktorá zaznamenáva zmeny napätia v čase. V digitálnych systémoch sa jedná o periodicky vzorkované napätie tvoriace diskrétny signál. Výkyv

krivky smerom nahor sa zachytáva v prípade, že smer vlny depolarizácie vedie k pozitívnej elektróde daného zvodu. Výkyv nadol nastáva naopak v prípade, že vedie k negatívnej elektróde. Výška výkyvu závisí na rovnobežnosti šírenia vlny depolarizácie s osou zvodu. Vlna rovnobežná s osou zvodu vytvára maximálny výkyv, pričom vlna kolmá na os zvodu nevytvára žiaden výkyv. Tento vzťah je načrtnutý na obrázku 2.5, kde sú zobrazené dve komory srdca oddelené septom. Zobrazené sú aj pozitívne elektródy zvodov aVL a aVF a na nich zachytávané napätie. Na obrázku je možné vidieť dopad smeru šírenia depolarizácie z AV uzla na rôznymi zvodmi zachytávané napätie. Každý z dvanástich vyššie uvedených zvodov má rozdielnu os, vďaka čomu je zo získaných EKG signálov možné vytvoriť viacrozmernú interpretáciu priebehu systoly v srdci. Zobrazením osí končatinových zvodov získame *hexaxiálny referenčný systém*.

Signál EKG je typicky zobrazovaný ako jednorozmerný spojitý signál. V signáli EKG je v rámci jednej periódy možné identifikovať niekoľko výchyliek (viď obrázok 2.6). Tieto výchylky sú označované ako vlny a sú pomenované v poradí od začiatku srdcového cyklu nasledovne: *P*, *Q*, *R*, *S*, *T* a *U*. Vlny Q, R a S tvoria *QRS komplex*. Vlna U často nie je na zázname prítomná a predpokladá sa, že je tvorená v dôsledku neskorých štádií repolarizácie komôr. Jednotlivé výchylky zodpovedajú konkrétnym fázam akčného potenciálu v rôznych častiach srdcového svalu. Fáza 0 akčného potenciálu v bunkách predsiene je zaznamenaná ako vlna P, pričom rovnaká fáza v bunkách komory je reprezentovaná QRS komplexom [37, 23]. Signál je možné rozdeliť na segmenty a intervaly. Segmenty sú definované ako regióny medzi dvoma vlnami, pričom intervaly sú časové úseky obsahujúce jeden segment a prinajmenšom jednu vlnu [13].



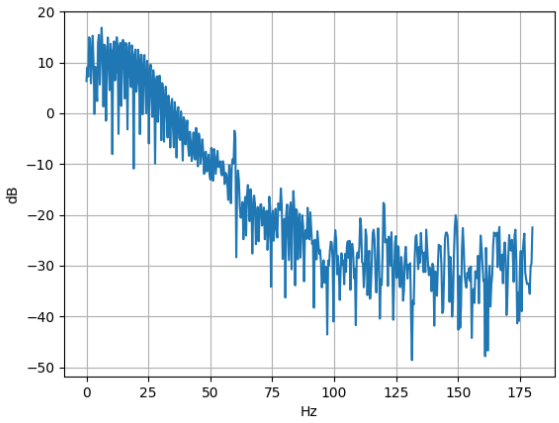
Obr. 2.6: Zobrazenie typickej periódy záznamu EKG. Zobrazené sú vlny P až T a segmenty a intervaly významné pre diagnostické účely. Prevzaté z [37].

Diagnóza pomocou EKG je vysoko vizuálna a kardiológovia sa spoliehajú na kvalitnú vizuálnu reprezentáciu EKG signálu. Počas čítania EKG sa skúma štruktúra signálu, čo zahŕňa niekoľko ukazovateľov, ako sú doba trvania konkrétnych intervalov a segmentov, amplitúdy jednotlivých vĺn, prípadne tvar a jemnejšia štruktúra vĺn a komplexov. Pre zdravé srdce je možné definovať špecifické rozmedzia a vzory pre tieto ukazovatele, od ktorých sa konkrétne patológie líšia veľmi špecificky. Pre jednoduché načrtnutie významu štruktúry signálu je uvedená tabuľka 2.1, uvádzajúca očakávané normálne hodnoty trvania istých intervalov a možné patológie pri ich odchýlení od normy [23].

Mnohé patológie sa teda v súčasnosti detegujú na základe nízkofrekvenčných komponent signálu. V kardiologických kruhoch sa vedú debaty o užitočnosti vyšších frekvencií EKG signálu, pričom rôzne experimenty a výskumy do problematiky udávajú rôzne výsledky.

Tabuľka 2.1: Krátky prehľad možných patológií pri líšiacich sa trvaní intervalov od normy. Prevzaté z [23].

Interval	Normálne trvanie	Znížené trvanie	Zvýšené trvanie
PR	$0.12 \leq \text{PR} \leq 0.2$ s	Syndróm preexcitácie komôr	AV blokáda prvého stupňa
QRS	≤ 0.1 s	-	Komorová extrasystola
QT	$\frac{QT}{\sqrt{R-R}} \leq 0.44$ s	Tachykardia	Srdcová ischémia



Obr. 2.7: Frekvenčné spektrum záznamu `100` databázy MIT-BIH [25].

V modernej medicíne sa však typicky obmedzujú zachytávané frekvencie pásmovým priepustom od 0.5 Hz do 150 Hz [34]. Referenčné databázy EKG signálov sa pri digitalizácii tiež často obmedzujú na vzorkovaciu frekvenciu do 400 Hz [25, 33]. Existujú však aj databázy s vysoko detailnými záznamami, ako je napríklad PTB diagnostická EKG databáza, ktorá poskytuje aj záznamy vzorkované na 10 kHz [6]. Je nutné poznamenať, že Nyquistov vzorkovací teorém udáva, že skutočné frekvencie digitalizovaného signálu sa rovnajú nanajvýš polovici vzorkovacej frekvencii. Tento vzťah je definovaný vzorcom 2.1, kde f_s je vzorkovacia frekvencia a f_M je maximálna frekvencia frekvenčného pásma vzorkovaného signálu.

$$f_s > 2f_M \tag{2.1}$$

Pri zobrazení frekvenčného spektra je tiež zrejmé, že najväčšia časť signálu je koncentrovaná v nízkych frekvenciách (obrázok 2.7). Zaujímavým fragmentom v EKG záznamoch získaných zo zariadení pripojených do elektrickej siete sú frekvencie danej siete. Na obrázku 2.7 je zrejmé navýšenie sily signálu na frekvencii Americkej elektrickej siete 60 Hz a na harmonických 120 Hz.

Medicínske zariadenia spracúvajúce EKG signál musia dbať na požiadavky kardiológov na kvalitu signálu po jeho digitálnom spracovaní. Z tohto dôvodu sú v medicíne lepšie vnímané bezstratové metódy spracovania a archivácie EKG záznamov, pretože zachovávajú všetky štrukturálne vlastnosti pôvodného signálu. Rôzne metódy stratovej a bezstratovej kompresie EKG signálu sú uvedené v kapitole 4.

2.2.3 Holter monitorovanie

Typické EKG záznamy sa vytvárajú v nemocničnom prostredí v kľudovom stave pacienta a za prítomnosti lekárov alebo lekárskych technikov. Získanie takéhoto záznamu zväčša trvá nanajvýš 15 minút, pričom záznam samotný trvá niekoľko sekúnd, prípadne obsahuje iba malé množstvo zachytených tlkotov srdca [20]. Takéto záznamy však nemusia poskytnúť dostatočné informácie o stave srdca, obzvlášť ak anomálie spôsobujúce ťažkosti u pacienta sú tranzientné a akurát sa nevyskytnú počas nemocničného EKG merania (jedná sa často o arytmie, akou je napríklad fibrilácia predsiení) [40]. V takýchto prípadoch je vhodné použiť ambulantné kardiografické monitorovanie srdca, najčastejšie nazývané Holter monitorovanie.

Norman Holter prvýkrát popísal praktické využitie jeho prenosného systému v roku 1961. Jednalo sa o zariadenie vážiace približne jeden kilogram, ktoré bolo schopné zaznamenávať srdcovú aktivitu na magnetickú pásku s kapacitou pre 10 hodinový záznam, a ktorý bol schopný zaznamenávať signál zachytený jedným zvodom[4] [17]. Holter systémy sa časom vyvinuli v mnohých ohľadoch a v súčasnosti ich je možné rozdeliť do kategórií podľa viacerých kritérií, ako napríklad do záznamov krátkodobých (24 až 48 hodinové monitorovanie) a dlhodobých (nad 48 hodín). Počet zvodov sa pri rôznych systémoch tiež líši. Môže sa jednať o tzv. *patch* monitorovacie zariadenie, ktoré zachytáva signály jedným zvodom, a pri ktorom sa typicky jedná o dlhodobé nosenie s bezdrôtovým odosielaním signálu na záznamové zariadenie. Prípadne môže ísť o dvoj-, troj-, sedem-, ale aj dvanásť-zvodové prístroje, v závislosti od lekárom vyžadovaných informácií o stave srdca. Väčšie množstvo zvodov totiž tvorí detailnejší obraz o fungovaní srdca, ktorý je potrebný pri určitých diagnózach [19, 32].

Vzorkovacia frekvencia EKG zariadení nie je nijak obmedzená a pre určenie jednotlivých diagnóz sa dajú využiť rôzne frekvencie. Vzorkovacia frekvencia sa môže pohybovať v rozmedzí niekoľkých stoviek Hz, až do 1000 Hz na jeden kanál [22, 37]. Aj z tohto dôvodu je v prenosných Holter systémoch vhodné implementovať mechanizmy schopné zachytávaný signál komprimovať, keďže vyššia vzorkovacia frekvencia znamená väčší počet ukladaných vzoriek. Kompresia signálu má pri Holter systémoch význam vo viacerých ohľadoch, napríklad pri ukladaní dlhodobého záznamu, prípadne pri vysielaní signálu bezdrôtovým spojením (časté pri spomenutých patch zariadeniach). Dôraz sa však musí klásť na vyvažovanie požiadaviek na Holter zariadenie. Nesmie napríklad dôjsť k predčasnému vybitiu zariadenia v dôsledku vysokej spotreby energie kompresným systémom. Aj z tohto dôvodu sa už dlhšiu dobu vykonáva výskum do energeticky efektívnych kompresných algoritmov určených pre EKG signál [24].

[4] Holter v článku predstavujúcom jeho systém popisuje využívanie zvodu V_1.

Kapitola 3

Kódovanie a kompresia dát

Kódovanie je využívané od pradávna. Dalo by sa povedať, že od počiatku písma, ak nie skôr, keď sa začali kódovať zvuky reči do vizuálnych znakov. Po čase sa začali objavovať aj prvé známky kompresie, ako napríklad v starovekom Grécku, kde sa manuskripty písali na drahý a ťažko dostupný papyrus. Prípadne v stredoveku sa na mince razili akronymy namiesto plného textu z dôvodu nedostatočného priestoru. Je možné teda vyvodiť, že kompresia je významná na miestach, kde sa pracuje s priestorom obmedzenej kapacity. Efekt kompresie na kapacitu úložného priestoru je načrtnuteľný jednoduchým príkladom, kde zmenšením využitého priestoru pre uloženie danej informácie o polovicu sa efektívne zdvojnásobí kapacita úložného priestoru. V modernom ponímaní kompresie dát sa kompresia berie ako obecná redukcia veľkosti digitálnych, teda binárnych, dát. Kompresia je teda aplikovateľná či už na bloky dát, alebo aj na dátové toky [31].

Táto práca sa zaoberá kompresiou aj kódovaním, preto je vhodné definovať rozdiel medzi týmito výrazmi. *Kódovanie dát* je vzájomne jednoznačné mapovanie symbolov zo zdrojovej množiny symbolov na symboly cieľovej množiny. *Kompresia dát* je taký proces kódovania dátového toku, v dôsledku ktorého má výsledný dátový tok menšiu veľkosť. Výsledný dátový tok bol pri tom vytvorený zredukovaním redundancie v zdrojových dátach. Je to teda proces prevedenia reprezentácie dát z neefektívnej na efektívnu [12, 30].

Kompresné metódy sú obecne delené na dve hlavné kategórie; *bezstratové* (angl. lossless) a *stratové* (angl. lossy). Bezstratová kompresia nevnáša do dátového toku žiadne chyby a po následnej dekompresii[1] je rekonštruovaný dátový tok identický so zdrojovým. Stratová kompresia obyčajne dosahuje lepšej kompresie za cenu straty nejakého množstva informácie z dátového toku. Voľba typu kompresie je závislá na aplikácii, v ktorej bude využívaná. Napríklad audio a video dáta je možné vo veľa prípadoch (napr. archivácia domácich videí, filmy) komprimovať stratovo, pretože ich účel nevyžaduje dokonalú kvalitu rekonštruovaného signálu. Pre niektoré účely sa však preferuje, alebo aj vyžaduje, zachovanie všetkých originálnych informácii z pôvodného dátového toku. K týmto patrí napríklad kompresia textu, kde aj zmena jediného bitu je okamžite zrejmá [30]. V medicíne sa preferujú bezstratové metódy a v niektorých prípadoch na stratovo komprimované dáta nie je braný zreteľ [36, 27]. Rôzne metódy stratovej kompresie zavádzajú rôznu mieru chyby do dátového toku, a preto je vhodné využiť metriky pre kvantifikáciu miery skreslenia zavedeného do dát, resp. signálu [30]. Prehľad rôznych prístupov k vyhodnocovaniu kompresných metód je uvedený v sekcii 3.1. V tejto kapitole sa uvedú definície späté s problematikou kompresie a načrtnú sa základné metódy kompresie.

[1] V texte sú výrazy dekompresie signálu a rekonštrukcie signálu ekvivalentné a často zamieňané.

3.1 Vyhodnocovanie kompresie

Kompresná výkonnosť, teda miera dosiahnutej kompresie, je typicky vyjadrovaná niekoľkými metrikami. Medzi najpoužívanejšie patria *kompresný pomer* (CR z angl. Compression Ratio) a *kompresný faktor* (CF z angl. Compression Factor). Miera CR definuje pomer veľkosti výstupného dátového toku k vstupnému dátovému toku, pričom CF udáva koľkonásobne je vstupný dátový tok objemnejší, než výstupný [12]. Vzťahy pre CR a CF sú vyjadrené vzorcami 3.1 a 3.2. Tieto hodnoty sú si navzájom prevrátené (recipročné). Kompresný faktor je zväčša lepšie zrozumiteľnou mierou z vyššie uvedených, pretože vyššia hodnota CF znamená efektívnejšiu kompresiu dát [12]. Z definícií je zrejmé, že pre dosiahnutie kompresie na dátovom toku, hodnota CR musí byť menšia, než 1 a hodnota CF naopak vyššia, než 1. V opačnom prípade buď nedochádza ku kompresii (CR = CF = 1), prípadne nastáva tzv. *expanzia* (CR > 1 a CF < 1), kde veľkosť výstupného dátového toku je väčšia, než zdrojového. Expanzia typicky nastáva v prípade, že množstvo odstránenej redundancie je prevýšené množstvom pridaných metadát do dátového toku aby bolo možné vykonať dekompresiu alebo dekódovanie.

$$\mathrm{CR} = \frac{\text{veľkosť výstupného toku}}{\text{veľkosť vstupného toku}} \tag{3.1}$$

$$\mathrm{CF} = \frac{\text{veľkosť vstupného toku}}{\text{veľkosť výstupného toku}} \tag{3.2}$$

Ďalšou metrikou kompresnej výkonnosti je *priemerná dĺžka slova* (avL z angl. average length), ktorá berie ohľad na počet vzoriek v zdrojovom signáli (vzorec 3.3) [27]. Výsledné číslo udáva priemerný počet bitov komprimovaného toku na jednu vzorku zdrojového signálu.

$$\mathrm{avL} = \frac{\text{veľkosť výstupného toku}}{\text{počet vzoriek v signáli}} \quad [\text{bit/vzorka}] \tag{3.3}$$

V prípade práce so signálovými tokmi je možné zaviesť metriky poukazujúce na množstvo prenesených komprimovaných dát za jednotku času. Pre tento účel je vhodnou metrikou tzv. *rýchlosť prenosu komprimovaných dát* (CDR z angl. Compressed Data Rate), ktorá udáva aký objem komprimovaného dátového toku je prenesený za sekundu. Metriku CDR je možné vypočítať viacerými spôsobmi (vzorce 3.4 a 3.5). Rozlíšenie vo vzorci 3.5 je chápané ako vzorkovacie rozlíšenie vstupného toku a je definované ako počet bitov na vzorku [27].

$$\mathrm{CDR} = \frac{\text{vzorkovacia frekvencia} \times \text{veľkosť výstupného toku}}{\text{doba trvania signálu}} \quad [\text{bit/sekunda}] \tag{3.4}$$

$$\mathrm{CDR} = \frac{\text{vzorkovacia frekvencia} \times \text{rozlíšenie}}{\mathrm{CF}} \quad [\text{bit/sekunda}] \tag{3.5}$$

3.1.1 Vyhodnotenie kvality signálu po stratovej kompresii

Ako bolo spomenuté, kompresia môže byť bezstratová alebo stratová. V prípade bezstratovej kompresie nie je nutné vyhodnocovať podobnosť rekonštruovaného signálu s originálnym, pretože počas kompresie nedošlo ku strate informácií v signáli. Stratová kompresia však vytvára komprimované dáta, ktoré sa po rekonštrukcii môžu v rôznych ohľadoch zásadne líšiť od pôvodných dát. V tejto sekcii je uvedený prehľad často využívaných všeobecných metrík hodnotenia kvality stratovej kompresie EKG signálu.

Chybový signál (angl. error signal) je veľmi jednoduchý spôsob porovnania pôvodného signálu so signálom po rekonštrukcii. Výpočet chybového signálu je popísaný vzorcom 3.6. Jedná sa o priame porovnanie jednotlivých vzoriek pôvodného signálu (vyjadrený ako x) s rekonštruovaným (vyjadrený ako $\tilde{x}$). Zápis $x[n]$ vyjadruje n-tú vzorku signálu x. Takto vytvorený chybový signál môže slúžiť k numerickému, ale aj vizuálnemu vyhodnoteniu kvality kompresie [27].

$$e[n] = x[n] - \tilde{x}[n] \tag{3.6}$$

Stredná kvadratická chyba (MSE z angl. Mean Square Error) vyjadruje priemer druhej mocniny chyby medzi dvoma signálmi (vzorec 3.7). Hodnota N udáva celkový počet vzoriek signálu. Normalizovaná stredná kvadratická chyba (angl. NMSE), popísaná vzorcom 3.8, limituje amplitúdu signálu do štandardného rozmedzia. Toto sa vykonáva, pretože veľkosti amplitúd signálu sa líšia medzi pacientmi a často aj medzi kanálmi jedného EKG zariadenia [28].

$$\mathrm{MSE} = \frac{\sum_{n=1}^{N} (x[n] - \tilde{x}[n])^2}{N} \tag{3.7}$$

$$\mathrm{NMSE} = \frac{\sum_{n=1}^{N} (x[n] - \tilde{x}[n])^2}{\sum_{n=1}^{N} (x[n])^2} \tag{3.8}$$

Odmocnina strednej kvadratickej chyby (RMS z angl. Root MSE) zachováva pôvodnú jednotku veličiny signálu (Volt v prípade EKG) a je definovaná vzorcom 3.9. Existuje viacero používaných verzií RMS pre vyhodnocovanie kvality stratovej kompresie EKG signálu, ako napríklad priemerovanie cez N vzoriek, prípadne cez $N-1$ vzoriek [27]. Využiť sa môže aj normalizované RMS (NRMSE), ktoré je získané odmocnením NMSE, avšak namiesto tejto miery sa využíva miera percentuálnej chyby, ktorá je popísaná nižšie.

$$\mathrm{RMS} = \sqrt{\frac{\sum_{n=1}^{N} (x[n] - \tilde{x}[n])^2}{N}} \quad [\text{veličina vzorky}] \tag{3.9}$$

Častou metrikou stratovej kompresie je *percentuálna chyba* (PRD z angl. Percentile RMS Difference) nahradzujúce NRMSE, s ktorou sa zhoduje až na dodatočné násobenie 100 (viď vzorec 3.10) [27, 42]. Táto miera vyjadruje percentuálne skreslenie rekonštruovaného signálu oproti originálnemu.

$$\mathrm{PRD} = \sqrt{\frac{\sum_{n=1}^{N} (x[n] - \tilde{x}[n])^2}{\sum_{n=1}^{N} (x[n])^2}} \cdot 100 \quad [\%] \tag{3.10}$$

V prípadoch, že komprimovaný signál obsahuje nenulovú DC zložku, tak je potrebné vykonať normalizáciu aby signál nebol závislý na jeho priemernej hodnote [42]. Normalizácia sa vykonáva odčítaním priemernej hodnoty pôvodného signálu $\bar{x}$ od každej vzorky, ako je znázornené vo vzorci 3.11. Ďalšia normalizácia je vyžadovaná pri signáloch s posunutou nulou, ako je to napríklad pri digitalizovaných signáloch databázy MIT-BIH [25]. Takáto normalizácia (vzorec 3.12) sa vykonáva odčítaním hodnoty posunutej nuly od originálneho signálu (pri signáloch databázy MIT-BIH sa jedná o hodnotu 1024). Obe tieto normalizácie je možné zlúčiť do jednej, v prípade, že v signáli je prítomná aj DC zložka aj posunutá nula. Priemernú hodnotu signálu označuje $\bar{x}$ a hodnotu posunutej nuly P.

$$\mathrm{PRDN}_1 = \sqrt{\frac{\sum_{n=1}^{N} (x[n] - \tilde{x}[n])^2}{\sum_{n=1}^{N} (x[n] - \bar{x})^2}} \cdot 100 \quad [\%] \tag{3.11}$$

$$\mathrm{PRDN}_2 = \sqrt{\frac{\sum_{n=1}^{N}(x[n]-\tilde{x}[n])^2}{\sum_{n=1}^{N}(x[n]-P)^2}} \cdot 100 \quad [\%] \tag{3.12}$$

Hodnota PRD vyjadruje mieru skreslenia rekonštruovaného signálu oproti originálnemu, a teda nízke hodnoty značia signál dobrej kvality. Typicky sú signály s hodnotou do 2 % hodnotené ako vysoko kvalitné signály a do 5 % až 9 % ako signály dobrej kvality, avšak prahová hodnota sa môže meniť v závislosti od kompresnej metódy [27, 28].

Metriky uvedené vyššie sú generalizované metriky vhodné pre hodnotenie kvality signálu ako celku. Pracuje sa v nich s priemernými hodnotami celého signálu a porovnaniami jednotlivých vzoriek naprieč celým signálom. Keďže metódy stratovej kompresie bývajú vysoko špecializované pre ich účel použitia, existuje veľké množstvo spôsobov hodnotenia kvality stratovej kompresie, ktoré berú špeciálny ohľad na komponenty signálu, ktoré je nutné zachovať v čo najvyššej kvalite. V prípade EKG signálu teda môžu byť vyššie uvedené všeobecné hodnotiace metriky zavádzajúce, pretože rôzne časti signálu majú rôzny diagnostický význam. Napríklad ak je signál kompresiou významne skresľovaný výlučne v QRS komplexoch, tak priemerná chyba celkového signálu nemusí byť výrazne navýšená, pretože QRS komplexy tvoria veľmi malú časť EKG signálu. Keďže rôzne časti EKG signálu majú rôzny diagnostický význam, je vhodné takýto signál vyhodnocovať za pomoci diagnostických informácií [1, 27, 42]. Tieto metódy sú výpočtovo náročnejšie a v niektorých prípadoch vyžadujú prítomnosť kardiológa pre správne nastavenie hodnotiacich algoritmov, kvôli čomu sú menej využívané oproti všeobecným metrikám [27].

3.2 Základné metódy kompresie

V tejto sekcii sú popísané niektoré zo základných metód kompresie ako sú uvedené v [30]. Tieto kompresné metódy samé o sebe obvykle neposkytujú vysokú mieru kompresie, ale môžu byť využité k zvýšeniu kompresného pomeru pomocou komplexnejších systémov. Sekcia čerpá z [30].

3.2.1 Run-Length Encoding

Táto metóda využíva sledy identických dát v dátovom toku pre zníženie jeho veľkosti. Základný princíp je veľmi jednoduchý a je možné ho definovať slovne nasledovne: sled symbolov s s dĺžkou sledu n zakóduj ako dvojicu ns. Dekódovanie je podobne triviálne: každú dvojicu ns nahraď sledom symbolov s s dĺžkou sledu n. Sledy symbolov s sa nazývajú behy a hodnota n označuje dĺžku behu. Názov metódy sa skracuje na RLE z anglického Run-Length Encoding. Táto kompresná metóda je nezávislá od typu dát a je možné ju aplikovať v rôznych situáciách, hoci efektivita kompresie je vysoko závislá na štruktúre komprimovaných dát.

V základnej verzii algoritmu, ktorá bola uvedená vyššie, sa pre jednoznačné dekódovanie musí kódovať každý symbol dátového toku nezávisle od dĺžok behov jednotlivých symbolov. Teda reťazec `abbbacc` sa kóduje ako `1a3b1a2c`. Je zrejmé, že v tomto prípade došlo k expanzii dátového toku o jeden symbol kvôli veľkému počtu krátkych behov. Expanzia nastala pri symboloch `a`, pretože ich behy sú kratšie, než počet miest zaberaných symbolom n v dátovom toku. V prípade, že sa počet miest potrebných pre reprezentáciu n rovná dĺžke behu, ako je tomu u behu `c`, tak nedochádza k expanzii, avšak ani ku kompresii.

Problém expanzie je možné vyriešiť vynechaním kódovania krátkych behov, čím sa získa kódové slovo `a3ba2c`. V tomto prípade sa dosiahla kompresia, avšak sa nemusí jednať o dekó-

dovateľné kódové slovo v prípade, že symboly `3` a `2` patria do vstupnej abecedy kódovaných symbolov. Tento problém je možné riešiť doplnením takej značky do kódu, ktorá nepatrí do vstupnej abecedy, napr. symbol `:`. Počas dekódovania sa dĺžka behu v dátovom toku dekóduje iba pri prečítaní tejto značky označujúcej nasledovné symboly ako dvojicu ns. Pri použití značky sa rozšíri veľkosť každého kódovaného symbolu o jeden, čím sa získa `a:3ba:2c`. Opäť sa preto musí predĺžiť minimálna dĺžka behu o jeden symbol, aby sa predišlo expanzii. Po tomto kroku sa získa jednoznačne dekódovateľné kódové slovo `a:3bacc`, ktoré však má rovnakú dĺžku ako vstupný reťazec. Je zrejmé, že takéto kódovanie má isté nedostatky:

- V prípade, že jeden symbol je kódovaný na n_b bitov, tak sa pri použití značky zmenší vstupná abeceda z 2^{n_b} symbolov na $2^{n_b} - 1$, čo má dopad na zníženie rozlíšenia vstupných dát.
- Maximálna dĺžka behu je $2^{n_b} - 1$ symbolov. Túto hodnotu je možné zvýšiť o minimálnu dĺžku behu využitím implicitnej informácie, ktorá je nesená samotnou prítomnosťou behu. Teda ak dekodér narazí na značku, môže automaticky predpokladať minimálnu dĺžku behu a k nemu pripočítať uvedenú hodnotu (vo vyššie uvedenom príklade by sa reťazec kódoval na `a:0bacc`).
- Kódovanie je efektívne pri dlhých behoch symbolov. Z tohto dôvodu je RLE nepraktické napríklad pre priamu kompresiu textu, pretože typicky neobsahuje dlhé behy rovnakých symbolov.

Priemerný kompresný pomer RLE je možné vypočítať pomocou vzťahu 3.13, kde N označuje dĺžku dátového toku s M behmi priemernej dĺžky L. Dĺžka S označuje počet symbolov využitých pre kódovanie jedného behu, tzn. v prípade trojice (značka, dĺžka behu, symbol) je za S dosadená hodnota 3.

$$\frac{N}{N - M(L - S)} \tag{3.13}$$

Dobrým kandidátom pre kompresiu pomocou RLE sú obrazové dáta, ktoré často obsahujú dlhé behy pixelov rovnakých hodnôt. Formáty BMP a TIFF okrem iných metód využívajú aj RLE.

3.2.2 Relatívne kódovanie

Jedná sa o ďalší prístup, ktorý sa často kombinuje s inými metódami kompresie. Relatívne kódovanie, označované aj diferenciálne kódovanie, je využívané ak dátový tok pozostáva zo sledu hodnôt, ktorých susedné hodnoty sa medzi sebou príliš nelíšia. Myšlienka spočíva vo vypočítaní rozdielu medzi súčasnou hodnotou a nasledujúcou hodnotou (vzťah 3.14), pričom veľkosť rozdielu týchto hodnôt je výrazne menší, než hodnoty samotné. Ak tento vzťah aplikujeme na sled čísel [50, 53, 51, 52], získame sled [50, 3, −2, 1]. Je zrejmé, že výstupný komprimovaný tok je nutné inicializovať pôvodnou vysokou počiatočnou hodnotou pre umožnenie dekompresie, avšak nasledujúce hodnoty sa udržiavajú v relatívne malom rozmedzí hodnôt.

$$d[n] = x[n + 1] - x[n] \tag{3.14}$$

V prípade, že rozdiel susedných hodnôt presahuje maximálnu kódovateľnú hodnotu (pri 8 bitovej reprezentácii −128 až 127), je možné do toku zakódovať hodnotu pôvodnú. V takomto prípade je však nutné poskytnúť spôsob, akým sa dajú rozlíšiť hodnoty rozdielov od priamych hodnôt. Toto je možné riešiť napríklad odoslaním jedného Bytu po každom ôsmom kódovom symbole, v ktorom sa na každom bite nachádza príznak pre jednu z kódovaných symbolov. Hodnoty [11, 14, 244, 200, 200, ...] by sa kódovali ako [11, 3, 244, −44, 0, ...] s bitovými príznakmi `00100...`, kde hodnota bitu `1` značí priamu hodnotu v treťom prijatom kódovom symbole.

3.3 Štatistické metódy kompresie

Najjednoduchšie metódy kompresie sú obecnými metódami, ktoré nemajú žiadne predpoklady o dátach, ktoré nimi budú komprimované. Takéto kódy dátam priraďujú kódové symboly rovnakej dĺžky nezávisle od kontextu alebo vlastností vstupných dát. Všetky vyššie vysvetlené metódy zapadajú do tejto kategórie. Štatistické metódy naopak obvykle priraďujú kódové symboly premenlivej dĺžky, pričom najčastejšie sa vyskytujúcim symbolom sú priradené tie najkratšie.

Štatistické metódy vychádzajú zo Shannonovej teórie informácie, ktorá sa zaoberá kvantifikáciou množstva informácie obsiahnutej v nejakej časti dát. Myšlienka teórie informácie vychádza z postrehu, že obsah informácie v zdelení je ekvivalentné miere prekvapenia týmto zdelením. Vyjadrením množstva informácie v dátach je *entropia* (označovaná H), ktorá má úzky súvis s pravdepodobnosťou výskytu časti dát v danej časti dátového toku [12].

Ak dáta vyjadríme ako sled prvkov z množiny A o n prvkoch $a_1, a_2, a_3, ...a_n$, kde každý prvok sa má šancu vyskytnúť v dátach s pravdepodobnosťou p_i, tak množstvo informácie nesené prvkom a_i je vyjadrené vzťahom 3.15. Základ logaritmu udáva jednotku, ktorou je množstvo informácie reprezentované, teda pre binárny systém sa do základu logaritmu dosadzuje hodnota 2 [38, 12].

$$I(a_i) = -\log_2(p_i) \tag{3.15}$$

Zovšeobecnením vzťahu 3.15 na celú množinu prvkov A sa získava H, teda entropia náhodnej veličiny. Tá je vyjadrením priemerného informačného obsahu dát daná vzťahom 3.16. Hodnota H teda udáva mieru neurčitosti, ktorú má priemerne jeden symbol správy [12].

$$H = \sum_i p_i I(a_i) = -\sum_i p_i \log_2(p_i) \tag{3.16}$$

Najmenšou hodnotou entropie je $H_{\min}(A) = 0$, ktorú nadobúda dátový zdroj o jedinom symbole a_1, ktorý má pravdepodobnosť výskytu $p_1 = 1$. Naopak najvyššiu entropiu s hodnotou $H_{\max}(A) = \log_2 n$ nadobúda dátový zdroj, ktorého n symbolov má rovnomernú pravdepodobnosť výskytu $p_i = \frac{1}{n}$. Redundanciu dát je potom možné vyjadriť vzorcom 3.17 ako rozdiel medzi najvyššou možnou entropiou množiny symbolov a jej skutočnou entropiou [12].

$$R = H_{\max}(A) - H(A) = \log_2 n + \sum_i p_i \log_2(p_i) \tag{3.17}$$

Hodnota entropie slúži ako referenčný bod optimálneho kódovania vstupných dát. Metódy štatistického kódovania majú za cieľ dosiahnuť čo najbližšiu priemernú dĺžku kódu

Tabuľka 3.1: Príklad kódov pre množinu symbolov s danými pravdepodobnosťami. Prevzaté z [38].

Symbol	a	b	c	d	e	f
Pravdepodobnosť	0.3	0.2	0.2	0.1	0.1	0.1
Kód	10	00	01	110	1110	1111

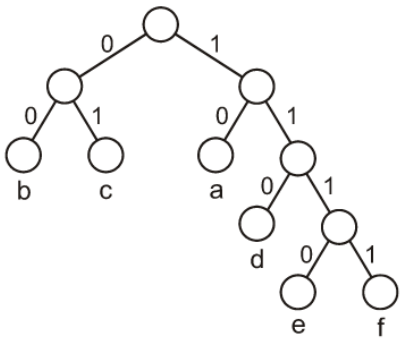
k hodnote entropie pre daný dátový tok. Medzi známe metódy štatistického kódovania patria napríklad Huffmanovo kódovanie, Golomb-Riceovo kódovanie, aritmetické kódovanie, prípadne Shannon-Fanovo kódovanie. Sekcia čerpá z [30].

3.3.1 Huffmanovo kódovanie

Jedná sa o populárnu metódu kódovania, využívanú či už ako samostatnú metódu v rámci aplikácie, alebo ako medzikrok komplexnejších metód. Huffmanovo kódovanie generuje najlepšie kódy ak sú pravdepodobnosti symbolov kódovanej správy zápornými druhými mocninami, teda 2^{-n}, kde $n \in \mathbb{N}$.

Základná verzia tejto metódy začína zoradením jednotlivých symbolov podľa ich pravdepodobnosti výskytu. Na smere zoradenia nezáleží. Následne sa nad zoradeným zoznamom pravdepodobností symbolov skonštruuje binárny strom. Strom sa vytvára od listov ku koreňu iteratívnym pridávaním uzlov, kde listové uzly predstavujú v predošlom kroku zoradené symboly. Strom sa skonštruuje tak, že sa vezmú dva uzly s najnižšou pravdepodobnosťou a vytvorí sa nad nimi spoločný rodičovský uzol ohodnotený súčtom ich pravdepodobností. Žiadny uzol nesmie byť použitý viac, než raz k vytvoreniu rodičovského uzla. Strom je hotový ak nie je možné ďalej vytvárať rodičovské uzly. Koreňový uzol stromu bude mať hodnotu súčtov všetkých pravdepodobností výskytov symbolov, a teda 1.

Ľavá a pravá hrana vychádzajúca z každého uzlu stromu sa následne ohodnotí binárnymi hodnotami 0 a 1. Na hodnote priradenej ľavej a pravej hrane nezáleží, kým je pre každý uzol rovnaká. Binárny prefixový kód pre symbol je reprezentovaný postupnosťou ohodnotení hrán od koreňa k listovému uzlu daného symbolu. Príklad stromu pre množinu symbolov a,b,c,d,e,f je znázornený na obrázku 3.1. Pravdepodobnosti jednotlivých symbolov a ich generované kódy sú uvedené v tabuľke 3.1.

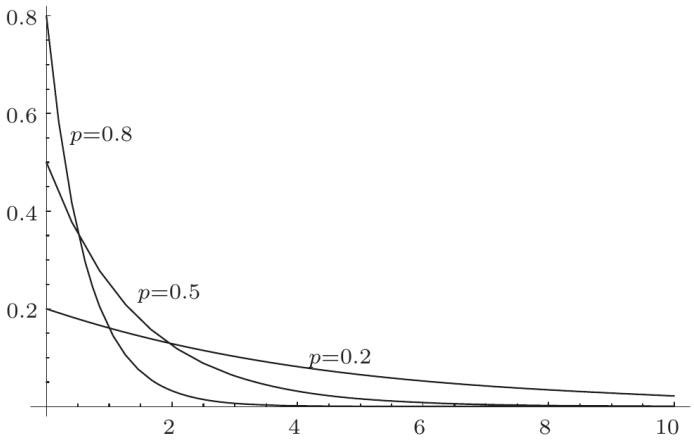


Obr. 3.1: Príklad Huffmanovho stromu. Prevzaté z [38].

3.3.2 Golomb-Riceovo kódovanie

Metóda kódovania bola pôvodne publikovaná v roku 1966. Solomon Golomb v tejto publikácii vysvetľuje svoj inovatívny kód na probléme, ktorému čelí agent `00111` v kasíne. Problém sa zakladá na predpoklade, že agent musí vysielať do svojej základne, či vyhral alebo prehral svoje hry na rulete. Vysielať však môže iba pomocou spojenia, ktoré dokáže odosielať správy iba po bitoch a každý odoslaný bit je nesmierne drahý. Teda je nutné čo najviac obmedziť veľkosť správ.

Problém je rámcovaný nasledovne: každá hra pozostáva zo sekvencie priaznivých udalostí (výhry), ktorá je ukončená jednou nepriaznivou udalosťou (prehrou). Výhra má pravdepodobnosť p, a prehra pravdepodobnosť $1 - p$. V prípade, že by sa jednalo o pravdepodobnosti $p = \frac{1}{2}$, tak podľa teórie informácie najlepšie možné kódovanie výsledku hry je na 1 bit. Avšak ak jedna z pravdepodobností je výrazne vyššia od druhej (pri rulete $p = \frac{1}{37}$), môže byť vhodné využiť kódovanie dĺžok behu (RLE). Pravdepodobnosť dĺžky behu o n prvkoch je $p^n(1-p)$, čo odpovedá geometrickej distribúcii (obrázok 3.2). Dĺžku behu je následne možné kódovať takým kódom variabilnej dĺžky, ktorý je aplikovateľný na nekonečnú abecedu vstupných symbolov (napr. prirodzené čísla $\mathbb{N}$) [14].



Obr. 3.2: Geometrická distribúcia pre rôzne p. Prevzaté z [30].

Golombov kód s parametrom m využíva pre kódovanie hodnoty n kvocient po celočíselnom delení $q = \lfloor \frac{n}{m} \rfloor$, zvyšok po tomto delení $r = n - qm$ a hodnotu $c = \lceil \log_2 m \rceil$. Samotný kód sa skladá z dvoch častí: prvá vyjadruje hodnotu q pomocou unárneho kódovania a druhá kóduje hodnotu r špeciálnym spôsobom. Prvých $2^c - m$ hodnôt je kódovaných ako neznamienková binárna reprezentácia r na $c - 1$ bitoch, pričom zvyšné hodnoty sú kódované na c bitoch tak, že posledná kódovaná hodnota r sú samé jednotkové bity. Kódy s parametrom $m = c^2$ teda využívajú výlučne c bitov pre kódovanie r. V tabuľke 3.2 je možné vidieť tento rozdiel v dĺžke druhej časti kódu pre $m \neq 2^k$ a $m = 2^k$. Kódy, kde $m = 2^k$ sa nazývajú Riceovými kódmi k-teho rádu [30, 12].

Priemerná dĺžka Golombovho kódu je úzko spätá s parametrom m a s vlastnosťami vstupných dát. Metóda kódovania s malým parametrom m totiž generuje veľmi krátke kódové symboly pre malé hodnoty n, avšak ich dĺžka rýchlo rastie pri prekročení hranice $n \leq m$ kvôli unárnemu kódovaniu hodnoty q. Naopak ak je zvolený parameter m väčší, tak sú kódy nízkych hodnôt n spočiatku dlhšie, avšak ich dĺžka rastie pomaly pre vyššie hodnoty n [12]. Tento vzťah medzi parametrom m a kódovanou hodnotou n je znázornený v tabuľke 3.2.

Tabuľka 3.2: Príklady Golombovho kódovania symbolov n s daným parametrom m. Kódy sa skladajú z dvoch častí, ktoré sú oddelené symbolom | pre ľahšiu čitateľnosť. Prevzaté z [30].

m/n	0	1	2	3	4	5	6	7	8	9	10	11	12
3	0|0	0|10	0|11	10|0	10|10	10|11	110|0	110|10	110|11	1110|0	1110|10	1110|11	11110|0
4	0|00	0|01	0|10	0|11	10|00	10|01	10|10	10|11	110|00	110|01	110|10	110|11	11110|00
5	0|00	0|01	0|10	0|110	0|111	10|00	10|01	10|10	10|110	10|111	110|00	110|01	110|10
8	0|000	0|001	0|010	0|011	0|100	0|101	0|110	0|111	10|000	10|001	10|010	10|011	10|100

Optimálny parameter m je možné odvodiť z pravdepodobnosti p na základe nerovnosti 3.18. Táto nerovnosť má pre hodnotu p jediné riešenie m dané vzťahom 3.19. Obecne je možné uviesť, že najlepšia hodnota parametru m je taká, ktorá je najbližšie hodnote $-\frac{1}{\log_2 p}$, resp. hodnota $p^m \approx \frac{1}{2}$. V prípade, že p nie je dopredu známe, tak je možné kódovať na dva behy, kde sa počas prvého zistí hodnota p pre dané dáta a počas druhého sa uskutoční kódovanie. Ďalšími možnosťami sú využitie adaptívneho Golombovho kódovania, kde sa po každom i-tom načítanom symbole prepočíta hodnota m aby sa dosahovala približná hodnota $p^m \approx \frac{1}{2}$, prípadne je možné využiť odhad hodnoty p [30].

$$p^m + p^{m+1} \leq 1 < p^m + p^{m-1} \tag{3.18}$$

$$m = \left\lceil -\frac{\log_2(1+p)}{\log_2 p} \right\rceil \tag{3.19}$$

Kód bol pôvodne navrhnutý pre RLE kódovanie jednej bitovej hodnoty v slede bitov, avšak tento kód je vhodný pre kódovanie ľubovoľnej hodnoty $n \in \mathbb{N}$ v prípade, že rozloženie hodnôt n je geometrické, a teda nízke hodnoty n sú omnoho pravdepodobnejšie, než vysoké.

Kódovať týmto kódom je možné aj záporné hodnoty, avšak je nutné vykonať jednoznačné mapovanie takýchto hodnôt na nezáporné. V prípade, že sú kódované hodnoty zložené výlučne zo záporných čísel, tak je mapovanie triviálne a je postačujúce každú hodnotu vynásobiť -1. Ak sa však vstupná abeceda skladá z ľubovoľného celého čísla, tak je potrebné zaviesť komplexnejšie mapovanie. Vhodným mapovaním môže byť mapovanie uvedené aj v [36], ktoré je popísané vzorcom 3.20, kde n je celé číslo a $M(n)$ je jeho mapovaním na nezáporné celé číslo. Táto operácia je reverzibilná, pretože znamienko pôvodného čísla je kódované pomocou parity mapovaného čísla.

$$M(n) = \begin{cases} 2n & \text{ak } n \geq 0 \\ 2|n| - 1 & \text{ak } n < 0 \end{cases} \tag{3.20}$$

Kapitola 4

Kompresia EKG signálu

Problematike kompresie EKG signálu sa je venované aspoň od sedemdesiatych rokov, prakticky od počiatkov digitalizácie EKG signálu [16]. Behom desaťročí bolo vyvinutých mnoho rôznych prístupov ku kompresii EKG dát a v súčasnosti sa prístupy delia do rôznych kategórii na základe vlastností daných metód. Tieto vlastnosti môžu byť napríklad stratovosť metódy, dimenzionalita metódy, či je metóda využiteľná pre kompresiu v (prakticky) reálnom čase (tzv. *online* metódy), prípadne aj obecný prístup metódy ku kompresii [27, 36, 35]. Častým delením kompresných metód je delenie na tri kategórie:

- priame metódy pracujúce so signálom v časovej doméne,
- metódy založené na transformáciách pracujúce so signálom transformovaným do inej, než časovej domény,
- metódy extrakcie parametrov pracujúce s parametrami EKG signálu (napr. tep srdca, RR intervaly) [36, 35].

Bezstratové metódy kompresie typicky zapadajú do priamych metód, prípadne do metód založených na transformáciách. Online metódy sú podľa prieskumu publikovaných metód [35] obvykle priamymi metódami založenými na lineárnej predikcii spojených s rôznymi formami štatistického kódovania. Metódy založené na transformáciách transformujú vstupný signál z časovej domény do domény inej. Zverejňované metódy využívajú transformácie do frekvenčnej domény (napr. Fourierova transformácia), prípadne aj do časovo-frekvenčnej domény (napr. vlnková transformácia). Transformácie sú však výpočtovo drahé operácie, a teda napriek ich vysokej kompresnej výkonnosti nie sú využívané pre kompresiu v reálnom čase, hoci výskum do podobných technológii už dlhšiu dobu existuje [2, 26].

4.1 Lineárna predikcia

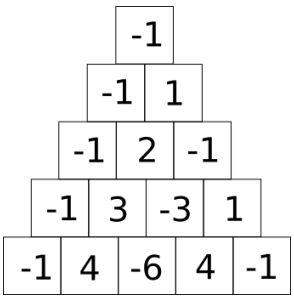
Ako bolo spomenuté vyššie, lineárna predikcia sa v rôznych formách využíva v priamych kompresných metódach. Najbežnejšou formou je dopredná lineárna predikcia odhadujúca vzorky $\hat{x}[n]$ na základe p predchádzajúcich vzoriek signálu x. Formálne je možné tento vzťah vyjadriť pomocou vzorca 4.1

$$\hat{x}[n] = \sum_{i=1}^{p} a_i x[n-i]\,, \tag{4.1}$$

kde $x[n-i]$ sú vzorky predchádzajúce vzorku $x[n]$ a a_i sú koeficienty prediktoru. Jedná sa o jednoduchý prediktor, ktorý je možné koncipovať ako lineárny filter. Takéto filtre sa

hardvérovo jednoducho implementujú, pričom v rôznych univerzálnych procesoroch existujú výpočtové moduly špecializované pre spracovanie a filtrovanie signálov. Jednoduchá je aj implementácia pre VLSI (angl. Very Large Scale Integration) riešenia [8]. Tento predikčný model je často využívaný pri spracovaní diskrétnych signálov, ako je tomu aj v prípade EKG [36, 8, 10].

Takýto prediktor je za vhodných podmienok a pri správne nastavených koeficientoch a_i schopný dokonale odhadovať nasledujúcu hodnotu signálu. Napríklad signály odpovedajúce polynómu p-teho stupňa je možné dokonale predikovať pomocou koeficientov získaných z príslušného riadku Pascalovho trojuholníka so striedajúcimi sa znamienkami koeficientov (obrázok 4.1). Predikcia prvého stupňa je rovná hodnote predošlej hodnote vzorky, teda $\hat{x}[n] = x[n-1]$. Predikcia druhého stupňa je rovná predošlej vzorke posunutej o rovnaký rozdiel, ako je rozdiel oproti jej predošlej vzorke, a teda $\hat{x}[n] = x[n-1] + (x[n-1] - x[n-2])$, čo je ekvivalentné zápisu $\hat{x}[n] = 2x[n-1] - x[n-2]$. Koeficienty prediktorov vyšších stupňov je možné získať analogickým spôsobom [21]. Tieto koeficienty sú využité aj v práci [36], kde sa autori pri použití koeficientov odvolávajú na nízku komplexitu výpočtu a dobrý odhad hodnôt.



Obr. 4.1: Prvých 5 riadkov Pascalovho trojuholníka so striedajúcimi sa znamienkami.

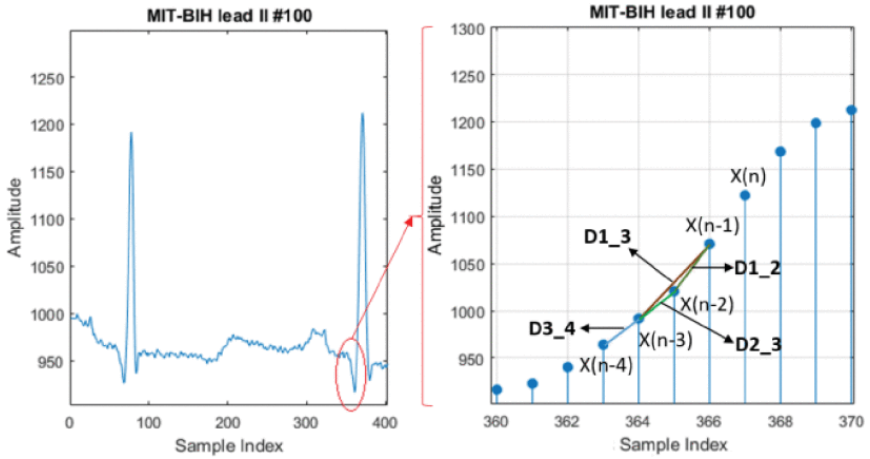
Skutočné signály však obyčajne nekonformujú priebehom polynómov akéhokoľvek stupňa. Z tohto dôvodu sa lineárna predikcia dopĺňa o výpočet chyby predikcie, ktorý je daný vzťahom 4.2. Na základe chyby predikcie je následne možné jednoducho a presne rekonštruovať signál, ak sú pri rekonštrukcii známe dva fakty: **a)** skutočná počiatočná hodnota signálu a **b)** prediktor, pomocou ktorého bola predikcia vykonaná (teda vzťah 4.1). Výhodou dobrého odhadu prediktorom je nízka priemerná hodnota chyby odhadu. Týmto je možné využiť menší počet bitov pre reprezentáciu hodnoty chyby predikcie. Ďalším dopadom tohto kroku je, že vzhľadom na vyššiu pravdepodobnosť, že hodnota chyby padne do relatívne malého intervalu hodnôt, je možné tieto hodnoty potenciálne veľmi efektívne entropicky kódovať, čím sa dosiahne dodatočná kompresia [7].

$$e[n] = x[n] - \hat{x}[n] \tag{4.2}$$

Spôsob, akým je možné zlepšiť presnosť odhadu hodnôt prediktorom, je využitie viacerých prediktorov, pričom voľba konkrétneho prediktoru pre danú vzorku závisí na istých definovaných podmienkach. Takýto prístup je často nazývaný *adaptívnou lineárnou predikciou* (angl. adaptive linear prediction) [36, 10].

4.1.1 Adaptívna lineárna predikcia

Táto metóda vykonáva rozhodnutia pre zámenu prediktoru na základe lokálnych vlastností signálu. Takáto adaptácia prediktoru je vhodná obzvlášť pri signáloch ako sú EKG signály,



Obr. 4.2: Vizualizácia relatívnych rozdielov medzi vzorkami. Prevzaté z [36].

pretože sa v ňom striedajú segmenty s plochým priebehom so segmentami s ráznymi zmenami amplitúdy signálu. Pre každú časť signálu sa teda môže využiť najvhodnejší možný prediktor. Prístupy, ako adaptovať prediktor sú rôzne. Tu sa uvádza metóda popísaná v [36].

Prediktory sú volené adaptívne pre každú vzorku signálu z množiny troch prediktorov. Jedná sa o prediktory prvého až tretieho rádu. Tieto sú vhodné pre predikciu konštantných, lineárnych a kvadratických priebehov signálu. Definície týchto prediktorov sú uvedené vo vzorcoch 4.3, 4.4 a 4.5.

$$\text{Prediktor1} : \hat{x}[n] = x[n-1] \tag{4.3}$$

$$\text{Prediktor2} : \hat{x}[n] = 2x[n-1] - x[n-2] \tag{4.4}$$

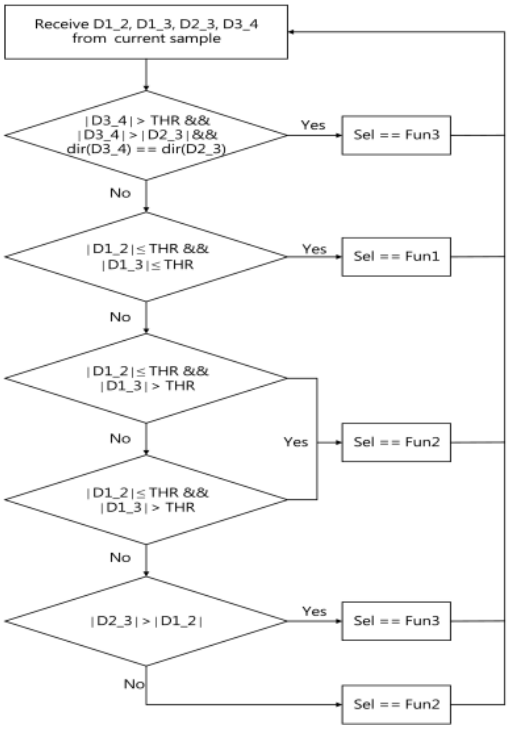
$$\text{Prediktor3} : \hat{x}[n] = 3x[n-1] - 3x[n-2] + x[n-3] \tag{4.5}$$

Rozhodovanie o zvolení správneho prediktoru pre súčasnú vzorku je vykonávané na základe priebehu signálu v rámci predchádzajúcich štyroch vzoriek. Ak sa uvedie, že hodnota $Di\_j[n]$ je výsledkom rozdielu vzoriek $x[n-i] - x[n-j]$, tak je možné definovať hodnoty $D1\_2[n]$, $D1\_3[n]$, $D2\_3[n]$ a $D3\_4[n]$. Vizualizácia vzťahov týchto hodnôt ku vzorkám signálu je na obrázku 4.2. Tieto hodnoty rozdielov slúžia ako pomocné hodnoty pri voľbe vhodného prediktoru pre súčasnú vzorku. Po získaní hodnôt $Di\_j[n]$ sa uplatnia pravidlá znázornené vo vývojovom diagrame na obrázku 4.3, kde funkcia *dir()* je aliasom pre funkciu *sign()*. Parameter THR udáva prahovú hodnotu pre rozlíšenie medzi malým a veľkým rozdielom $Di\_j[n]$. Výsledná zvolená funkcia je jednou z troch preddefinovaných prediktorov. Táto funkcia je následne použitá pre krok bežnej lineárnej predikcie uvedenej vyššie.

Jedným z rozdielov medzi prístupmi statického prediktoru a uvedeného adaptívne voleného prediktoru je, že pri adaptívnom prístupe musia byť chyby predikcie ukladané bezstratovo. V prípade straty informácie nie je uvedený algoritmus schopný správne rekonštruovať signál z dôvodu neschopnosti korektne zvoliť rovnaký prediktov počas rekonštrukcie, než pri kompresii.

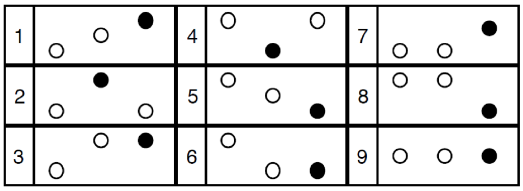
4.2 Turning Point (TP)

Algoritmus bol pôvodne navrhnutý pre redukciu vzorkovacej frekvencie EKG z 200 Hz na 100 Hz, avšak bolo odpozorované, že poskytuje dobrú kompresiu za použitia výpočtovo



Obr. 4.3: Vývojový diagram rozhodovacieho algoritmu pre voľbu prediktora. Prevzaté z [36].

nenáročného spracovania. Algoritmus poskytuje konštantný kompresný pomer CR = 2, pričom je ho možné dvojnásobne zvyšovať opakovaným použitím nad už komprimovaným signálom [3].



Obr. 4.4: Konfigurácie tojice vzoriek x_0, x_1 a x_2 v algoritme Turning Point. Prevzaté z [39].

Algoritmus dosahuje kompresiu selektívnym odstraňovaním vzoriek zo signálu. Pracuje sa v ňom s trojicami vzoriek, kde súčasná vzorka x_0 je braná ako referenčná a vzorky x_1, x_2 sú vzorky nasledujúce po referenčnej. Z týchto dvoch nasledujúcich vzoriek je vždy ponechaná iba jedna, pričom druhá je zo signálu odstránená. Vzorka, ktorá je ponechaná je tá, ktorá zachováva bod zlomu (turning point) v rámci vyhodnocovanej trojice. Vzťah, na základe ktorého sa vykonáva rozhodnutie pre ponechanie vzorky je uvedený vo vzorci 4.6, kde $\hat{x}$ je výsledná vzorka zvolená pre ponechanie v signáli. Všetkých deväť konfigurácií tejto trojice vzoriek je možné vidieť na obrázku 4.4.

$$\hat{x} = \begin{cases} x_1 & \text{ak } (x_1 - x_0) * (x_2 - x_1) < 0 \\ x_2 & \text{ak } (x_1 - x_0) * (x_2 - x_1) \geq 0 \end{cases} \tag{4.6}$$

Rekonštrukcia takto komprimovaného signálu je vykonávaná pomocou interpolácie medzi vzorkami komprimovaného signálu. Chyba rekonštruovaného signálu je obecne nízka [3],

avšak vzhľadom na odstraňovanie vzoriek zo signálu v nepravideľných intervaloch nastáva časové skreslenie v signáli [39].

4.3 Amplitude Zone Time Epoch Coding (AZTEC)

Algoritmus AZTEC (Amplitude Zone Time Epoch Coding) bol pôvodne navrhnutý ako krok predspracovania EKG signálu pre následné digitálne spracovanie, pričom v jeho základnej verzií dosahoval kompresného pomeru v priemere $CR = 10$ [9]. Jedná sa o stratovú metódu online kompresie. Metóda spočíva v transformácii vstupného signálu s vysokofrekvenčnými zmenami amplitúdy na signál tvorený segmentami s vodorovnými úsečkami (plató) a segmentami tvorenými úsečkami so sklonom. Segmenty so sklonom sú najčastejšie tvorené QRS komplexami. Metóda je zložená z dvoch krokov, a to z detekcie úsečky a spracovania úsečky [9, 39].

Detekcia úsečky plní účel rozpoznania, či je nutné vytvoriť novú úsečku v komprimovanom signáli. Tento krok je inicializovaný prvou hodnotou signálu, ako $v_{\max} = v_{\min} = v_0$. Premenné $v_{\max}$ a $v_{\min}$ udržiavajú maximálnu a minimálnu hodnotu prijímaných vzoriek vstupného signálu a v_0 je prvá vzorka vstupného signálu po vytvorení predchádzajúcej úsečky. Vstupné vzorky sú následne kontrolované, či zapadajú do rozsahu $v_{\min} \leq v_i \leq v_{\max}$. V prípade, že do tohto rozsahu vzorka nezapadá, je rozsah rozšírený. Toto rozširovanie rozsahu je vykonávané až kým sa nepresiahne predom špecifikovaný prahový limit $v_{\max} - v_{\min} < K$ alebo kým $i < L_{\text{maxlen}}$. Hodnota L_{maxlen} udáva maximálnu dĺžku vodorovnej úsečky v počte vzoriek [9].

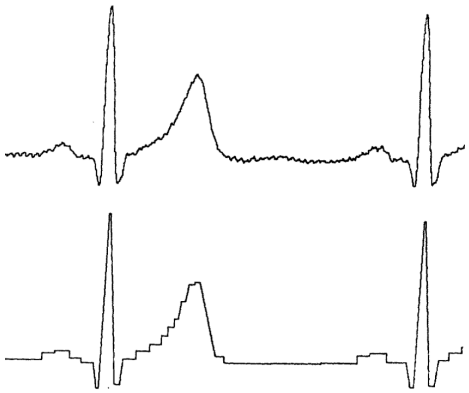
Spracovanie úsečky má za účel vygenerovať vhodnú úsečku na základe kroku detekcie. V tomto kroku môžu nastať dve situácie:

1. prahový limit K bol prekročený a $4 < i <= L_{\text{maxlen}}$,
2. prahový limit K bol prekročený a $4 \geq i$.

Ak nastane prvá situácia, tak sa do komprimovaného signálu vloží vodorovná úsečka o dĺžke i s hodnotou amplitúdy $(v_{\max} + v_{\min})/2$. V opačnom prípade bola detegovaná priveľká variácia v signáli za prikrátky čas a je nutné vytvoriť úsečku so sklonom. Toto je vykonávané iteratívne, a to spôsobom, že sa najskôr zaznamená rozdiel medzi hodnotami amplitúdy predošlej vodorovnej línie a súčasnej vzorky. Následne je opäť spustený krok detekcie úsečky. Ak opäť nastane vyššie uvedená druhá situácia pri prekročení K, tak sa celý proces opakuje. Inak je uložená klonená úsečka v komprimovanom signáli, ktorá je reprezentovaná smernicou úsečky a jej dĺžkou [9, 39]. Na obrázku 4.5 je možné vidieť porovnanie vizualizácie EKG signálu pred kompresiou a po kompresii pomocou metódy AZTEC.

Výsledný komprimovaný signál je tvorený dvojicami 12-bitových slov, kde prvé slovo naraz kóduje informáciu o type úsečky a jej dĺžke – kladná hodnota reprezentuje vodorovnú úsečku a záporná klonenú. Druhé slovo obsahuje amplitúdu, prípadne smernicu uloženej úsečky, kde hodnota je interpretovaná v závislosti od znamienka prvého slova dvojice [9].

Existuje niekoľko vylepšených verzií tejto kompresnej metódy, ako je napríklad MAZTEC (Modified AZTEC) alebo CORTES (Coordinate Reduction Time Encoding System). Algoritmus MAZTEC využíva štatistické informácie o vstupnom signáli pre adaptívne prispôsobovanie parametrov algoritmu. Algoritmus CORTES je hybridom metód AZTEC a TP, kde nízkofrekvenčné časti signálu sú spracované metódou AZTEC a vysokofrekvenčné metódou TP [28].

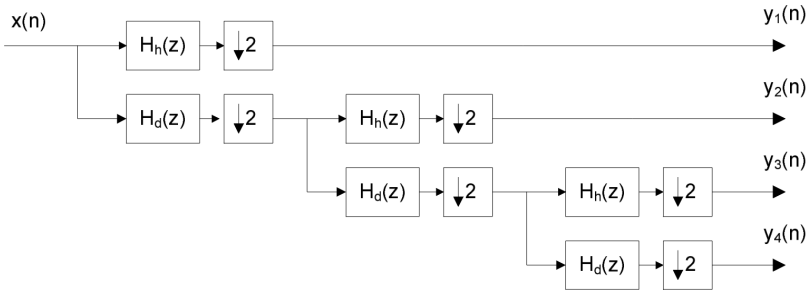


Obr. 4.5: Ukážka EKG signálu pred (hore) a po kompresii (dole) pomocou metódy AZTEC. Prevzaté z [9].

4.4 Set Partitioning in Hierarchical Trees (SPIHT)

Táto metóda je vylepšením kompresnej metódy Embedded Zerotree Wavelet Coding (EZW) a je častým bodom záujmu vo výskume stratových offline kompresných algoritmov pre EKG signály [27]. Algoritmus využíva 1D (jednorozmernú) vlnkovú transformáciu, po ktorej sú jej koeficienty do výstupného dátového toku kódované pomocou prístupu SPIHT. V tejto sekcii je uvedené obecné fungovanie algoritmu, pričom sekcia čerpá z [18] ak nie je uvedené inak.

Prvým krokom algoritmu je diskrétna vlnková transformácia vstupného signálu. Týmto krokom sa získajú koeficienty dekompozície signálu $y_c[n]$. Na obrázku 4.6 je zobrazená troj-úrovňová dekompozícia vstupného signálu $x[n]$. Dekompozičné filtre horného a dolného priepustu sú označené ako $H_h(z)$ a $H_d(z)$, v rovnakom poradí. Po každom filtračnom bloku nasleduje blok dvojnásobného podvzorkovania.

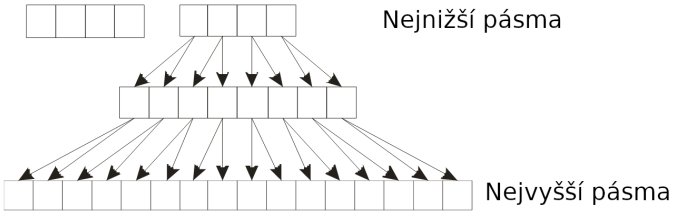


Obr. 4.6: Diagram troj-úrovňovej diskrétnej vlnkovej transformácie jednorozmerného signálu x na koeficienty y_1 až y_4. Prevzaté z [18].

Filtračný blok horného priepustu $H_h(z)$ vytvára signál koeficientov vysokých detailov signálu, pričom blok dolného priepustu $H_d(z)$ vytvára tzv. aproximačný signál tvorený koeficientmi nízkych detailov. Pri viacúrovňovej dekompozícii signálu, ako je aj schéma dekompozície na obrázku 4.6, sa nižšie úrovne dekompozície získavajú z aproximačného signálu.

Vďaka tejto stromovej štruktúre dekompozície sú výsledkom koeficienty, ktoré sú si navzájom príbuzné naprieč úrovňami dekompozície. Tieto príbuzné koeficienty tvoria strom pre každú skupinu vstupných vzoriek signálu. Jedná sa o skupinu vzoriek z dôvodu kroku

podvzorkovania počas vlnkovej transformácie. Pre ilustráciu je uvedený obrázok 4.7. Najnižšie pásmo odpovedá najväčšiemu stupňu rozkladu. Týchto koeficientov je najmenšie množstvo, pretože prešli viacerými krokmi podvzorkovania. Teda každý koeficient z y_3 a y_4 je možné chápať ako koreňový uzol stromu, ktorý má dvoch potomkov z y_2, ktorý má následne dvoch potomkov z y_1. Koeficienty sú delené do dvoch množín, a to $\mathcal{D}(i)$ a $\mathcal{L}(i)$. Množina $\mathcal{D}(i)$



Obr. 4.7: Ilustrácia stromovej štruktúry koeficientov viacúrovňovej vlnkovej transformácie. Prevzaté z [18].

obsahuje všetkých potomkov koeficientu $y_i[n]$ a množina $\mathcal{L}(i)$ obsahuje všetkých *nepriamych* potomkov. Koeficienty v $\mathcal{D}(i)$ sú označované ako typ A a $\mathcal{L}(i)$ ako typ B [29].

Algoritmus SPIHT, podobne ako EZW, pracuje s myšlienkou dôležitých a nedôležitých koeficientov, avšak na rozdiel od EZW, pracuje s troma zoznamami pre vytváranie výstupného kódu. **LIP** (List of Insignificant Pixels) je zoznam nedôležitých koeficientov, **LIS** (List of Insignificant Sets) je zoznam nedôležitých množín a **LSP** (List of Significant Pixels) je zoznam dôležitých koeficientov[1].

Celý algoritmus sa skladá zo štyroch hlavných krokov, pričom tri z nich sa opakujú kým sa nedosiahne požadovanej kompresie. Nižšie je uvedený zjednodušený popis algoritmu, pričom pre úplný algoritmus je čitateľ odkázaný na pôvodnú publikáciu [29], prípadne na jednu z publikácií popisujúce variantu pre jednorozmerný signál [18].

Prvým krokom je inicializácia parametru prahu T pre rozhodovanie medzi dôležitým a nedôležitým koeficientom a inicializácia zoznamov LIP, LSP a LIS. Zoznam LSP je inicializovaný ako prázdny, do LIP sú uložené všetky koreňové koeficienty a do LIS sú uložené všetky koeficienty typu A. Počiatočný prah T je nastavený podľa hodnoty n_T, ktorá je vypočítaná vzťahom 4.7, kde $y_{\max}$ označuje najvyššiu absolútnu hodnotu zo všetkých koeficientov $y_i[n]$. Prah T je následne vypočítaný vzorcom 4.8.

$$n_T = \lfloor \log_2(y_{\max}) \rfloor \tag{4.7}$$

$$T = 2^{n_T} \tag{4.8}$$

Druhým krokom je *zoraďovacia fáza* (sorting pass). V tejto fáze sa najskôr skontroluje, či sú koeficienty v LIP dôležité. Dôležitosť je testovaná ako $y_i[n] \geq T$, kde pravdivé vyhodnotenie znamená, že koeficient je dôležitý. Ak koeficient nie je dôležitý, na výstup je odoslaný bit 0. Ak je dôležitý, na výstup je odoslaný bit 1 a ďalší bit značiaci znamienko koeficientu. Ďalej je otestovaná dôležitosť všetkých stromov v LIS. Strom je označený ako dôležitý ak je niektorý potomok dôležitý, pričom dôležitosť je kontrolovaná v závislosti od jeho typu – A alebo B. V závislosti od typu sú pri zistení dôležitosti generované bity na výstupe a koeficienty sú presúvané medzi troma zoznamami algoritmu.

[1] Názvy týchto zoznamov sú prebrané z pôvodného návrhu SPIHT algoritmu pre kompresiu obrazových dát.

Tretím krokom je *zdokonaľovacia fáza* (refinement pass). V tomto kroku sa pre každý koeficient v LSP, s výnimkou tých pridaných v poslednej zoraďovacej fáze, pošle na výstup n_T-ty najvýznamnejší bit hodnoty koeficientu. Posledným krokom je dekrementácia hodnoty n_T, prepočítanie hodnoty T a opakovaním algoritmu od druhého kroku.

Táto metóda kompresie je progresívna, a teda je výstupný kód tvorený stále sa zvyšujúcou presnosťou reprezentácie koeficientov. Vďaka tejto vlastnosti je možné pred spustením algoritmu nastaviť požadovanú presnosť reprezentácie koeficientov, čím sa efektívne ladí stratovosť a kompresná výkonnosť algoritmu.

Kapitola 5

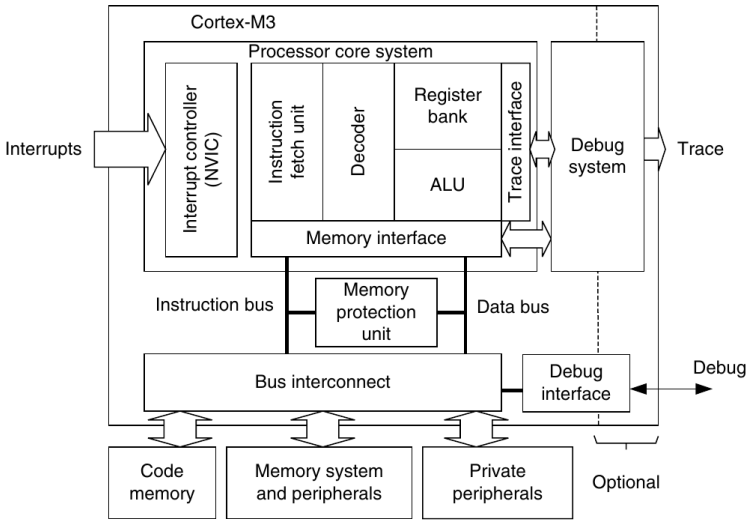
Procesory ARM Cortex-M

Mikrokontroléry sa významne rozšírili a našli uplatnenie v obrovskom množstve rôznych priemyslov, ale aj u hobbistov. Je možné sa s nimi stretnúť kdekoľvek od aeronautiky, cez výrobné haly, v domácich projektoch a v neposlednom rade v medicíne. Elektronické zariadenia sú využívané v čím ďalej, tým viac ohľadoch v rámci medicíny, pričom nahrádzajú ich analógové protiklady. Tento trend je ľahko ilustrovateľný aj v domácich medicínskych pomôckach, kde sú nahrádzané teplomery, merače srdcového tlaku, hoci aj terapeutické masážne zariadenia. Samozrejme stále existujú isté výhody analógových prístupov (napr. jednoduchá dezinfekcia sklenených teplomerov), ale dopad rozšírenia mikrokontrolérov na medicínu je nespochybniteľný, obzvlášť v oblastiach, kde analógové riešenia nie sú prevediteľné.

Rôzne mikrokontroléry majú rôzne vlastnosti udávané prítomnými komponentami. Môže sa jednať o telemetrické zariadenia poskytujúce informácie o svojom prostredí alebo o zariadenia poskytujúce nejakú službu – v závislosti od určenia a schopností zariadenia. Častým výrazom spájaným s mikrokontrolérmi je tzv. *IoT*, teda Internet of Things, ktorý označuje zariadenia schopné pripojenia sa na nejakú komunikačnú sieť, v ktorej sú tieto zariadenia adresovateľné. Tieto zariadenia disponujú komunikačnou jednotkou schopnou komunikovať či už cez internet, cez Bluetooth, alebo cez iné médium. Všetky programovateľné mikrokontroléry však obsahujú aspoň jednu riadiacu jednotku, procesor, vykonávajúci inštrukcie programu, ktorý definuje správanie zariadenia. Procesory s ARM architektúrou sú rozšírenou rodinou procesorov naprieč mikrokontrolérmi a SoC (angl. System on Chip) rôznych využití [41].

Firma Arm[1] v súčasnosti navrhuje a licenciuje niekoľko rodín jadier v rámci série Cortex. Rodina **Cortex-A** poskytuje vysoký výkon a typicky je využívaný v zariadeniach s operačnými systémami, ako sú Linux a Android. Zariadenia s **Cortex-R** sú zasadzované do systémov vyžadujúcich vysoký výkon a priepustnosť v reálnom čase, ako sú napríklad pevné disky alebo sieťové zariadenia. Rodina **Cortex-M** je asi najrozšírenejšia a využívaná v širokej škále mikrokontrolérov. Tieto jadrá poskytujú dobrý výkon s rôznymi možnosťami dodatočných modulov v závislosti od použitého modelu [41, 5]. Táto kapitola sa zaoberá jadrami Cortex-M3 a Cortex-M4F a čerpá z [41] ak nie je uvedené inak.

[1] Pre jednoznačnosť uvádzam, že v texte je názov firmy vysádzaný ako „Arm“ a názov procesorov ako „ARM“.



Obr. 5.1: Zjednodušený blokový diagram procesoru Cortex-M3. Prevzaté z [41].

5.1 Architektúra

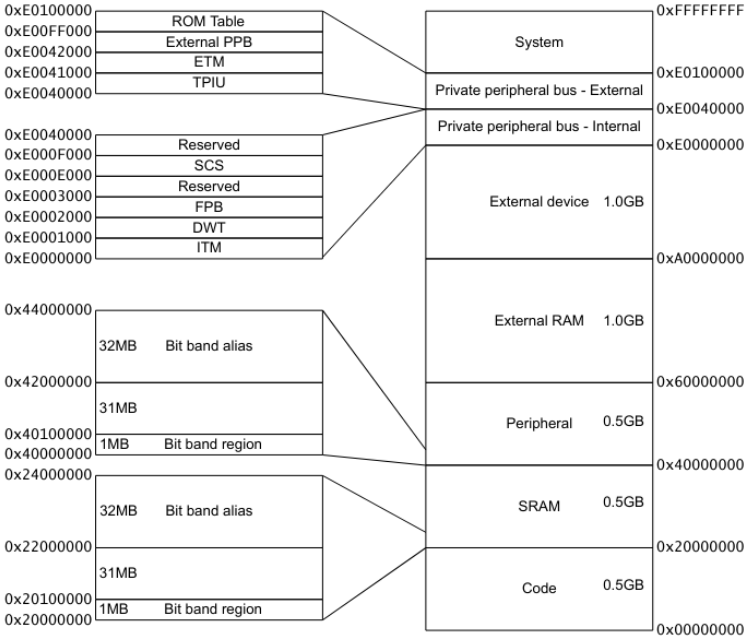
Procesory Cortex-M3 a Cortex-M4F sú obe 32 bitové, čo znamená 32 bitovú adresovateľnosť, registre a dátové zbernice. V kontexte ARM je teda dĺžka slova (angl. word) definovaná na 32 bitov. Štruktúra týchto dvoch procesorov je identická a blokový diagram pre Cortex-M3 je zobrazený na obrázku 5.1. Jadro procesora obsahuje typické jednotky ako IFU (Instruction Fetch Unit), dekodér inštrukcií, registre a ALU (Arithmetic Logic Unit). Nechýba ani radič prerušení NVIC (Nested Vectored Interrupt Controller) podporujúci externé prerušenia alebo aj dynamické nastavenie priorít prerušení [5]. Trace jednotka umožňuje jednoduché ladenie programu v prípade, že procesor disponuje voliteľným ladiacim modulom.

5.1.1 Pamäť

Procesory ARM sú tzv. *bi-endian*, a teda podporujú režimy little-endian a big-endian v závislosti od nastavenia programátorom. Endianita označuje poradie Bytov viac-Bytového slova v pamäti. Pri big-endian systéme sa ukladá najvýznamnejší Byte slova na najnižšej adrese a najmenej významný Byte na najvyššej adrese slova. Little-endian je označenie pre opačné usporiadanie Bytov slova v pamäti [5].

Vzhľadom na 32 bitovú adresovateľnosť týchto procesorov je najvyššia možná adresa `0xFFFFFFFF`, čo činí 4 GB adresovateľnej pamäte. Pamäť je rozdelená na niekoľko regiónov, ktorých mapovanie je zobrazené na obrázku 5.2. Procesory architektúry ARMv7 sú založené na modifikovanej Harvardskej architektúre, a teda disponujú oddelenými zbernicami pre inštrukcie a dáta (na obrázku 5.1 označené ako Instruction a Data bus). Modifikácia Harvardskej architektúry navyše umožňuje využitie aj dátového priestoru pre ukladanie inštrukcií. V architektúre ARMv7 sa však obmedzujú regióny, z ktorých je možné načítať inštrukcie na regióny Code, SRAM a RAM.

Procesory disponujú troma zbernicovými rozhraniami AHB-Lite (z angl. Advanced High performance Bus) a jedným APB-Lite (z angl. Advanced Periferal Bus). Zbernice AHB sú delené na ICode, DCode a System zbernice. V závislosti od regiónu pamäte a typu načítavaných slov sú využívané príslušné zbernicové rozhrania (napr. pre región Code sa využívajú



Obr. 5.2: Adresné mapovanie pamäte pre Cortex-M3. Prevzaté z [4].

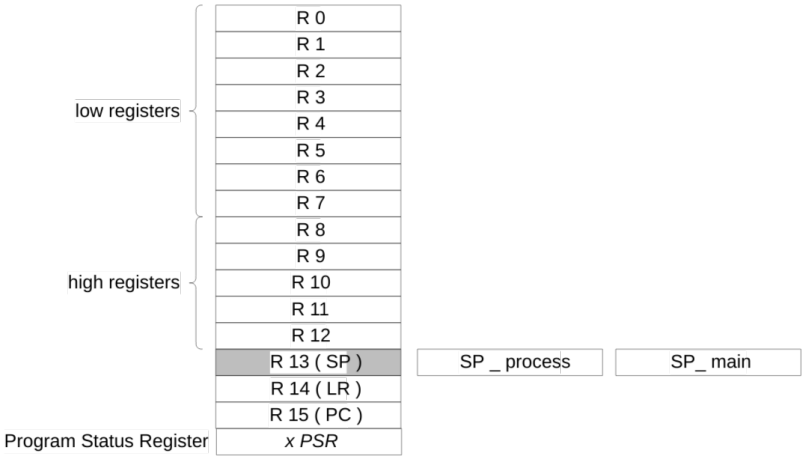
zbernice ICode pre inštrukcie a DCode pre dáta). Periférna zbernica APB zodpovedá za komunikáciu medzi jadrom procesora a perifémymi modulmi ako sú UART, časovače apod.

5.1.2 Registre

Procesory založené na ARMv7 architektúre disponujú šestnástimi tzv. *ARM core* registrami R0 až R15 a niekoľkými dodatočnými špeciálnymi registrami. Registre R0-R12 sú všeobecnými registrami a rozdeľujú sa na dve skupiny. Registre R0-R7 sú tzv. spodné registre a R8-R12 sú tzv. vrchné registre (viď obrázok 5.3). Registre R13-R15 majú preddefinované určenie a v poradí číslovania sa nazývajú: *SP* (Stack Pointer), *LR* (Link Register) a *PC* (Program Counter). Register SP je *bankovaný*, čo znamená, že každý *operačný režim* procesoru (viď sekciu 5.1.3) disponuje vlastným SP registrom [5, 41]. Tento register je využívaný pre ukladanie a obnovenie hodnôt zo zásobníku. Register LR je využívaný pri vetvení programu a typicky je v ňom ukladaná návratová adresa pre návrat z volanej podrutiny. Register PC obsahuje adresu v pamäti súčasne vykonávanej inštrukcie.

Špeciálne registre sa delia do troch skupín. **Program status register PSR** s podregistrami APSR, IPSR a EPSR. Tiež označovaný ako xPSR, tento 32-bitový register zastrešuje tri podregistre obsahujúce informácie o behu programu. APSR obsahuje príznaky využívané pre vyhodnotenie podmienených inštrukcií. V prípade, že je v konkrétnom SoC implementované rozšírenie DSP (Digital Signal Processor) je v APSR aj bitové pole využívané SIMD (Single Instruction Multiple Data) inštrukciami. IPSR obsahuje číslo výnimky, ktorá sa aktuálne vykonáva. Tento register je používaný iba v Handler operačnom režime, inak je hodnota tohto poľa 0. EPSR obsahuje bit T, ktorý pre procesory ARMv7-M musí byť nastavený na hodnotu 1. Nakoniec sa v tomto registri nachádza pole pre ukladanie stavu procesora pre exception-continuable inštrukcie (ICI), prípadne If-Then (IT) inštrukcie. **Maskovacie registre** PRIMASK, FAULTMASK, BASEPRI. Registre PRIMASK a FAULTMASK sú jednobitové a nastavujú prioritu výnimiek na 0 a −1. BASEPRI je 8-bitový a nastavuje

potrebnú úroveň priority výnimiek pre ich preempciu. **Ovládací CONTROL register** je dvoj- alebo troj-bitový register nastavujúci prístupové privilégiá v Thread režime prvým bitom a využívaný SP register v Thread režime druhým bitom. Ak má SoC implementované FP (Floating Point) rozšírenie, tak tretí bit nastavuje, či je v súčasnom kontexte FP aktívny [5].

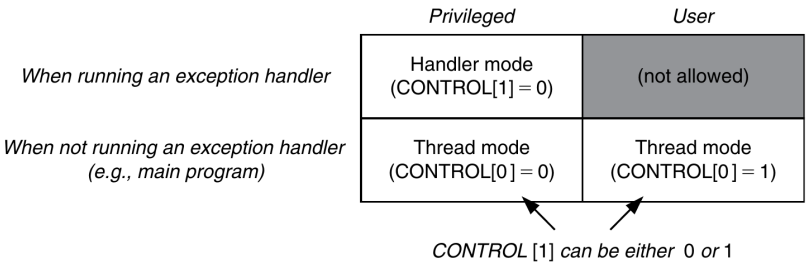


Obr. 5.3: Registre architektúry ARMv7. Prevzaté z [4].

5.1.3 Operačné režimy

Popisovaná architektúra podporuje dva operačné režimy a dva úrovne privilégií; režimy *Thread* a *Handler* a úrovne *privilegovaná* a *užívateľská*. Tieto režimy poskytujú úroveň zabezpečenia a robustnosti architektúry. Dosahuje sa to tým, že rôzne kombinácie operačných režimov a privilégií obmedzujú prístup k častiam pamäte, prípadne k vykonávaným rutinám.

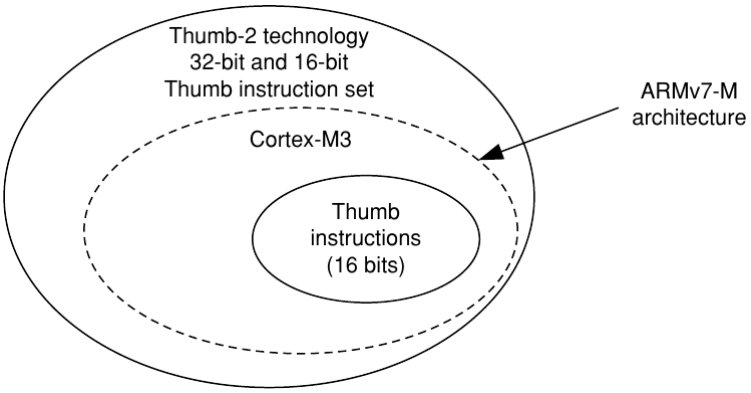
Operačný režim Handler je využívaný výlučne na obsluhu výnimiek. Tento operačný režim má vždy privilegovanú úroveň. Bežné vykonávanie programu sa deje v Thread režime, ktorý môže byť privilegovaný alebo užívateľský. Do privilegovaného Thread režimu je možné sa dostať jedine výlučne prepnutím z privilegovaného Handler režimu a nijak inak. Prehľad možných kombinácií operačných režimov a úrovní privilégií je možné vidieť na obrázku 5.4. Typicky sa v jednoduchých aplikáciach užívateľská úroveň nevyužíva a celý beh programu je vykonávaný v privilegovanom Thread a Handler režime.



Obr. 5.4: Kombinácie operačných režimov a úrovní privilégií počas obsluhy výnimky a počas bežnej operácie. Prevzaté z [41].

5.2 Inštrukčná sada Thumb

Jadrá rodiny Cortex-M sú navrhnuté nad architektúrou ARMv7 profilu M nazývanom ARMv7-M. Tento profil podporuje výlučne inštrukčnú sadu Thumb s technológiou Thumb-2. Pôvodná, 16 bitová, verzia tejto inštrukčnej sady bola vydaná ako komplementárna k 32 bitovej inštrukčnej sade ARM a jej cieľom bolo dosiahnuť čo najvyššiu *hustotu kódu*, čím by sa znižovali pamäťové nároky na uloženie kódu v pamäti mikrokontroléru. Sada Thumb sa z tohto dôvodu skladala z výlučne 16 bitových inštrukcií, ktoré boli tvorené podmnožinou inštrukcií inštrukčnej sady ARM. Následne sa v roku 2003 vydala technológia Thumb-2, ktorá rozširovala 16 bitovú inštrukčnú sadu Thumb o dodatočné 32 bitové inštrukcie [41]. Profil ARMv7-M podporuje podmnožinu týchto dodatočných 32 bitových inštrukcií. Podporované inštrukcie závisia na architekturálnych rozšíreniach. V prípade profilu ARMv7-M sa jedná o rozšírenia DSP (Digital Signal Processor) a o rozšírenie pre podporu plávajúcej rádovej čiarky[2] (angl. floating point extension). Jadro Cortex-M4 disponuje DSP modulom a Cortex-M4F je navyše doplnené aj o podporu plávajúcej rádovej čiarky, a teda podporuje príslušné 32 bitové inštrukcie [5]. Znázornenie podmnožín Thumb inštrukcií je na obrázku 5.5.



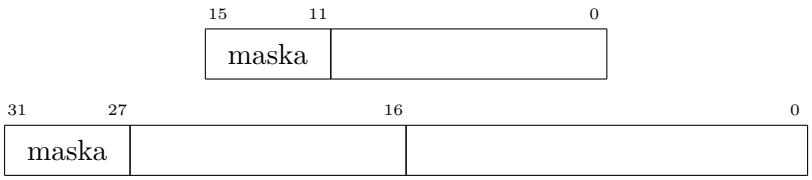
Obr. 5.5: Znázornenie podmnožín inštrukcií inštrukčnej sady Thumb. Prevzaté z [41].

Ako bolo spomenuté, Thumb inštrukcie môžu byť dĺžky 16 alebo 32 bitov a slovo je definované na dĺžku 32 bitov. Samotný inštrukčný tok je sekvenciou polslov (angl. halfword) zarovnaných na polslová. Inštrukcie sa vždy dekódujú po polslovách. Ak aktuálne dekódované polslovo obsahuje špecifickú sekvenciu bitov na pozíciách [15:11], tak sa načíta aj nasledujúce polslovo, ktoré je využité pre úplné dekódovanie dlhšej, 32 bitovej, inštrukcie [5]. Tieto bity môžu mať jednu z troch podôb: `0b11101`, `0b11110`, alebo `0b11111`. Na obrázku 5.6 sú zobrazené bitové polia dvoch typov inštrukcií. Sekcie označené „maska“ predstavujú polohy bitov rozhodujúcich o veľkosti načítavanej inštrukcie. Tento prístup miešania krátkych a dlhých inštrukcií umožňuje vykonávanie programu využívajúceho 32 bitové inštrukcie bez drahého prepínania režimu procesora z jednej inštrukčnej sady do druhej a zároveň poskytuje dobrú hustotu kódu.

5.2.1 Unified Assembly Language

Sekcia čerpá z [41] a uvádza základné koncepty jazyku Unified Assembly Language. Pre ARM systémy bol po vydaní technológie Thumb-2 zavedený jazyk Unified Assembly Lan-

[2] Často označované aj ako pohyblivá rádová čiarka.



Obr. 5.6: Dve možnosti veľkosti inštrukcií Thumb. Hore znázornená 16 bitová a dole 32 bitová inštrukcia.

guage (tzv. UAL). Zjednocuje sa ním syntax pre inštrukčné sady ARM a Thumb, pričom sa podporujú 16 aj 32 bitové inštrukcie. Zjednotením syntaxe sa zjednodušuje prenositeľnosť kódu medzi rôznymi ARM modelmi. Základná syntax jazyka UAL je nasledovná:

```
label
  opcode operand1, operand2, ..., operandN ; Comment
```

Syntax je veľmi podobná iným Assembly jazykom, kde `label` priraďuje zvolené meno adrese nasledujúcej inštrukcie, `opcode` udáva názov inštrukcie a počet operandov je závislý na danej inštrukcii. Komentáre sa začínajú bodkočiarkou a končia koncom riadku.

Práca so zásobníkom

Zásobník je možné využiť pre dočasné uloženie a obnovenie hodnôt registrov pomocou inštrukcií `PUSH` a `POP`. Tieto inštrukcie podporujú iba prácu so všeobecnými registrami (R0 až R12). Inštrukcia `PUSH` podporuje jeden dodatočný register, a to LR, pričom inštrukcia `POP` podporuje dodatočné registre LR a PC [5].

Prípony inštrukcií

Takmer všetky inštrukcie spracúvajúce dáta (`ADD`, `SUB`, `MOV` apod.) podporujú možnosť aktualizácie podmienkových príznakov v APSR. Pre túto aktualizáciu sa využíva prípona `S`. Príznaky v APSR je možné využiť pre podmienené vykonanie väčšiny inštrukcií. Toto je dosiahnuté podmienenými verziami inštrukcií, ktoré využívajú podmienkové prípony. Väčšina inštrukcií teda podporuje dva voliteľné typy prípon upravujúce ich vykonávanie. Ako ilustračný príklad sa uvedie úplný tvar inštrukcie pre načítanie hodnoty do registra `MOV{cond}{S} Rd, Operand2`, kde `cond` môže byť jednou z mnohých prípon pre podmienené vykonanie. Pre úplný prehľad týchto prípon podporovaný architektúrou ARMv7 je čitateľ odkázaný na [5]. Podmienené vykonávanie inštrukcií znižuje výpočtovú cenu podmienok pri procesoroch bez prediktorov vetvenia.

Flexibilný druhý operand

Veľké množstvo inštrukcií podporuje aj tzv. flexibilný druhý operand (flexible second operand). Je nazývaný flexibilný, pretože môže mať dva tvary: `#immed_8r` alebo `Rm{, shift}`. Priama číselná konštanta `#immed_8r` musí korešpondovať 8-bitovému vzoru rotovanému vľavo o párny počet bitov v rámci 32 bitového slova (napr. hodnota `#1020` je hodnota `0xFF` rotovaná vpravo o 30 bitov). Tvar `Rm{, shift}` umožňuje použiť jeden z ARM registrov pre dodanie hodnoty druhého operandu. Voliteľná časť `shift` umožňuje posunúť alebo rotovať hodnotu v špecifikovanom registri pred jej použitím. Teda napríklad inštrukcia tvaru

`SUB r11,r12,r3,ASR #5` je čítaná ako `r11 = r12 - (r3 » 5)`, pričom sa jedná o aritmetický bitový posun vpravo (symbol »).

Vetvenie a volanie podrutín

Podporovaných je niekoľko rôznych inštrukcií vetvenia, pričom najzákladnejšími sú inštrukcie `B label` a `BL label`. Inštrukcia `B` vykoná vetvenie (skok) na danú adresu, pričom adresa je špecifikovaná pomocou `label`. Inštrukcia `BL` sa využíva pre volanie podrutín, pretože ukladá návratovú adresu do registru LR. Pre návrat z podrutín je možné využit inštrukciu `BX`, ktorá má jeden parameter, a to register obsahujúci adresu, na ktorú sa má skočiť. Čiže pre návrat z podrutiny sa použije inštrukcia `BX LR`.

Inštrukcia `BL` zakaždým prepisuje register LR. Pri zanorovanom volaní podrutín je preto nutné hodnotu uloženú v LR nejakým spôsobom zachovať, aby sa bolo možné zo zanorených volaní vrátiť. Typicky sa na uloženie a obnovenie LR využíva zásobník. Podrutina volajúca ďalšiu podrutinu môže vyzerať nasledovne:

```
podrutinaA
  PUSH LR
  ...
  BL podrutinaB
  ...
  POP PC
podrutinaB
  ...
  BX LR
```

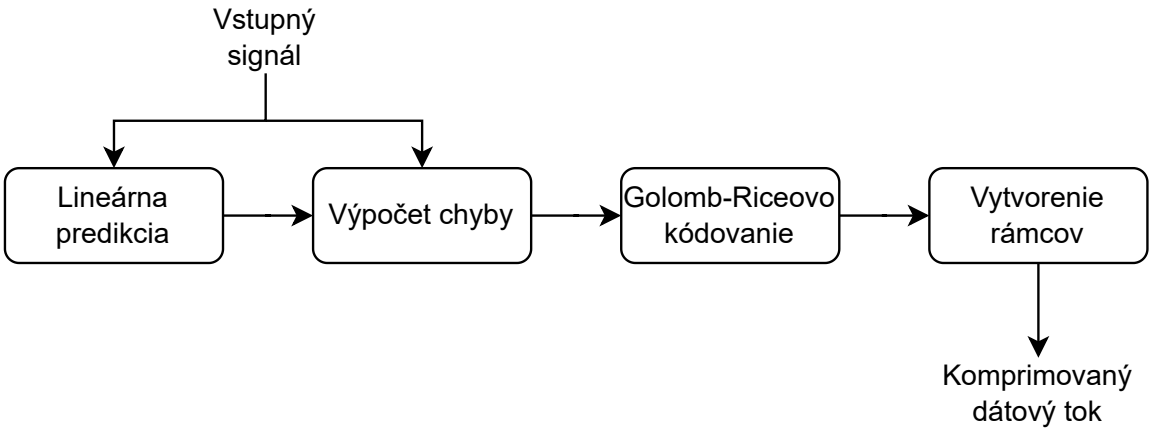
Podrutina B nevolá ďalšie podrutiny, preto nie je nutné ukladať hodnotu v LR a návrat z podrutiny je možný pomocou inštrukcie `BX`. Jednoduchý skok pomocou uvedených vetviacich inštrukcií je vo svojej podstate iba prepísanie obsahu registra PC. Preto je návrat z podrutiny A možný jednoduchým obnovením hodnoty návratovej adresy zo zásobníka do registra PC.

Kapitola 6

Návrh algoritmu

Navrhovaný algoritmus je založený na lineárnej predikcii signálu a následnom entropickom kódovaní chybového signálu predikcie. Jedná sa o úpravu často využívanej metódy v oblasti výskumu bezstratových kompresných algoritmov vhodných pre využitie v MCU (mikrokontroléroch), prípadne vo VLSI (angl. Very Large Scale Integration) systémoch. Pre hlbší prehľad takýchto algoritmov viď [35].

Navrhovaný kompresný algoritmus sa skladá z troch hlavných častí: lineárnej predikcie s výpočtom chyby predikcie, Golomb-Riceovho kódovania chyby predikcie a z vytvárania dátových rámcov pre tok komprimovaných dát. Na obrázku 6.1 je zobrazený dátový tok naprieč navrhovaným kompresným systémom. Výpočtová náročnosť a pamäťové požiadavky kompresie pomocou navrhovaných metód sú nízke a vhodné pre zdrojovo obmedzené výpočtové systémy ako sú MCU. Na komplexitu dekompresie dát nie je braný špeciálny ohľad, pretože predpokladom práce je iba kompresia na MCU. Navrhovaná metóda kompresie je založená na metóde uvedenej v [36], ktorú upravuje za cieľom dosiahnutia lepšej kompresie.



Obr. 6.1: Znázornenie dátového toku naprieč krokmi navrhovaného algoritmu.

6.1 Lineárna predikcia a výpočet chyby

Prvým krokom navrhovaného algoritmu je lineárna predikcia budúcej hodnoty signálu na základe jednej alebo niekoľkých minulých hodnôt. Ako bolo uvedené v sekcii 4.1 venovanej lineárnej predikcii, rôzne prediktory poskytujú rôznu presnosť pre rôzne priebehy signálu.

Z tohto dôvodu sa navrhuje využitie viacerých prediktorov, ktoré sa pre každú vzorku volia na základe lokálneho priebehu signálu. Adaptívne prispôsobenie prediktoru bolo využité v niekoľkých publikáciách (napr. [36, 10]) s rôznymi prístupmi. Prístup v navrhovanej metóde je založený na prístupe popísanom v sekcii 4.1.

Pôvodná varianta tejto metódy využíva pevne stanovenú hodnotu prahu *THR* (angl. threshold). Avšak rôzne hodnoty prahu sú vhodné pre rôzne časti EKG signálu. Pri nízkych prahových hodnotách sa pomocou adaptívne volených prediktorov generujú nižšie hodnoty chyby predikcie v častiach signálu s prudkými zmenami amplitúdy. Naopak pri vyšších prahových hodnotách je chyba predikcie nižšia v častiach signálu s plochým priebehom. To znamená, že obzvlášť v okolí QRS komplexov a prípadných anomálnych segmentov signálu je vhodné použiť nižšiu prahovú hodnotu, pričom pre zvyšok signálu je vhodné použiť prahovú hodnotu vyššiu. Z tohto dôvodu sa navrhuje úprava publikovanej metódy, kde úprava spočíva v adaptívnom prispôsobovaní prahovej hodnoty. Voľba tohto parametru THR by mala byť závislá iba na predošlých vzorkách, aby sa predišlo potrebe túto hodnotu zahrnúť do výsledného dátového toku. Voľba THR však musí byť nezávislá od využitia ďalšej prahovej hodnoty, čím by sa problém voľby správnej prahovej hodnoty iba hlbšie zanáral.

Dobrý odhad o zmene amplitúdy signálu je možné získať z relatívnych rozdielov hodnôt medzi susednými vzorkami. Ak definujeme súčasnú vzorku signálu ako $x[n]$, tak k-ta predošlá vzorka je vzorka $x[n-k]$. Rozdiely medzi hodnotami susedných vzoriek $d[k]$ je následne možné definovať vzorcom 6.1.

$$d[k] = x[n-k] - x[n-k-1] \tag{6.1}$$

Vo svojej podstate sa jedná o aproximáciu prvej derivácie diskrétneho signálu a hodnotu $d[k]$ je možné vnímať ako smernicu signálu v bode $x[n-k]$. Ako bolo uvedené vyššie, vzťah medzi lokálnym priebehom signálu a vhodnou výškou hodnoty prahu THR je inverzný. Z tohto dôvodu sa navrhuje využitie smerníc niekoľkých predošlých vzoriek k výpočtu vhodného prahu. Vzorec 6.3 definuje vzťah výpočtu vhodného prahu $\text{THR}_{\text{adaptive}}$, kde $\overline{w}$ je aritmetický priemer absolútnych hodnôt smerníc $d[k]$ pred súčasnou vzorkou daný vzorcom 6.2. Konštanta THR_MAX je maximálna povolená hodnota parametru $\text{THR}_{\text{adaptive}}$.

$$\overline{w} = \frac{1}{k}\sum_{i=0}^{k} |d[k]| \tag{6.2}$$

$$\text{THR}_{\text{adaptive}} = \begin{cases} \text{THR\_MAX} - \overline{w}^2, & \text{ak } \overline{w}^2 < \text{THR\_MAX} \\ 1, & \text{inak} \end{cases} \tag{6.3}$$

Následne sa vypočítaná hodnota $\text{THR}_{\text{adaptive}}$ použije pri adaptívnej voľbe prediktora tak, ako bolo popísané v sekcii 4.1. Na základe predikovanej hodnoty sa nakoniec vypočíta hodnota chyby predikcie $e[n]$, ktorá je využívaná v ďalšom kroku kompresie.

6.2 Golomb-Riceovo kódovanie

Vypočítaná chyba predikcie je v tomto kroku kódovaná do kódu premenlivej dĺžky Golomb-Riceovým kódovaním. Tento kód vyžaduje parameter k, na základe ktorého sa získava kvocient a zvyšok po delení kódovanej hodnoty týmto parametrom. Pôvodný prístup v uvedenej literatúre prepočítava tento parameter pre jednotlivé disjunktné okná signálu. Najprv sa získa suma všetkých absolútnych hodnôt už získaných chýb predikcie E_w v danom okne

(vzorec 6.4). Táto suma sa potom použije vo vzťahu 6.5 pre získanie parametru k, kde WS je veľkosť okna v počte vzoriek.

$$E_w = \sum_{i=0}^{\mathrm{WS}} |e[n-i]| \tag{6.4}$$

$$k = \log_2(E_w/\mathrm{WS}) \tag{6.5}$$

Vypočítaný parameter k je potom použitý pre kódovanie každej hodnoty $e[n]$ v danom okne. Výhodou tohto prístupu je, že je získavaná veľmi dobrá hodnota pre parameter k. Nevýhodou je nutné zakódovanie tejto hodnoty do kódového toku. Táto sekcia obsahuje dva návrhy na úpravu kroku Golomb-Riceovho kódovania z pôvodného článku [36].

6.2.1 Odhad parametru k

Signál EKG je relatívne pravidelný signál s niekoľkými markantmi v rámci periód. Tieto markanty môžu byť samotné vlny (napr. vlna P), QRS komplex, prípadne aj iné časti signálu. Využitím niekoľkých po sebe nasledujúcich markantov a prechodov medzi nimi je možné odhadnúť aká časť periódy po nich nasleduje. Keďže každá časť periódy vyžaduje inú hodnotu k, je možné predom priradiť vhodnú hodnotu k pre jednotlivé sledy takýchto markantov. Ak je možné s dobrou úspešnosťou takto odhadovať parameter k z predošlých vzoriek, nie je potrebné tento parameter kódovať do dátového toku. Za predpokladu, že takto definované hodnoty parametru k budú mať dobrú zhodu s hodnotami počítanými pomocou vzorca 6.5, tak je možné dostatočne ušetriť na kódovaní týchto hodnôt v dátovom toku, aby bol výsledný dátový tok kratší.

Tabuľka 6.1: Priradenia hodnôt k súčasnému oknu podľa identifikátorov podmienok štyroch predošlých okien. Okná sú zoradené od najstaršieho po okno predchádzajúce súčasnému.

ID podmienok predošlých okien				
E_{w-4}	E_{w-3}	E_{w-2}	E_{w-1}	Hodnota k súčasného okna
x	x	4	4	0
x	x	x	4	2
x	x	1	2	1
(4|3)	1	2	2	0
x	x	2	1	0
x	2	1	1	0
x	1	2	2	2
x	x	x	3	1
x	x	3	1	0
x	x	x	x	1

Navrhovaná úprava algoritmu naďalej pracuje nad disjunktnými oknami nad signálom. V rámci nich sa kategorizujú jednotlivé markanty. Pre rozpoznávanie markantov a častí periódy EKG signálu sa zavádzajú symboly ROC a MM_DIFF. Symbol ROC (angl. Rate-Of-Change) označuje hodnotu priemerného rozdielu medzi hodnotami vzoriek v rámci okna. Symbol MM_DIFF označuje hodnotu rozdielu medzi maximálnou a minimálnou hodnotou v rámci okna. Pomocou týchto hodnôt a k nim prináležiacich prahových hodnôt $\mathrm{THR}_{\mathrm{ROC}}$ a $\mathrm{THR}_{\mathrm{MM\_DIFF}}$ sa definujú nasledovné štyri pravidlá:

1. $|\text{ROC}| < \text{THR}_{\text{ROC}} \wedge \text{MM\_DIFF} < \text{THR}_{\text{MM\_DIFF}}$
2. $|\text{ROC}| < \text{THR}_{\text{ROC}} \wedge \text{MM\_DIFF} \geq \text{THR}_{\text{MM\_DIFF}}$
3. $|\text{ROC}| < \text{THR}_{\text{ROC}} \wedge \text{MM\_DIFF} \gg \text{THR}_{\text{MM\_DIFF}}$
4. $|\text{ROC}| \geq \text{THR}_{\text{ROC}} \wedge \text{MM\_DIFF} \gg \text{THR}_{\text{MM\_DIFF}}$.

Výraz $A \gg B$ je pravdivý ak hodnota A je výrazne väčšia, než hodnota B. Ak dané okno zapadá pod prvé pravidlo, jedná sa o prevažne plochý priebeh signálu. Druhé pravidlo deteguje nízke vlny ako sú vlny P alebo T. Tretie pravidlo deteguje veľmi krátke a veľmi veľké vychýlenia ako sú QRS komplexy. Posledné pravidlo deteguje krátke a vysoké vychýlenie signálu na hranici okna. Podmienky sa vyhodnocujú sekvenčne a identifikátor splnenej podmienky sa pre dané okno uloží. Identifikátor splnenej podmienky sa udržiava pre štyri predošlé okná a na základe týchto identifikátorov sa odhaduje prináležiaca hodnota k pre súčasné okno. Tabuľka 6.1 zobrazuje pri akých sledoch okien sa priradí aká hodnota k súčasnému oknu. Pre ilustráciu princípu sa využije prvý riadok tabuľky. Ak sa po sebe nachádzajú dve okná typu 4, tak súčasné okno sa nachádza v časti signálu tesne po QRS komplexe. Pre toto okno je na základe pozorovaní vhodné použiť hodnotu parametru $m = 0$.

Tento odhad parametru k bol navrhnutý na základe pozorovaní signálov s nízkym množstvom anomálií zachytených na zvode MLII. Vzhľadom na veľkú variabilitu skutočného EKG signálu medzi jednotlivými zvodmi sa od tejto naivnej úpravy algoritmu neočakávajú príliš dobré výsledky. Avšak v prípade, že by dosiahnutá kompresia dosahovala výsledky nanajvýš niekoľkonásobne horšie, než pôvodný algoritmus, otvárala by sa možnosť doladenia navrhnutého prístupu.

6.2.2 Úprava geometrického rozloženia chýb predikcie

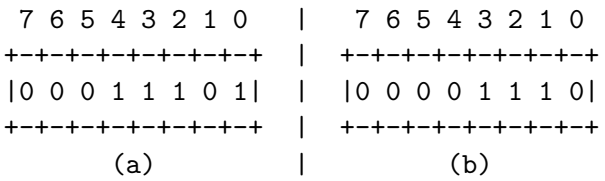
Ako bolo uvedené v sekcii 3.3.2, Golomb-Riceovo kódovanie je kompresne najefektívnejšie v prípade, že distribúcia kódovaných hodnôt odpovedá geometrickej distribúcii. Algoritmus adaptívneho prispôsobenia prediktoru (viď sekciu 4.1.1) vyžaduje bezstratovú kompresiu signálu aby bola rekonštrukcia možná. Teda ak existuje prístup, ktorý dokáže bez straty informácie transformovať distribúciu chýb predikcie tak, aby výsledná distribúcia obsahovala výrazne vyšší počet nízkych hodnôt, je možné vylepšiť efektívnosť Golomb-Riceovho kódovania.

Za predpokladu, že touto prácou upravovaná metóda [36] skutočne generuje chyby predikcie v približne geometrickej distrubúcii, je možné túto distribúciu jednoducho dodatočne upraviť. Táto úprava spočíva v „zúžení“ distribúcie chýb predikcie bitovým posunutím *už mapovaných* (viď sekciu 3.3.2) chýb predikcie vpravo. Tento vzťah je popísaný vzorcom 6.6, kde $M(e[n])$ je mapovaná chyba predikcie a zápis $>> 1$ je bitový posun vpravo o 1 bit.

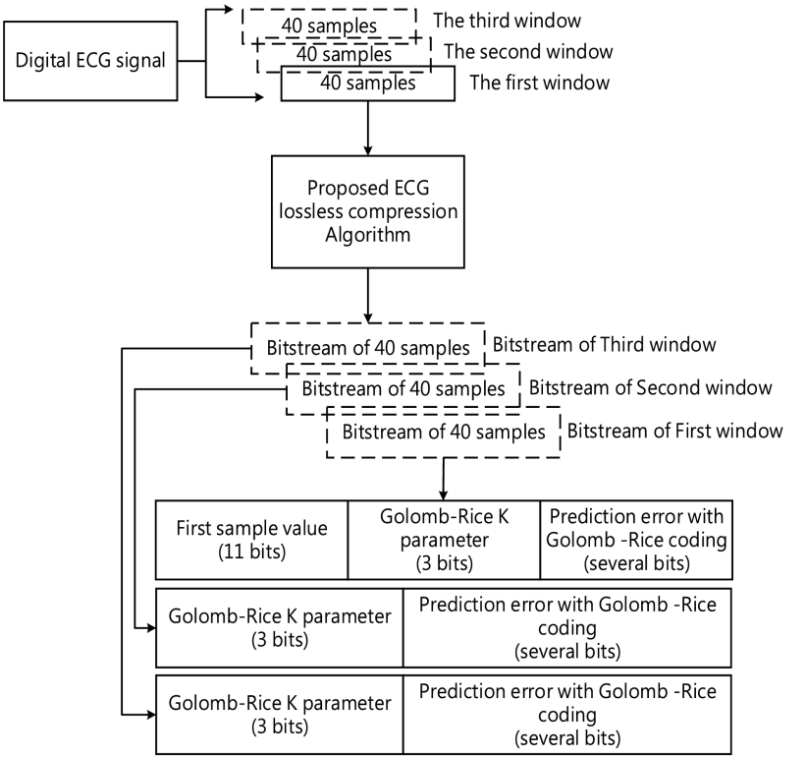
$$\overrightarrow{M(e[n])} = M(e[n]) >> 1 \tag{6.6}$$

Na tom, či sa jedná o logický alebo aritmetický posun, nezáleží, pretože zachovať znamienkový bit nie je nutné vďaka predošlému mapovaniu chýb predikcie na výlučne kladné hodnoty. Bitovým posunutím týchto hodnôt (efektívne celočíselným vydelením hodnotou 2) sa získa distribúcia hodnôt vhodnejšia pre Golomb-Riceovo kódovanie. Avšak týmto posunutím sa stráca najmenej významný bit, a teda informácia o parite pôvodnej hodnoty. Pre ilustráciu je uvedený obrázok 6.2.

Aby bola zaistená bezstratovosť metódy, je nutné odstránený bit každej hodnoty chyby predikcie dodatočne kódovať do výsledného dátového toku (viď sekciu 6.3). Výsledkom celej



Obr. 6.2: Ilustrácia bitového posunutia (a) 8-bitovej hodnoty 29 o jeden bit vpravo, ktorého výsledkom je (b) s hodnotou 14, čo odpovedá celočíselnému deleniu dvoma.



Obr. 6.3: Formát dátového toku. Prevzaté z [36].

tejto úpravy kódovania je výrazne skrátený (v priemere o viac bitov, než 1) Golomb-Riceov kód pre každú hodnotu chyby predikcie, pričom je následne pridaný iba 1 dodatočný bit parity pôvodnej hodnoty. Dekódovanie je plne reverzibilné pomocou vzťahu 6.7, kde p je paritný bit odstránený po aplikovaní vzťahu 6.6 a zápis $<< 1$ je bitový posun vľavo o 1 bit. Je nutné poznamenať, že hodnota parametru k Golomb-Riceovho kódovania je počítaná na základe *tu uvedeným prístupom už upravených* hodnôt chýb predikcie $\overrightarrow{M(e[n])}$.

$$M(e[n]) = (\overrightarrow{M(e[n])} << 1) + p \tag{6.7}$$

6.3 Vytváranie dátového toku

Posledným krokom kompresie je vytvorenie dekódovateľného dátového toku komprimovaných dát. Prvá hodnota dátového toku je nekomprimovaná počiatočná hodnota EKG signálu, ktorá je potrebná pre úspešnú dekompresiu. Zvyšok toku sa skladá z dvoch typov dát, a to z parametru k pre Golomb-Riceovo kódovanie a z kódovaných hodnôt chyby lineárnej

predikcie. Po jednej hodnote k kódovanej na 3 bity nasleduje toľko hodnôt chyby predikcie, akú hodnotu malo WS nastavujúce veľkosť okna pre výpočet k. Tieto dva druhy dát sa opakujú až do konca toku. Znázornenie formátu dát v toku je ukázané na obrázku 6.3.

Pri využití prístupu s odhadom parametru k (sekcia 6.2.1) nie je nutné tento parameter do dátového toku kódovať, čiže tento krok tvorby toku je možné vynechať. Pri využití prístupu úpravy distribúcie rozloženia (sekcia 6.2.2) sa naopak do dátového toku pridáva ešte jeden dodatočný bit parity pôvodnej hodnoty chyby predikcie. Tento bit je do toku pridaný za každú jednu kódovanú hodnotu, ktorej daný paritný bit prináleží.

Kapitola 7

Implementácia navrhnutého algoritmu

Táto kapitola, rovnako ako samotná implementácia, je rozdelená na dve časti: prototyp algoritmu v jazyku Python a implementácia pre mikrokontrolér (MCU) v jazyku C. Vzhľadom na to, že navrhovaný algoritmus sa zakladá na algoritme navrhnutom Tsai et al. [36], sú implementované dve verzie algoritmu – pôvodná verzia bez modifikácií a upravená verzia s adaptívnym prispôsobením prahovej hodnoty THR a modifikáciou distribúcie hodnôt chýb predikcie. Každá relevantná funkcia a premenná je preto pomenovaná na základe tohto rozdelenia, kde ich názvy obsahujú buď podreťazec `tsai` alebo `nemeth`.

7.1 Python prototyp

Táto implementácia slúži pre jednoduchšie a rýchlejšie prototypovanie navrhnutého algoritmu a zároveň slúži pre vyhodnocovanie jeho kompresného výkonu. Bola využitá verzia `Python 3.9.2` s niekoľkými rozširujúcimi balíčkami[1].

Algoritmus uvedený v kapitole 6 predpokladá, že vstupný signál nie je dostupný celý, ale je priebežne prijímaný vzorka po vzorke (online algoritmus). V prípade vyhodnocovania kompresného výkonu pomocou dopredu získaných signálov však nie je nutné vstupný signál spracovávať po vzorkách. Takéto vyhodnocovanie by v jazyku Python ani nebolo žiaduce vzhľadom na implicitne neefektívnu iteráciu. Z tohto dôvodu bol využitý aj rozširujúci balíček `numpy`, ktorý podporuje jednoduchú vektorizáciu rôznych operácii nad dátami. Takmer každá funkcia pracujúca nad spracúvaným signálom má v implementácii dve verzie. Jedna je „naivnou“ implementáciou bez vektorizácie a druhá s podporou vektorizácie. Vektorizované verzie majú príponu `_faster`. Dôvodom za takouto dvojitou implementáciou bolo jednoduchšie overovanie správnosti vektorizovaného kódu.

Python prototyp je rozdelený do logických celkov a skladá sa z niekoľkých zdrojových súborov:

- `Predictor.py` definuje obaľovaciu triedu pre prediktory rôznej dĺžky a metódu pre predikciu hodnoty na základe vstupných dát,
- `deciders.py` definuje funkcie pre voľbu správneho prediktoru pre jednotlivé vzorky,

[1] Viď súbor `requirements.txt`.

- `rice.py` definuje funkcie pre kódovanie/dekódovanie hodnôt do/z Golomb-Riceovho kódu a pomocné funkcie pre vytváranie dátového (bitového) toku,
- `main.py`, v ktorom sa implementuje samotné spracovanie vstupného signálu navrhovaným algoritmom
- a `evaluation.py`, ktorý slúži na vyhodnotenie implementácie.

7.1.1 Načítanie signálu a výpočet predikcie signálu

Prvým krokom spracovania je načítanie EKG signálu do pamäte. Vstupné signály využívané pri testovaní a vyhodnocovaní boli získané z webstránok PhysioNet[2]. Dáta poskytované na týchto stránkach sú ukladané vo formáte WFDB (WaveForm DataBase). Pre narábanie s týmto formátom sú dodávané referenčné implementácie knižníc v rôznych jazykoch. Pre jazyk Python je poskytovaný balíček `wfdb`, ktorý bol využitý pre načítavanie EKG signálu. Celý signál je možné jednorazovo načítať pomocou funkcie `wfdb.rdrecord()`, ktorá signál uloží ako celočíselné `numpy` pole.

Ďalším krokom je získanie hodnôt predikcií na základe implementovaného algoritmu. Tieto hodnoty sú počítané funkciami `prediction_*()`, kde sa najprv pripravia výstupné `numpy` polia pre predikované hodnoty. Následne je celý vstupný signál spracovaný vzorka po vzorke, kde sa pre každú vzorku získa vhodný prediktor zo sady prediktorov pomocou funkcií `decider_*()`. Tento prediktor je nakoniec použitý pre výpočet predikovanej hodnoty, ktorá sa ukladá do predalokovaného výstupného `numpy` poľa.

```
        Vzorky signálu | 1 2 3 4 5 6 7 8 9 |
---------------------------------------------
Oneskorenie o 1 vzorku | x 1 2 3 4 5 6 7 8 |
Oneskorenie o 2 vzorky | x x 1 2 3 4 5 6 7 |
Oneskorenie o 3 vzorky | x x x 1 2 3 4 5 6 |
                               /  |
          Trojica pre vzorku 4  /
            Trojica pre vzorku 5
```

Obr. 7.1: Ukážka vytvárania n-tíc predchádzajúcich n vzoriek signálu, kde n-tica každej vzorky je stĺpec oneskorených signálov pod danou vzorkou. Zobrazené sú trojice, teda $n = 3$.

Vektorizované verzie týchto funkcií sú výrazne rýchlejšie, avšak vyžadujú omnoho viac pamäte z dôvodu vytvárania niekoľkých pomocných polí o rovnakej veľkosti ako je vstupný signál. Vzhľadom na využívanie prediktorov prvého až tretieho rádu, sú vytvorené tri reorganizované „pohľady“ na vstupný signál. Táto reorganizácia spočíva vo vytvorení n-tíc predchádzajúcich n vzoriek pre každú vzorku vstupného signálu, kde n závisí od rádu prediktoru. Tzn. pri prediktore prvého rádu sa jedná o jednoduchý posun signálu vpravo (teda jeho oneskorenie) o jednu vzorku. Pri prediktore tretieho rádu sa jedná o vytvorenie trojíc predchádzajúcich troch vzoriek. Oneskorenie signálu je prevedené pomocou vektorizovaného `numpy.roll()`. Pre ilustráciu vytvorených trojíc oneskoreného signálu viď obrázok 7.1, kde každý stĺpec oneskorení predstavuje jednu trojicu pre danú vzorku signálu. Prvých n získaných n-tíc je však, samozrejme, neplatných kvôli neexistujúcim vzorkám pred začiatkom signálu. Po vytvorení n-tíc pre každý z rádov prediktorov, je možné nad každou n-ticou

[2]Odkaz na PhysioNet: https://www.physionet.org/

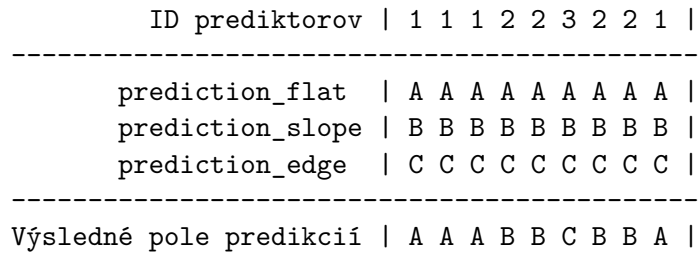
každej vzorky vykonať vektorizované násobenie koeficientmi daného prediktoru. Nakoniec sa každá n-tica sčítaním redukuje na výslednú hodnotu predikcie. Načrtnutie týchto výpočtov je zobrazené na obrázku 7.2, kde sú zobrazené rovnaké n-tice ako na predchádzajúcom obrázku, akurát sú orientované zvislo. Koeficienty prediktoru tretieho rádu sú uvedené nad n-ticami. Prvý krok zobrazuje násobenie každého stĺpca prináležiacim koeficientom prediktoru a druhý krok ukazuje redukciu sčítaním vo vodorovnej osi. Výsledkom vykonania týchto krokov pre každý prediktor sú tri polia (`prediction_flat`, `prediction_slope` a `prediction_edge`) s predikovanými hodnotami jednotlivých prediktorov pre celý signál.

```
prediktor   3  -3   1
           -------------       -------------          -----
           | x   x   x |       | x   x   x |          | x |
           | 1   x   x |       | 3   x   x |          | x |
           | 2   1   x |       | 6  -3   x |          | x |
           | 3   2   1 |       | 9  -6   1 |    ADD   | 4 |
n-tice     | 4   3   2 |--MUL->| 12 -9   2 |--REDUCE->| 5 |
           | 5   4   3 |       | 15 -12  3 |          | 6 |
           | 6   5   4 |       | 18 -15  4 |          | 7 |
           | 7   6   5 |       | 21 -18  5 |          | 8 |
           | 8   7   6 |       | 24 -21  6 |          | 9 |
```

Obr. 7.2: Ukážka vektorizovaného výpočtu predikovaných hodnôt signálu pomocou prediktoru tretieho rádu.

Po predpočítaní predikcií pomocou všetkých prediktorov je nutné zistiť, na ktorom indexe signálu použiť ktorú predikovanú hodnotu. Toto je vykonané vo funkciách pomenovaných `deciders_*_faster()`, ktoré fungujú na podobnom princípe posúvania signálu ako pri výpočte predikcií. Všetky potrebné pomocné hodnoty pre voľbu prediktoru sa vektorizovane predpočítajú, po čom nasleduje sled volaní funkcií `numpy.where()`. Tieto slúžia ako vektorizovaná náhrada rozhodovacej `if/elseif/else` štruktúry pre voľbu prediktoru. Pre umožnenie jednoduchej vektorizácie je každému z prediktorov priradený celočíselný identifikátor, teda napr. 1 až 3 pre prediktory prvého až tretieho rádu. Identifikátory zvolených prediktorov sa pomocou `numpy.where()` zapíšu do výstupného poľa, kde identifikátor prediktoru na i-tej pozícií značí voľbu predikovanej hodnoty daným prediktorom na i-tej pozícií signálu. Toto pole vyplnené identifikátormi prediktorov je následne vrátené a nakoniec sa jednoduchým použitím `numpy.where()` vytvorí výsledné pole predikcií na základe získaných identifikátorov. Na obrázku 7.3 je načrtnuté ako sa vytvára výsledné pole hodnôt predikcií $\hat{x}$ z troch pomocných polí pre jednotlivé prediktory a z poľa identifikátorov prediktorov. Po získaní hodnôt predikcií $\hat{x}$ je možné vypočítať chybu predikcie pomocou vzťahu $e[n] = x[n] - \hat{x}[n]$. Tieto hodnoty chybového signálu e budú v ďalšom kroku kódované Golomb-Riceovým kódovaním.

Je vhodné uviesť, že originálny článok [36] obsahuje chybu v rámci definície popisovaného algoritmu. Ako je možné vidieť na obrázku 4.3, jeden z krokov pre voľbu vhodného prediktoru je duplicitný, a teda nemá vplyv na algoritmus. Tento krok bol v implementácii vynechaný.



Obr. 7.3: Načrtnutie vektorizovanej voľby hodnôt predikcií na danom indexe poľa na základe identifikátorov prediktorov.

7.1.2 Vytváranie výstupného dátového toku

Golomb-Riceovo kódovanie je vykonávané vo funkcii `encode()`. Táto funkcia obaľuje počítanie parametru k pre Golomb-Riceov kód (funkcia `estimate_m()`) a samotné kódovanie hodnôt chýb predikcie (funkcia `rice_encode()`). Vo funkcii `rice_encode()` je vykonávané aj mapovanie hodnôt chýb predikcie na výlučne nezáporné hodnoty. Vzhľadom na vlastnosť premenlivej dĺžky Golomb-Riceových kódov je tiež využívaná pomocná funkcia `unpackbits()`. Táto funkcia slúži na vytváranie Python zoznamu, ktorá obsahuje binárnu reprezentáciu vstupného celého čísla na špecifikovaný počet bitov. Pre ilustráciu volanie funkcie `unpackbits(0b101, 5)` vytvorí zoznam `[0,0,1,0,1]`. Výstupný dátový tok kodéru je reprezentovaný zoznamom hodnôt 0 a 1.

Na začiatok dátového toku sú vložené priame, nekomprimované, hodnoty pôvodného signálu pre umožnenie dekompresie. Z dôvodu jednoduchšej implementácie a takmer žiadnej pridanej hodnoty pri kompresii dlhých signálov je v implementácii upravená štruktúra výstupného dátového toku oproti popisu algoritmu v sekcii 6.3. V návrhu sa totiž na začiatok výsledného toku vkladá jediná priama hodnota pôvodného signálu. Toto obmedzenie bolo v implementácii odstránené a bola pridaná možnosť vložiť ľubovoľný počet priamych hodnôt na začiatok toku. Prítomnosť jedinej priamej hodnoty na začiatku toku vyžaduje pri dekódovaní komplikovanejšie štartovanie dekodéru pri adaptívnej voľbe prediktorov. Do výsledného toku sa touto úpravou pridá o malý počet viac bitov, avšak sa tým zjednodušuje komplexnosť kodeku. V prípade, že by do dátového toku boli pravidelne pridávané priame hodnoty (napr. z dôvodu zotavenia sa pri výpadku signálu), už by bolo vhodné počet týchto priamych hodnôt obmedziť za cenu komplexnejšieho kodeku. V tejto práci však nebol braný ohľad na zotavenie sa z výpadku dátového toku.

Po zakódovaní počiatočných priamych hodnôt signálu sa začnú vkladať kódované hodnoty chyby predikcie. Tu neboli vykonané žiadne úpravy oproti návrhu. Do toku sa striedavo vkladajú hodnoty parametru k a taký počet kódovaných chýb predikcie, aké veľké je okno pre výpočet parametru k. Veľkosť výsledného zoznamu s kódovanými hodnotami `bitstream` je možné získať prečítaním jeho dĺžky `len(bitstream)`.

Za účelom ľahšej kontroly korektnosti algoritmu bol implementovaný aj dekodér funkciou `decode()`. Jedná sa o jednoduchý cyklus iterujúci cez celý tok. Po každých WS dekódovaných hodnotách sa načítajú tri bity kódujúce hodnotu k. Táto hodnota je konvertovaná na celočíselnú hodnotu pomocou pomocnej funkcie `packbits()`. Následne sa dekódujú jednotlivé hodnoty chýb predikcie. Tento krok je znázornený algoritmom 1, ktorý je na prvých dvoch riadkoch inicializovaný hodnotou $m = 2$ a dátovým tokom `stream` obsahujúci jednu kódovanú hodnotu. Dekódovanie prebieha tak, že sa najskôr prečítajú hodnoty q a r. Hodnota q je kódovaná unárne a je dekódovaná na riadkoch 5 až 9. Hodnota r je kódovaná

ako priame binárne číslo na k bitov, a teda je možné túto hodnotu jednoducho prečítať, k čomu slúži funkcia `packbits()` (riadok 11). Po získaní hodnôt q a r je možné dekódovať kódovanú hodnotu na riadku 12. Túto hodnotu je nakoniec ešte nutné spätne mapovať na pôvodnú hodnotu, keďže tá mohla byť zápornou hodnotou. Týmto reverzným mapovaním sa získava kódovaná hodnota -10.

```
1  stream = [ 1, 1, 1, 1, 0, 1, 1 ]
2  m = 2
3  i = 0
4  while (i < len(stream)):
5      q = 0
6      while (stream[i] != 0):
7          q += 1
8          i += 1
9      i += 1
10
11     r = packbits(stream[i:i+m], m)
12     tmp = (q * (2 ** m)) + r
13     decoded = -(tmp // 2) - 1 if tmp & 0b1 else (tmp // 2)
```

Algoritmus 1: Algoritmus dekódovania Golomb-Riceovho kódu v Pythone.

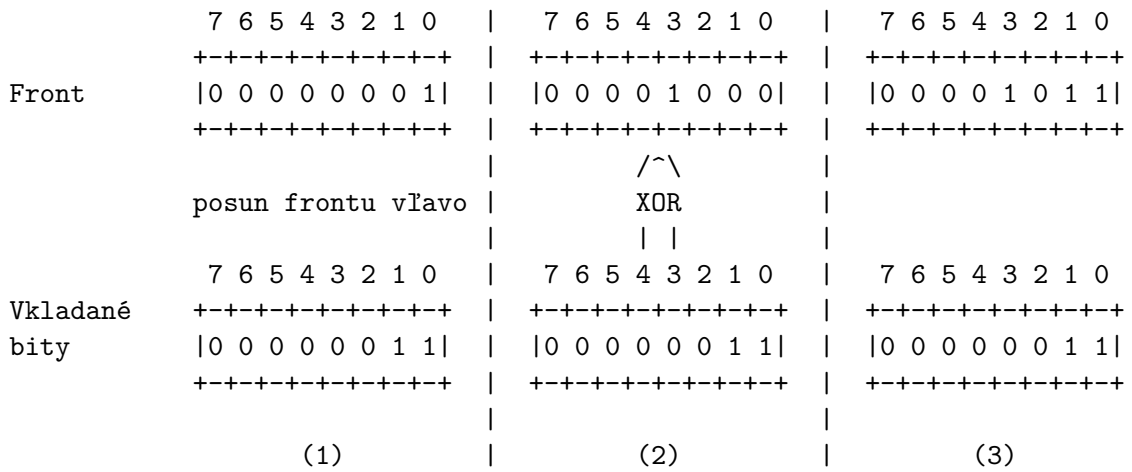
7.2 C implementácia

Postup implementácie bol dvojfázový. Najskôr sa implementovala algoritmická časť za účelom jednoduchšieho testovania mimo MCU platformy, na PC. Vývoj prebiehal na operačnom systéme Debian GNU/Linux 11 (bullseye) x86_64 s kompilátorom GCC verzie 10.2.1. Po overení funkčnosti algoritmu v tejto implementácií, boli aplikované potrebné zmeny zdrojového kódu pre umožnenie jeho kompilácie pre MCU. Využitá vývojová doska bola nRF52 DK od výrobcu Nordic Semicondructor s nRF52832 SoC (angl. System on Chip). Uvedený čip obsahuje procesor ARM Cortex M4 s FPU jednotkou. Kompilovanie pre MCU bolo vykonávané pomocou nRF Connect SDK v2.3.0. Zdrojový kód je rozdelený na niekoľko logických celkov, podobne ako v Python implementácií:

- `prediction.c/h` implementuje výpočet predikcie nasledujúcej vzorky,
- `decider.c/h` implementuje voľbu prediktoru,
- `rice.c/h` implementuje Golomb-Riceovo kódovanie a funkcie umožňujúce kódovanie ľubovoľných hodnôt do bitovej postupnosti,
- `samples.c/h` implementuje rozhranie pre jednoduché použitie kódu na PC aj na MCU,
- `main.c` spája vyššie uvedené časti algoritmu a spracúva nimi vstupný signál.

7.2.1 Načítanie signálu a výpočet predikcie signálu

Výpočet predikcie a voľby prediktoru je takmer identický s „naivnou“ verziou Python implementácie. Jediné rozdiely spočívajú v prepísaní prípadných slice operátorov, `numpy` redukcií a iných syntaktických a funkcionálnych vymožeností špecifických pre Python.



Obr. 7.4: Kroky vloženia troch bitov `011` uložených v 8-bitovom dátovom type do 8-bitového frontu. Krok (1) zobrazuje pôvodný stav frontu a vkladaných bitov, (2) je front posunutý vľavo o tri bity a (3) je výsledkom vloženia troch vkladaných bitov pomocou operátoru XOR.

Zmeny boli vykonané v získavaní vstupného signálu. Počas testovania na PC boli využívané záznamy databázy WFDB pomocou poskytovanej `WFDB` knižnice pre jazyk C. Táto knižnica sa používa iným spôsobom oproti Python verzii. Vstupný signál sa nenačítava celý a jednorazovo, ale jednotlivé vzorky signálu sú získavané pomocou funkcie `getvec()`. Zdrojový kód je ošetrený preprocesorovými direktívami, ktoré kontrolujú prítomnosť definovaného makra `USE_WFDB`. Toto makro je na PC definované pri invokácií GCC kompilátoru, a teda počas kompilácie pre MCU je jeho definícia vynechaná. Týmto mechanizmom je možné využiť rovnaké zdrojové súbory na oboch platformách. V prípade, že sa kompiluje pre MCU, je do kompilácie zahrnutý aj obsah súborov `samples.c/h`. V rámci nich je definované pole so vzorkami EKG signálu určenými na vyhodnocovanie algoritmu na MCU. Je v nich definovaná aj funkcia `getvec()` upravená pre prácu s týmto poľom vzoriek.

7.2.2 Vytváranie výstupného dátového toku

Implementácia Golomb-Riceovho kódovania je od Python prístupu výrazne odlišná. Keďže sa jedná o online algoritmus implementovaný na MCU, nie je vhodné lokálne ukladať výsledný dátový tok, a teda bol využitý UART (angl. Universal Asynchronous Receiver-Transmitter) pre overovanie funkčnosti implementácie. V tejto implementácií bol braný aj väčší ohľad na pamäťovú a výpočtovú efektivitu konštrukcie výsledného dátového toku. Kodér a všetky pomocné funkcie v súbore `rice.c` pracujú s kontextom kodéru definovaným nasledovne:

```
typedef struct rice_ctx
{
  uint32_t shift_cnt;
  uint8_t buffer[BUFF_LEN];
} rice_ctx_t;
```

Kodér pracuje s pakovacím[3] zásobníkom `buffer`, do ktorého sa vkladajú postupnosti bitov. Pakovanie bitov spočíva v naplnení dátovej štruktúry zarovnanej na istý počet bitov

[3] V slovenčine pakovať = baliť. Tento preklad bol zvolený pre anglický termín *bit packing* pre jeho fonetickú a zároveň sémantickú podobnosť.

sledom hodnôt, ktoré sú reprezentované menším počtom bitov. Pre načrtnutie, do 8-bitového dátového typu je možné uložiť dve 4-bitové hodnoty. Tento princíp napĺňania dátovej štruktúry využívajúc každý bit ako nosič informácie je využívaný počas tvorby výstupného toku. Dátovú štruktúru `buffer` je možné vnímať ako front (angl. queue), z ktorého sú dáta vyplachované na UART akonáhle sa naplní. Naplnenie je sledované atribútom `shift_cnt`, ktorý je inkrementovaný každým vloženým bitom.

Vkladanie do frontu je vykonávané funkciou s prototypom `uint32_t pushl(uint32_t bits, uint8_t n_bits, rice_ctx_t *ctx)`. Táto funkcia sa pokúsi vložiť `n_bits` počet bitov uložených v parametri `bits` do frontu. Počet `n_bits` sa počíta od najmenej významného bitu parametru `bits`, pričom prebytočné bity musia byť nulové, inak sa môžu prepísať už uložené bity vo fronte. Načrtnutie fungovania funkcie `pushl()` je zobrazené na obrázku 7.4. Prvým krokom je posunutie frontu vľavo o `n_bits` bitov, čím sa uvoľní nulami inicializovaný priestor pre vkladané bity. Následne sa `bits` vložia na uvoľnené miesto XOR operáciou, čím sa získava výsledná podoba frontu. Funkcia vracia počet bitov, ktoré sa podarilo vložiť do frontu. V prípade, že vrátená hodnota sa líši od hodnoty `n_bits`, tak je nutné vypláchnuť frontu na UART a resetovať hodnotu `shift_cnt` pomocou funkcie `flush_buffer_with_shift_reset()`, po čom je možné vložiť zvyšné bity.

Kapitola 8

Vyhodnotenie navrhnutého algoritmu

Vyhodnotenie sa, podobne ako kapitola 7, skladá z dvoch častí. Prvá je zameraná na vyhodnotenie navrhovaného algoritmu v zmysle dosiahnutej kompresie. Pre vyhodnotenie tejto časti bola využitá implementácia algoritmu v jazyku Python. V druhej časti je uvedené vyhodnotenie implementácie algoritmu v jazyku C na konkrétnom mikrokontroléri.

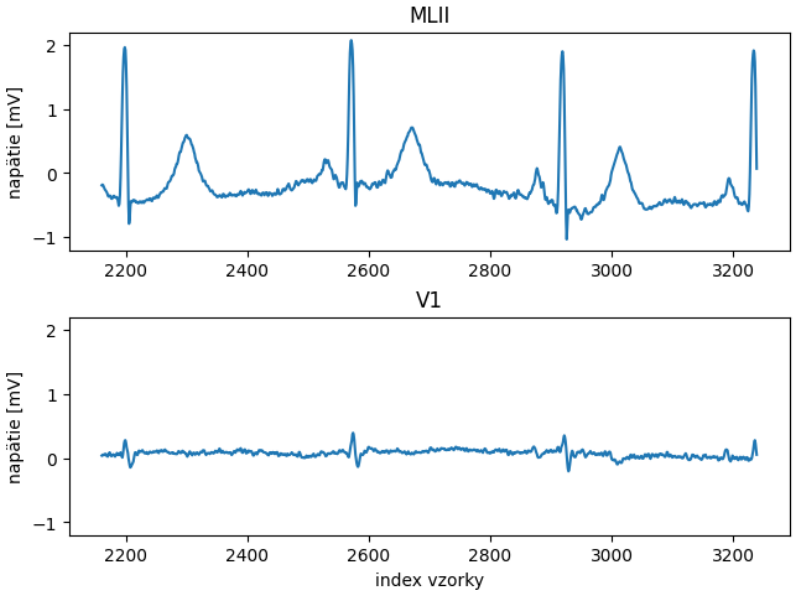
8.1 Voľba dátovej sady

Vyhodnocovanie bolo vykonávané nad databázou MIT-BIH (viď [25]). Táto databáza sa skladá zo 48 záznamov EKG. Každý zo záznamov je dvojkanálový, kde na každom z kanálov je paralelne zaznamenávaný iný zvod EKG. Každý záznam obsahuje presne 650000 vzoriek signálu pri vzorkovacej frekvencii 360 Hz a s rozlíšením 11 bitov naprieč 10 mV rozsahom [25]. Celá databáza sa skladá zo 46 signálov zachyteného na zvode MLII, 40 signálov zvodu V_1, 5 signálov zvodu V_5, 4 signálov zvodu V_2, a 1 signálu zvodu V_4.

Dáta záznamov v databáze MIT-BIH sú ukladané pomocou metódy pakovania bitov (viď sekciu 7.2.2 pre ukážku pakovania). Význam takéhoto ukladania dát spočíva v schopnosti ukladať vzorky inej dĺžky, než násobkov ôsmych bitov, bez plýtvania prebytočných nevyužitých bitov. Z tohto dôvodu posledná vzorka záznamu nemusí byť zarovnaná na rozmedzie 8 bitov. Veľkosti súborov sú však v moderných operačných systémoch typicky zarovnávané na Byty, prípadne ich násobky. Z tohto dôvodu sa pre presné zistenie bitovej dĺžky záznamu využil vzťah 8.1. Táto bitová dĺžka bola následne využívaná pri vyhodnocovaní algoritmov.

$$\text{signal\_bitlen} = \text{sample\_bitlen} * \text{signal\_len} \tag{8.1}$$

Na obrázku 8.1 je zobrazená časť signálu dvoch zvodov záznamu 106 databázy MIT-BIH. Na tomto obrázku je možné pozorovať rôznorodosť EKG signálov, vrátane ich tendencie „plávať“, ako je možné vidieť na signáli zvodu MLII. Tento úkaz je po anglicky nazývaný *baseline drift*, prípadne *baseline wander*. Ukážka signálov tiež dobre zobrazuje zjavnú odlišnosť rôznych EKG signálov, s čím sa kompresné algoritmy pre takéto signály tiež musia vedieť vysporiadať.



Obr. 8.1: Časť signálov oboch zaznamenaných zvodov záznamu 106 databázy MIT-BIH.

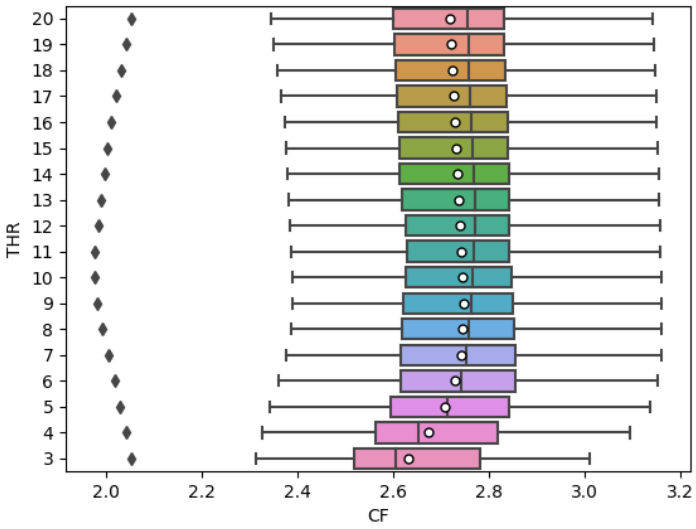
8.2 Vyhodnotenie parametrov algoritmu

Vyhodnocovanie bolo vykonávané na štyroch variantách algoritmu: **1)** na reimplementácii nemodifikovaného algoritmu z [36], **2)** na implementácii s úpravou odhadu prahovej hodnoty $\text{THR}_{\text{adaptive}}$, **3)** na predošlej uvedenej implementácii doplnenej o úpravu distribúcie hodnôt chýb predikcie a nakoniec **4)** na implementácii s naivným odhadom hodnoty k pre Golomb-Riceovo kódovanie. Vyhodnocovalo sa pomocou Python skriptu `evaluation.py`.

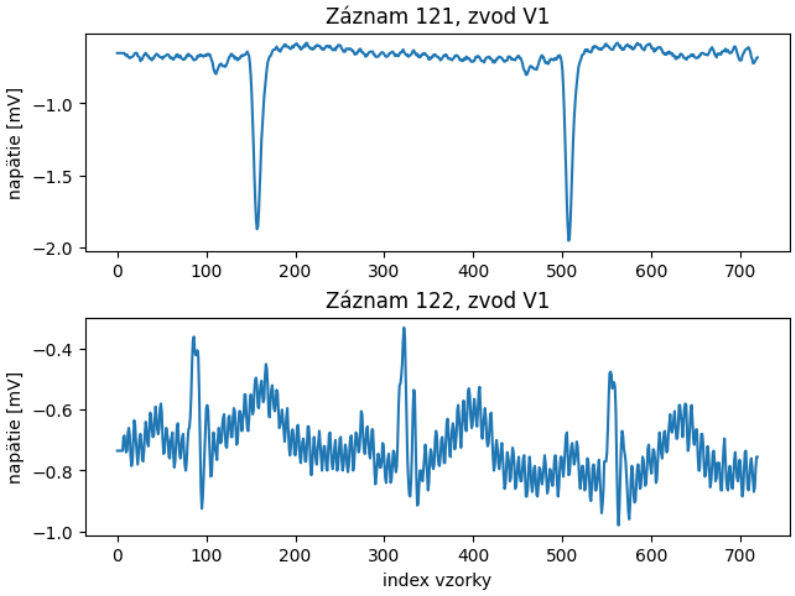
8.2.1 Voľba parametru $\text{THR}_{\text{static}}$

Vzhľadom na nešpecifikovanú hodnotu parametru THR v pôvodnom článku [36] (tu označovaný ako $\text{THR}_{\text{static}}$), boli vyhodnotené všetky hodnoty od 3 do 20. Algoritmus bol spustený nad celou databázou MIT-BIH pre každú z hodnôt. Na obrázku 8.2 je možné vidieť boxplot pre každú z uvedených hodnôt parametru. Do boxplotu boli zahrnuté aj priemerné hodnoty kompresného faktoru naprieč databázou, ktoré sú zobrazené bielymi krúžkami. Je možné vidieť, že so zvyšujúcou sa hodnotou parametru $\text{THR}_{\text{static}}$ sa prudko zvyšuje kompresná výkonnosť až po hodnoty 8 až 12, kde sa výsledky približne ustália. Následne je ďalším zvyšovaním hodnoty parametru možné vidieť klesajúci trend. Na základe výsledkov sa pre porovnávanie pôvodného algoritmu s verziami s navrhnutými modifikáciami využívala hodnota $\text{THR}_{\text{static}} = 9$.

Na obrázku 8.2 je vidieť aj jednu odľahlú hodnotu kompresného faktoru. Táto odľahlá hodnota CF má približne opačný priebeh so zvyšujúcim sa $\text{THR}_{\text{static}}$ oproti zvyšným. Tento dátový bod prináleží signálu zvodu V_1 záznamu 122. Jedná sa o zdravý srdcový signál s iba dvoma anomálnymi fragmentami, avšak celý tento signál obsahuje veľké množstvo vysokofrekvenčného šumu. Porovnanie signálov V_1 medzi záznamami 121 a 122 je možné vidieť na obrázku 8.3. Prítomný vysokofrekvenčný šum je tak prevalentný, že najlepších výsledkov kompresie sa na ňom dosahuje pomocou veľmi nízkej hodnoty $\text{THR}_{\text{static}}$.



Obr. 8.2: Znázornenie kompresného faktoru metódy Tsai et. al. [36] pri použití rôznych hodnôt parametru THR.



Obr. 8.3: Porovnanie prvých dvoch sekúnd signálov V_1 záznamov 121 a 122 MIT-BIH databázy.

8.2.2 Porovnanie navrhnutých úprav s prístupom $\mathbf{THR_{static}}$

Po zvolení najlepšej hodnoty THR_{static} pre pôvodnú kompresnú metódu bolo vykonané porovnanie tejto metódy s navrhovanými metódami adaptívneho prispôsobovania prahovej hodnoty $THR_{adaptive}$ a s dodatočnou úpravou distribúcie hodnôt chýb predikcie. Veľkosti okien pre výpočet $THR_{adaptive}$ boli testované od 3 do 5, pričom najlepšie výsledky boli dosiahnuté s oknom veľkosti 5. Tabuľka 8.1 zobrazuje porovnanie kompresných faktorov pre

prvých 10 záznamov databázy MIT-BIH[1]. Je uvedený aj percentuálny rozdiel PR oproti pôvodnej metóde s $\mathrm{THR}_{\mathrm{static}}$, ktorý bol vypočítaný podľa vzťahu 8.2, kde v_p označuje pôvodnú hodnotu porovnania a v_n naopak hodnotu novú. Kladná hodnota rozdielu značí zlepšenie kompresného faktoru novou metódou a záporná zhoršenie.

$$\mathrm{PR} = \frac{v_p - v_n}{v_p} \cdot 100 \quad [\%] \tag{8.2}$$

Tabuľka 8.1: Porovnanie pôvodnej kompresnej metódy s navrhovanou kompresnou metódou s adaptívnym prispôsobením parametru THR s oknom veľkosti 5 vzoriek a s dodatočnou úpravou distribúcie hodnôt chýb predikcie.

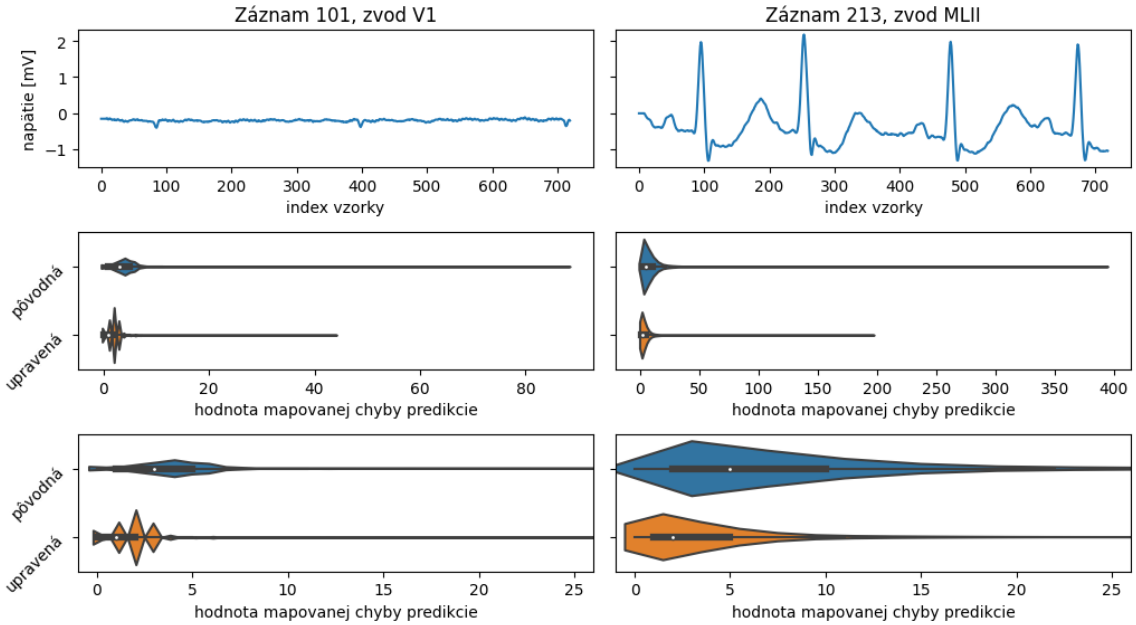
		$\mathrm{THR}_{\mathrm{static}}$	$\mathrm{THR}_{\mathrm{adaptive}}$		$\mathrm{THR}_{\mathrm{adaptive}}$ s úpravou distribúcie	
Záznam	Zvod	CF	CF	PR	CF	PR
100	MLII	2.8373	2.8415	0.1482 %	3.2301	13.8431 %
100	V5	2.8284	2.8255	-0.1029 %	3.2326	14.2902 %
101	MLII	2.8042	2.8107	0.2337 %	3.1419	12.0442 %
101	V1	2.9991	2.9964	-0.0902 %	3.4793	16.0096 %
102	V5	2.7635	2.7592	-0.1566 %	3.1242	13.0499 %
102	V2	2.5830	2.5546	-1.1007 %	2.8469	10.2150 %
103	MLII	2.7626	2.7593	-0.1219 %	3.0816	11.5462 %
103	V2	2.7486	2.7474	-0.0432 %	3.0700	11.6910 %
104	V5	2.6502	2.6416	-0.3229 %	2.9739	12.2143 %
104	V2	2.6558	2.6495	-0.2357 %	2.9868	12.4637 %
105	MLII	2.6720	2.6539	-0.6776 %	2.9293	9.6289 %
105	V1	2.7792	2.7779	-0.0451 %	3.1049	11.7186 %
106	MLII	2.5438	2.5500	0.2431 %	2.7745	9.0667 %
106	V1	2.6849	2.6986	0.5114 %	2.9276	9.0428 %
107	MLII	2.4743	2.4430	-1.2649 %	2.6723	8.0029 %
107	V1	2.5247	2.5050	-0.7809 %	2.7648	9.5081 %
108	MLII	2.6191	2.6238	0.1780 %	2.9078	11.0220 %
108	V1	2.5819	2.5854	0.1356 %	2.8749	11.3491 %
109	MLII	2.7360	2.7092	-0.9794 %	2.9757	8.7626 %
109	V1	2.8065	2.7837	-0.8123 %	3.1801	13.3110 %

Priemerná hodnota rozdielu medzi kompresnými faktormi $\mathrm{THR}_{\mathrm{static}}$ a $\mathrm{THR}_{\mathrm{adaptive}}$ je približne $-0.2034\,\%$. Najlepší výsledok (rozdiel $1.3941\,\%$) sa dosiahol na zvode V_1 záznamu 212. Najhorší výsledok s rozdielom kompresného faktoru $-4.8512\,\%$ sa dosiahol na zvode MLII záznamu 213. Priemerná dĺžka slova dosiahnutá upravenou metódou je $\mathrm{avL} = 4.4002$ bitov na vzorku. Z celkových 96 EKG signálov bola navrhovanou metódou zlepšená kompresia na 35 signáloch.

V prípade, že sa pre výpočet parametru $\mathrm{THR}_{\mathrm{adaptive}}$ využijú floating point hodnoty (hodnoty s posuvnou desatinnou čiarkou), tak sa výsledky mierne zmenia. Priemerný rozdiel je v takom prípade $-0.2297\,\%$, najlepší výsledok $2.2886\,\%$ a najhorší $-3.4886\,\%$. Kompresia sa zlepší na 28 signáloch. Najlepší výsledok takejto implementácie je síce lepší oproti celočíselnej, avšak v priemere sa jedná o horšiu metódu výpočtu parametru $\mathrm{THR}_{\mathrm{adaptive}}$.

[1]Pre úplnú tabuľku viď prílohu B.

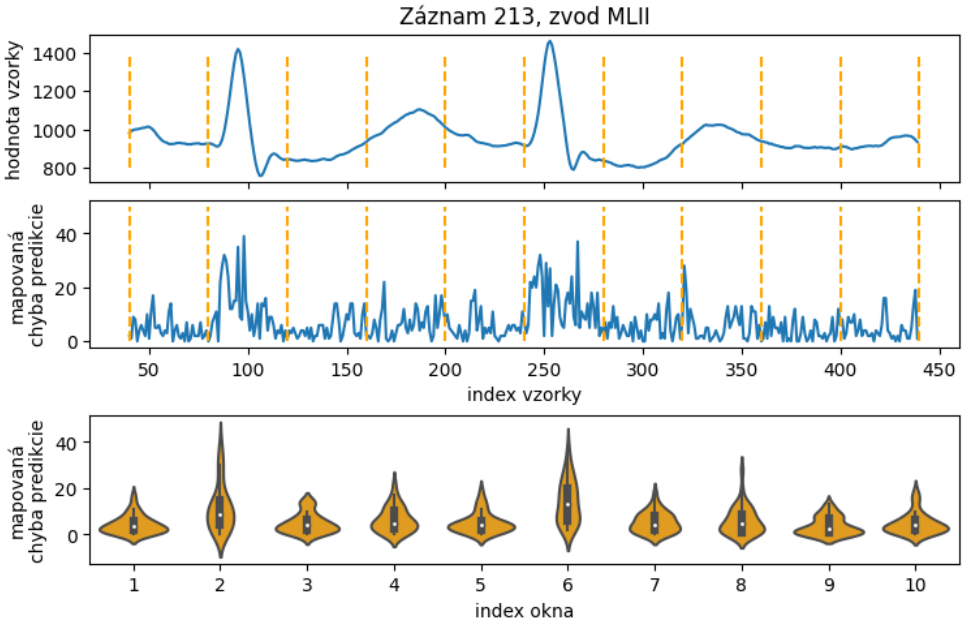
Výrazne lepšie výsledky boli dosiahnuté aplikovaním úpravy distribúcie rozloženia chýb predikcie. Priemerná hodnota rozdielu medzi kompresnými faktormi $\mathrm{THR}_{\mathrm{static}}$ a tohto prístupu je približne 10.5665 %. Najlepší výsledok (rozdiel 16.0096 %) sa dosiahol na zvode V_1 záznamu 101. Najhorší výsledok s rozdielom kompresného faktoru 5.9435 % sa dosiahol na zvode MLII záznamu 213. Priemerná dĺžka slova dosiahnutá upravenou metódou je avL = 3.9756 bitov na vzorku. Z celkových 96 EKG signálov bola navrhovanou metódou zlepšená kompresia na všetkých signáloch. Znázornenie zúženia distribúcií mapovaných chýb predikcie naprieč celým signálom je zobrazené na obrázku 8.4. Sú tu uvedené aj ukážky signálov, ktorým tieto distribúcie prináležia. Je zrejmé, že sa po úprave distribúcie hodnoty chýb predikcie výrazne posunuli k hodnote 0, čím sa zvyšuje efektivita Golomb-Riceovho kódovania. Pri signáli zvodu V_1 záznamu 101 boli generované nízke chyby predikcie aj bez uvedenej úpravy distribúcie. Avšak napriek tomu bol zisk v efektivite kompresie výrazne vyšší, než pri druhom uvedenom signáli. Na kompresiu signálu MLII zo záznamu 213 mala táto úprava distribúcie menší dopad, avšak aj zúženie tejto distribúcie malo za dôsledok takmer 6 % zlepšenie kompresie. Na obrázku 8.5 je možné vidieť distribúcie mapovaných chýb signálu MLII záznamu 213 v jednotlivých oknách o šírke 40 vzoriek. Je zrejmé, že najväčšie chyby predikcie vznikajú v okolí QRS komplexov a naopak najmenšie v okolí plochého priebehu signálu.



Obr. 8.4: Ukážky prvých dvoch sekúnd EKG signálov s najlepším (vľavo) a najhorším (vpravo) zlepšením kompresného faktoru. Druhý riadok obsahuje pôvodné a upravené distribúcie chýb predikcie vzoriek týchto signálov. Posledný riadok je priblíženým pohľadom na najnižšie hodnoty huslových grafov v predchádzajúcom riadku.

8.2.3 Odhad parametru k

Navrhnuté riešenie pre odhad parametru k pre Golomb-Riceovo kódovanie (sekcia 6.2.1) dosahovalo výraznej expanzie aj na signáloch, ktoré boli využívané pre pozorovanie vlastností EKG signálov pri návrhu algoritmu. Táto metóda generovala napr. pre signál MLII zá-



Obr. 8.5: Hodnoty mapovaných chýb predikcie a ich distribúcie v oknách 1 až 10 EKG signálu. Na EKG signáli (hore) sú zobrazené hranice jednotlivých okien, ktorým distribúcie (dole) prináležia.

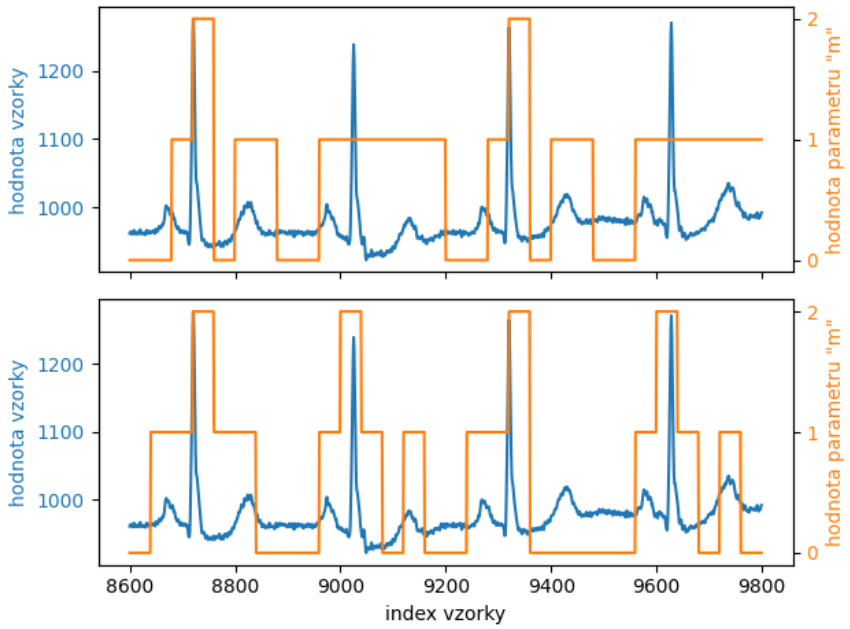
znamu 101 zhodu hodnôt parametru k približne 60.04%. Takéto percento zhody je prinízke pre vykrytie kompresie kódovaním veľkých čísel s vyššou hodnotou k. Generovaný dátový tok dosiahol kompresného pomeru CR ≈ 109, čo činí 109-násobnú expanziu dátového toku oproti nekomprimovanému signálu. Na základe týchto výsledkov neboli vykonávané žiadne dodatočné experimenty využívajúce tento prístup. Na obrázku 8.6 je zobrazená časť signálu, kde je jasne vidieť, že navrhovaná metóda nie je schopná konzistentne správne predikovať hodnotu k.

8.3 Vyhodnotenie implementácie na MCU

Vyhodnotenie spočívalo v meraní priemerného času potrebného pre vykonanie jednotlivých častí algoritmu. Čip osadený vo vývojovej doske nRF52 DK neobsahuje žiadne hardvérové riešenie umožňujúce priame profilovanie kódu (napr. pomocou čítačov inštrukcií), disponuje ale časovacím modulom TIMER. Tento časovač má podporu rôznych rozlíšení, pričom najvyššia podporovaná frekvencia časovača je 16 MHz. Takéto rozlíšenie je však aj pri 64 MHz frekvencií procesora dostatočné pre vyhodnotenie algoritmu pre účely práce. Z tohto dôvodu bol zvolený pre profilovanie implementovaného algoritmu.

Časovací modul bol inicializovaný v `main()` funkcii, pričom samotné časovanie bolo vykonávané pomocou volaní funkcie `nrfx_timer_capture()`, ktorá poskytuje jednoduché rozhranie pre zachytenie a vrátenie hodnoty časovača. Časovanie bolo vykonané na rôznych miestach pre detailnejšie profilovanie implementácie. Všetka komunikácia cez UART bola počas meraní vypnutá. Vstupný signál bol tvorený prvými 1000 vzorkami záznamu 100 databázy MIT-BIH.

Na základe meraní bolo zistené, že implementácia algoritmu s adaptívnym výpočtom parametru $\text{THR}_{\text{adaptive}}$ je približne 1.55-násobne pomalšia, než implementácia pôvodného



Obr. 8.6: Vizualizácia odhadu parametru k navrhovanou metódou (hore) a pôvodnou metódou (dole).

algoritmu. Využitie float inštrukcií pri adaptívnom výpočte parametru THR spomaľuje implementáciu na približne 4.4-násobne pomalšiu. V tabuľke 8.2 je možné vidieť porovnanie rýchlostí častí algoritmu, na ktoré má vplyv zmena v návrhu. Prvý stĺpec tabuľky zobrazuje čas trávený vo funkciách `prediction_tsai()` a `prediction_nemeth()` počas jedného volania. Druhý stĺpec obsahuje priemerný čas strávený predikciou hodnôt v rámci jedného okna signálu. Posledný stĺpec tabuľky nakoniec obsahuje priemerný čas Golomb-Riceovho kódovania chýb v rámci jedného okna. Tento čas je nepriamo ovplyvňovaný verziou algoritmu a využitím float inštrukcií, pretože závisí na presnosti predikcie. Tzn. v prípade, že sú predikcie presné, a teda chyby predikcie nízke, tak vytváranie výsledného kódu trvá kratšiu dobu. Odhad parametru k pre jednotlivé okná a zakódovanie piatich priamych hodnôt na začiatok dátového toku trvá vždy rovnako dlhý čas nezávisle od použitého algoritmu, a to 7.8125 μs a 19.1875 μs v rovnakom poradí.

Tabuľka 8.2: Doba trvania jednotlivých častí kódu na MCU. Uvedené sú aj časovania pri využití float inštrukcií.

		`prediction_*()`	Výpočet chyby predikcie v jednom okne	Kódovanie chýb v jednom okne
$\text{THR}_{\text{static}}$	-	2.6875 μs	94.53375 μs	241.8625 μs
$\text{THR}_{\text{adaptive}}$	int	4.1875 μs	147.1768 μs	236.9525 μs
	float	12.0625 μs	407.7575 μs	238.4475 μs
$\text{THR}_{\text{adaptive}}$ s úpravou distr.	int	4.1875 μs	151.6475 μs	313.5400 μs
	float	12.0625 μs	430.3475 μs	312.8075 μs

Algoritmus spracúva vzorky po oknách o 40 vzorkách. Najskôr sa vypočítajú chyby predikcie pre jednotlivé vzorky v okne, pričom sa v tomto kroku spracúva každá nová vzorka

Tabuľka 8.3: Priemerné kompresné faktory nad MIT-BIH databázou rôznych publikácií zameraných na online bezstratovú kompresiu.

Metóda predikcie	Štatistické kódovanie	CF	Referencia
Adaptívna lineárna predikcia	Joint coding-packaging scheme	2.15	[10]
Lineárna predikcia	Joint coding-packaging scheme	2.25	[11]
Adaptívny prediktor trendu	Huffmanovo kódovanie	2.43	[8]
Adaptívna lineárna predikcia	Adaptívne Golomb-Riceovo kódovanie	2.77	[36]
Adaptívna lineárna predikcia	Adaptívne Golomb-Riceovo kódovanie	2.74	Reimplementácia [36]
Adaptívna lineárna predikcia	Adaptívne Golomb-Riceovo kódovanie s úpravou distribúcie chýb predikcie	3.03	Navrhovaný algoritmus

osobitne. Následne sa vypočíta parameter k pre celé okno, čím sa začína blokovanie príjmu ďalších vzoriek. Nakoniec sa kódujú všetky hodnoty chýb predikcie v danom okne do dátového toku, čím sa naďalej blokuje príjem ďalších vzoriek. Najdlhšiu dobu (v celočíselnej implementácii) zaberá práve krok kódovania chýb predikcie. Za predpokladu, že príjem nových vzoriek a samotný algoritmus nie sú hardvérovo alebo softvérovo paralelizované, tak musí byť súčet doby výpočtu parametru k a zakódovania celého súčasného okna menší, než čas medzi prijatím poslednej vzorky súčasného okna a prijatím prvej vzorky nasledujúceho okna. To znamená, že maximálna podporovaná vzorkovacia frekvencia algoritmu je obmedzená týmito dvoma udalosťami.

Na základe meraní sa v závislosti od metódy priemerná doba kódovania chýb jedného okna pohybuje okolo 240 μs a 320 μs (zaokrúhlené nahor v rádoch desiatok). Ako bolo uvedené, výpočet parametru k trvá konštantný čas 7.8125 μs. Po súčte a zaokrúhlení týchto dôb nahor sa získava čas 250 μs a 330 μs, čo činí 4 kHz, resp. 3 kHz, ako frekvenciu spracovania jedného okna metódou bez úpravy distribúcie a s úpravou distribúcie chýb predikcie, v rovnakom poradí. Táto frekvencia je teda približnou maximálnou možnou frekvenciou prijímania nových vzoriek. Nezávisle od použitia floating point inštrukcií je táto frekvencia približne rovnaká. Na základe týchto výsledkov je preto možné usúdiť, že obe verzie navrhnutého algoritmu sú vhodné pre online kompresiu EKG signálu na testovanom mikrokontroléri.

8.4 Diskusia

Na základe výsledkov vyhodnotenia implementácie algoritmu so statickým a adaptívnym parametrom prahu THR je možné uviesť, že adaptívny prístup má potenciál konzistentne zlepšiť kompresnú výkonnosť algoritmu po dodatočných úpravách. Ako ďalší krok vo vylepšení metódy sa navrhuje náhrada alebo úprava funkcie pre presnejší odhad parametru THR. Využitím floating point operácií počas výpočtu parametru THR sa kompresná výkonnosť

líšila zanedbateľne, pričom sa doba trvania výpočtu takmer zpäťnásobila. Využitie floating point inštrukcií sa z tohto dôvodu v navrhovanej metóde neodporúča.

Veľmi dobré výsledky boli dosiahnuté dodatočnou úpravou distribúcie hodnôt chýb predikcie. Toto vylepšenie algoritmu zaistilo približne 5 až 15 percentné zlepšenie kompresie oproti pôvodnej metóde na každom jednom signáli databázy MIT-BIH. Navrhnutá metóda dosahuje priemerného kompresného faktoru 3.03. Porovnanie s inými publikáciami zameranými na online bezstratovú kompresiu je uvedené v tabuľke 8.3. Rozdiel v kompresných faktoroch medzi metódou v pôvodnom článku [36] a reimplementáciou metódy v tejto práci je pravdepodobne spôsobený chybou v pôvodnej publikácii, kde je nesprávne uvedená zdvojená podmienka pre voľbu prediktoru. Vzhľadom na to, že správnu podmienku nebolo možné dohľadať, bola metóda reimplementovaná s touto chybou. Na kompresnú výkonnosť reimplementácie mala táto úprava malý dopad (rozdiel v CF o 0.03).

Modifikácia algoritmu pomocou odhadu parametru k pre Golomb-Riceovo kódovanie bola neúspešná. Na základe výsledkov sa vyvodzuje, že prístup odhadu tohto parametru pomocou série rozhodnutí na základe štatistík o vzorkách z predošlých okien nie je vhodný. Možnou náhradou tohto prístupu by bola metóda využívajúca strojové učenie, kde by model bol trénovaný na rôznych sledoch okien EKG signálu, čím by sa mohla zýšiť presnosť predikcie parametru. V konečnom dôsledku je však potenciálna úspora získaná vypustením kódovania hodnoty parametru k do dátového toku minimálna.

Kapitola 9

Záver

Táto práca sa zaoberala úvodom do teórie za rôznymi metódami kompresie signálu elektrokardiogramu. Bol uvedený prehľad o fyziológii ľudského srdca, vďaka ktorej je možné takýto signál zachytávať a interpretovať. Uviedli sa rôzne vlastnosti EKG signálu, vrátane štruktúry a frekvenčnej kompozície a bolo naznačené aký majú tieto vlastnosti dopad na diagnostiku srdcových ochorení.

Prebrali sa objektívne prístupy k hodnoteniu kompresných algoritmov a uviedli sa rôzne metódy kompresie, ako sú Run-Length Encoding, Huffmanovo kódovanie a Golomb-Riceovo kódovanie. Bol poskytnutý prehľad niekoľkých prístupov ku kompresii EKG signálu, vrátane detailného popisu prístupu lineárnej predikcie, ktorá je využívaná v implementačnej časti práce. Ďalej sa načrtla architektúra procesorov ARM Cortex-M s ohľadom na konkrétne implementačné detaily a inštrukčné sady procesorov Cortex-M3 a Cortex-M4F. Následne bol uvedený návrh algoritmu, ktorý je vhodný pre použitie na mikrokontroléroch so spomenutými procesormi. Tento návrh bol implementovaný ako prototyp v jazyku Python a následne ako C program kompilovateľný pre mikrokontrolér nRF-52 DK s procesorom Cortex-M4F.

Navrhnutý algoritmus a jeho parametry boli vyhodnotené pomocou Python prototypu na dátovej sade MIT-BIH. Výsledný C program bol vyhodnotený aj na uvedenom mikrokontroléri a potvrdila sa možnosť využitia algoritmu pre kompresiu v reálnom čase nezávisle od použitia FPU jednotky. Výsledky riešenia boli diskutované a porovnané s publikáciami využívajúce podobné prístupy ku kompresii.

Literatúra

[1] Al Fahoum, A. S. Quality assessment of ECG compression techniques using a wavelet-based diagnostic measure. *IEEE Transactions on Information Technology in Biomedicine*. IEEE. 2006, zv. 10, č. 1, s. 182–191. DOI: 10.1109/titb.2005.855554.

[2] Alesanco, A., Olmos, S., Istepanian, R. a Garcia, J. A novel real-time multilead ECG compression and de-noising method based on the wavelet transform. In: *Computers in Cardiology, 2003.* 2003, s. 593–596. DOI: 10.1109/CIC.2003.1291225.

[3] Anjum, M. S. a Chakraborty, M. ECG data compression using turning point algorithm. *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*. 2014, zv. 2. ISSN 2349-7300.

[4] Arm Limited. *Arm Cortex-M3 Processor: Technical Reference Manual*. 2016. Dostupné z: https://developer.arm.com/documentation/100165/latest.

[5] Arm Limited. *ARMv7-M Architecture Reference Manual*. 9. vyd. 2021. Dostupné z: https://developer.arm.com/documentation/ddi0403/latest.

[6] Bousseljot, R., Kreiseler, D. a Schnabel, A. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. Walter de Gruyter, Berlin/New York Berlin, New York. 1995. DOI: 10.1515/bmte.1995.40.s1.317.

[7] Chen, C.-A., Chen, S.-L., Huang, H.-Y. a Luo, C.-H. An efficient micro control unit with a reconfigurable filter design for wireless body sensor networks (WBSNs). *Sensors*. Molecular Diversity Preservation International (MDPI). 2012, zv. 12, č. 12, s. 16211–16227. DOI: 10.3390/s121216211.

[8] Chen, S.-L. a Wang, J.-G. VLSI implementation of low-power cost-efficient lossless ECG encoder design for wireless healthcare monitoring application. *Electronics Letters*. Wiley Online Library. 2013, zv. 49, č. 2, s. 91–93. DOI: 10.1049/el.2012.3505.

[9] Cox, J., Nolle, F., Fozzard, H. a Oliver, G. AZTEC, a preprocessing program for real-time ECG rhythm analysis. *IEEE Transactions on Biomedical Engineering*. IEEE. 1968, č. 2, s. 128–129. DOI: 10.1109/TBME.1968.4502549.

[10] Deepu, C. J., Zhang, X., Heng, C. H. a Lian, Y. A 3-lead ECG-on-chip with QRS detection and lossless compression for wireless sensors. *IEEE Transactions on Circuits and Systems II: Express Briefs*. IEEE. 2016, zv. 63, č. 12, s. 1151–1155. DOI: 10.1109/TCSII.2016.2613564.

[11] Deepu, C. J., Zhang, X., Liew, W.-S., Wong, D. L. T. a Lian, Y. An ECG-on-Chip With 535 nW/Channel Integrated Lossless Data Compressor for Wireless Sensors. *IEEE Journal of Solid-State Circuits*. 2014, zv. 49, č. 11, s. 2435–2448. DOI: 10.1109/JSSC.2014.2349994.

[12] Drábek, V. *Kódování a komprese dat KKO: Studijní opora*. Brno: Fakulta informačních technologií VUT v Brně, 2008.

[13] Francis, J. *Segments and Intervals in an ECG* [online]. December 2015. Dostupné z: https://johnsonfrancis.org/professional/segments-and-intervals-in-an-ecg/.

[14] Golomb, S. Run-length encodings (Corresp.). *IEEE Transactions on Information Theory*. 1966, zv. 12, č. 3, s. 399–401. DOI: 10.1109/TIT.1966.1053907.

[15] Grider, M. H., Jessu, R. a Kabir, R. Physiology, action potential. *National Library of Medicine* [online]. Treasure Island (FL): StatPearls Publishing. Január 2022. Dostupné z: https://www.ncbi.nlm.nih.gov/books/NBK538143/.

[16] Hayden, W., Conover, M. a Bennett, W. *Compression and R-wave detection of ECG/VCG data*. NASA, júl 1972. Dostupné z: https://ntrs.nasa.gov/api/citations/19720000390/downloads/19720000390.pdf.

[17] Holter, N. J. New method for heart studies. *Science*. JSTOR. 1961, zv. 134, č. 3486, s. 1214–1220. DOI: 10.1126/science.134.3486.1214.

[18] Hrubeš, J. a Kozumplík, J. Možnosti algoritmu SPIHT při kompresi signálů EKG. *Elektrorevue*. 2007, č. 55, s. 12. ISSN 1213 - 1539.

[19] Kabali, C., Xie, X. a Higgins, C. Long-term continuous ambulatory ECG monitors and external cardiac loop recorders for cardiac arrhythmia: A health technology assessment. *Ontario health technology assessment series*. Január 2017, zv. 17, č. 1, s. 1–56. ISSN 1915-7398. PMID: 28194254.

[20] Kennedy, H. L., Bavishi, N. S. a Buckingham, T. A. Ambulatory (holter) electrocardiography signal-averaging: A current perspective. *American Heart Journal*. November 1992, zv. 124, č. 5, s. 1339–1346. DOI: 10.1016/0002-8703(92)90421-q. ISSN 0002-8703.

[21] Koski, A. Lossless ECG encoding. *Computer Methods and Programs in Biomedicine*. 1997, zv. 52, č. 1, s. 23–33. DOI: https://doi.org/10.1016/S0169-2607(96)01779-8. ISSN 0169-2607.

[22] Kwon, O., Jeong, J., Kim, H. B., Kwon, I. H., Park, S. Y. et al. Electrocardiogram sampling frequency range acceptable for heart rate variability analysis. *Healthcare Informatics Research*. Júl 2018, zv. 24, č. 3, s. 198–206. DOI: 10.4258/hir.2018.24.3.198.

[23] Lilly, L. S. *Pathophysiology of heart disease: a collaborative project of medical students and faculty*. 6. vyd. Lippincott Williams & Wilkins, 2015. ISBN 978-1-4511-9275-9.

[24] Mamaghanian, H., Khaled, N., Atienza, D. a Vandergheynst, P. Compressed Sensing for Real-Time Energy-Efficient ECG Compression on Wireless Body Sensor Nodes. *IEEE Transactions on Biomedical Engineering*. 2011, zv. 58, č. 9, s. 2456–2466. DOI: 10.1109/TBME.2011.2156795.

[25] Moody, G. B. a Mark, R. G. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*. IEEE. 2001, zv. 20, č. 3, s. 45–50. DOI: 10.1109/51.932724.

[26] Mota, H., Vasconcelos, F. a Silva, R. da. Real-time wavelet transform algorithms for the processing of continuous streams of data. In: *IEEE International Workshop on Intelligent Signal Processing, 2005.* 2005, s. 346–351. DOI: 10.1109/WISP.2005.1531683.

[27] Němcová, A. *Komprese a hodnocení kvality signálů EKG* [online]. Brno, 2021. Dizertačná práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedúci práce Vítek, M. Dostupné z: http://hdl.handle.net/11012/196772.

[28] Rajankar, S. O. a Talbar, S. N. An electrocardiogram signal compression techniques: a comprehensive review. *Analog Integrated Circuits and Signal Processing*. Springer. 2019, zv. 98, č. 1, s. 59–74. DOI: 10.1007/s10470-018-1323-1.

[29] Said, A. a Pearlman, W. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*. 1996, zv. 6, č. 3, s. 243–250. DOI: 10.1109/76.499834.

[30] Salomon, D. *Data Compression: The Complete Reference*. 4. vyd. Springer, december 2006. ISBN 1846286026.

[31] Salomon, D. *A concise introduction to data compression*. 2008. vyd. Guildford, England: Springer, december 2007. Undergraduate Topics in Computer Science. ISBN 978-1-84800-072-8.

[32] Su, L., Borov, S. a Zrenner, B. 12-lead Holter Electrocardiography. *Herzschrittmachertherapie + Elektrophysiologie*. 2013, zv. 24, č. 2, s. 92–96. DOI: 10.1007/s00399-013-0268-4.

[33] Taddei, A., Distante, G., Emdin, M., Pisani, P., Moody, G. et al. The European ST-T database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography. *European heart journal*. Oxford University Press. 1992, zv. 13, č. 9, s. 1164–1172. DOI: 10.1093/oxfordjournals.eurheartj.a060332.

[34] Tereshchenko, L. G. a Josephson, M. E. Frequency content and characteristics of ventricular conduction. *Journal of Electrocardiology*. 2015, zv. 48, č. 6, s. 933–937. DOI: 10.1016/j.jelectrocard.2015.08.034. ISSN 0022-0736.

[35] Tiwari, A. a Falk, T. H. Lossless electrocardiogram signal compression: A review of existing methods. *Biomedical Signal Processing and Control*. 2019, zv. 51, s. 338–346. DOI: 10.1016/j.bspc.2019.03.004. ISSN 1746–8094.

[36] Tsai, T.-H. a Kuo, W.-T. An Efficient ECG Lossless Compression System for Embedded Platforms With Telemedicine Applications. *IEEE Access*. 2018, zv. 6, s. 42207–42215. DOI: 10.1109/ACCESS.2018.2858857.

[37] Vale Madeiro, J. P. do, Cortez, P. C., Monteiro Filho, J. M. D. S. a Brayner, A. R. A. *Developments and applications for ECG signal processing: Modeling, segmentation, and pattern recognition*. Academic Press, 2018. ISBN 9780128140369.

[38] Večerka, A. *Komprese dat* [online]. 2008. Dostupné z: https://phoenix.inf.upol.cz/esf/ucebni/komprese.pdf.

[39] Vyas, N. *Biomedical Signal Processing*. Laxmi Publications Pvt, 2011. ISBN 9789381159040. Dostupné z: https://books.google.cz/books?id=Khj-99m8DG4C.

[40] Wachter, R., Groeschel, K., Gelbrich, G., Hamann, G. F., Kermer, P. et al. Holter-electrocardiogram-monitoring in patients with acute ischaemic stroke (Find-AFRANDOMISED): an open-label randomised controlled trial. *The Lancet Neurology*. Elsevier. 2017, zv. 16, č. 4, s. 282–290. DOI: 10.1016/S1474-4422(17)30002-9.

[41] Yiu, J. *The Definitive Guide to the ARM Cortex-M3*. 2. vyd. Elsevier, november 2009. ISBN 978-1-85617-963-8.

[42] Zigel, Y., Cohen, A. a Katz, A. The weighted diagnostic distortion (WDD) measure for ECG signal compression. *IEEE transactions on biomedical engineering*. IEEE. 2000, zv. 47, č. 11, s. 1422–1430. DOI: 10.1109/TBME.2000.880093.

Príloha A

Inštalácia a spustenie programov

A.1 Python prototyp

Zdrojové súbory sa nachádzajú v priečinku `src/python_proto`. Prerekvizitami pre spustenie skriptov sú `Python 3.9`, príslušný správca balíčkov `pip` a EKG záznamy vo formáte WFDB[1]. Požadované balíčky sa nachádzajú v súbore `requirements.txt`. Odporúčaný postup inštalácie balíčkov je nasledovný:

1. vytvorenie virtuálneho prostredia príkazom `python -m venv ./venv`,
2. aktivácia virtuálneho prostredia príkazom `source venv/bin/activate`,
3. inštalácia balíčkov príkazom `python -m pip install -r requirements.txt`.

Po inštalácii balíčkov je možné spúšťať odovzdané Python skripty `evaluation.py`, prípadne `main.py`. Oba zo skriptov akceptujú prepínač `-h` pre výpis podporovaných prepínačov.

Skript `evaluation.py` bol využívaný pre hromadné vyhodnocovanie algoritmov. Výstupom je výpis štatistík z vyhodnotenia zvoleného algoritmu vo formáte DSV, kde oddeľovačom sú tabulátory. Skript `main.py` bol využívaný pre priebežné vyhodnocovanie algoritmov počas vývoja. Výstupom sú rôzne štatistiky a informácie o spracovaní jedného signálu zvoleného záznamu.

A.2 C program

Zdrojové súbory sa nachádzajú v priečinku `src/c`. Prerekvizitou je operačný systém Linux, kompilátor GCC[2], knižnica WFDB[3] a EKG záznamy vo formáte WFDB. Z dôvodu jednoduchšej práce s WFDB záznamami sa odporúča vloženie jednotlivých záznamov do priečinka so zdrojovými súbormi programu. Program je možné skompilovať pomocou príkazu `make` a spustiť pomocou príkazu `./ecg`. Na štandardný výstup je po spustení vypísaný *celý* kódovaný signál. Spustenie programu s prepínačom `-h` sa na štandardný výstup vypíšu možnosti spustenia.

[1] Databáza MIT-BIH, na ktorej bolo riešenie vyhodnocované je dostupná na adrese https://www.physionet.org/content/mitdb/1.0.0/.

[2] Verzia využívaná počas vývoja bola 10.2.1.

[3] Pre inštaláciu viď https://archive.physionet.org/physiotools/wfdb.shtml.

Pre spustenie programu na vývojovej doske nRF52 DK je nutné nainštalovať Visual Studio Code[4] a program nRF Connect for Desktop[5], pomocou ktorého sa jednoducho môže nainštalovať nRF Connect SDK (počas vývoja bola využívaná verzia v2.3.0). Najjednoduchší spôsob ako od tohto bodu pokračovať je nasledovný:

- pomocou rozšírenia pre VSCode nainštalovaného počas kroku inštalácie SDK vytvoriť projekt na základe šablóny `hello_world`,
- úprava súboru `CMakeLists.txt` aby obsahoval nasledovné riadky

```
target_sources(app PRIVATE <cesta>/src/c/main.c)
target_sources(app PRIVATE <cesta>/src/c/decider.c)
target_sources(app PRIVATE <cesta>/src/c/prediction.c)
target_sources(app PRIVATE <cesta>/src/c/rice.c)
target_sources(app PRIVATE <cesta>/src/c/samples.c)
```

Na uvedených riadkoch je `<cesta>` relatívnou cestou od vygenerovaného projektu k zdrojovým súborom implementácie. Následnú kompiláciu a načítanie binárneho súboru do vývojovej dosky je možné vykonať pomocou nainštalovaného VSCode rozšírenia. Program po spustení vypíše kódovaný signál cez sériové rozhranie UART.

[4] Pre inštaláciu viď https://code.visualstudio.com/.

[5] Pre inštaláciu viď https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop.

Príloha B

Úplné tabuľky vyhodnotenia algoritmov

Tabuľka B.1: Porovnanie pôvodnej kompresnej metódy s navrhovanou kompresnou metódou s adaptívnym prispôsobením hodnoty $THR_{adaptive}$. Stĺpec PR vyjadruje percentuálny rozdiel CF oproti hodnote pre THR_{static}. Uvedené sú výsledky vyhodnotenia nad všetkými záznamami MIT-BIH databázy.

		THR_{static}	$THR_{adaptive}$		$THR_{adaptive}$ s úpravou distribúcie	
Záznam	Zvod	CF	CF	PR	CF	PR
100	MLII	2.8373	2.8415	0.1482 %	3.2301	13.8431 %
100	V5	2.8284	2.8255	-0.1029 %	3.2326	14.2902 %
101	MLII	2.8042	2.8107	0.2337 %	3.1419	12.0442 %
101	V1	2.9991	2.9964	-0.0902 %	3.4793	16.0096 %
102	V5	2.7635	2.7592	-0.1566 %	3.1242	13.0499 %
102	V2	2.5830	2.5546	-1.1007 %	2.8469	10.2150 %
103	MLII	2.7626	2.7593	-0.1219 %	3.0816	11.5462 %
103	V2	2.7486	2.7474	-0.0432 %	3.0700	11.6910 %
104	V5	2.6502	2.6416	-0.3229 %	2.9739	12.2143 %
104	V2	2.6558	2.6495	-0.2357 %	2.9868	12.4637 %
105	MLII	2.6720	2.6539	-0.6776 %	2.9293	9.6289 %
105	V1	2.7792	2.7779	-0.0451 %	3.1049	11.7186 %
106	MLII	2.5438	2.5500	0.2431 %	2.7745	9.0667 %
106	V1	2.6849	2.6986	0.5114 %	2.9276	9.0428 %
107	MLII	2.4743	2.4430	-1.2649 %	2.6723	8.0029 %
107	V1	2.5247	2.5050	-0.7809 %	2.7648	9.5081 %
108	MLII	2.6191	2.6238	0.1780 %	2.9078	11.0220 %
108	V1	2.5819	2.5854	0.1356 %	2.8749	11.3491 %
109	MLII	2.7360	2.7092	-0.9794 %	2.9757	8.7626 %
109	V1	2.8065	2.7837	-0.8123 %	3.1801	13.3110 %
111	MLII	2.7296	2.7306	0.0375 %	2.9630	8.5486 %
111	V1	2.7880	2.7801	-0.2811 %	3.0141	8.1117 %
112	MLII	2.9513	2.9358	-0.5275 %	3.3074	12.0632 %
112	V1	3.0013	2.9818	-0.6519 %	3.4503	14.9577 %

		THR_{static}	$THR_{adaptive}$		$THR_{adaptive}$ s úpravou distribúcie	
Záznam	Zvod	CF	CF	PR	CF	PR
113	MLII	2.6121	2.6087	-0.1279 %	2.8055	7.4060 %
113	V1	2.8951	2.8901	-0.1728 %	3.1683	9.4366 %
114	V5	2.7663	2.7944	1.0146 %	2.9788	7.6818 %
114	MLII	2.6613	2.6745	0.4972 %	2.9029	9.0784 %
115	MLII	2.8958	2.8975	0.0590 %	3.2455	12.0760 %
115	V1	3.0753	3.0683	-0.2270 %	3.5363	14.9900 %
116	MLII	2.5909	2.5596	-1.2075 %	2.8071	8.3456 %
116	V1	2.7964	2.8075	0.3971 %	3.0027	7.3800 %
117	MLII	2.8216	2.8047	-0.5993 %	3.1350	11.1061 %
117	V2	2.7818	2.7686	-0.4731 %	3.0296	8.9090 %
118	MLII	2.4979	2.4835	-0.5742 %	2.7232	9.0221 %
118	V1	2.4920	2.4773	-0.5917 %	2.7074	8.6428 %
119	MLII	2.6610	2.6278	-1.2509 %	2.9281	10.0366 %
119	V1	2.8044	2.7901	-0.5097 %	3.0394	8.3810 %
121	MLII	3.0715	3.0577	-0.4478 %	3.4876	13.5456 %
121	V1	2.8510	2.8474	-0.1239 %	3.2236	13.0695 %
122	MLII	2.6742	2.6845	0.3844 %	2.9163	9.0535 %
122	V1	1.9803	2.0029	1.1399 %	2.2482	13.5277 %
123	MLII	2.7736	2.7618	-0.4241 %	3.0962	11.6314 %
123	V5	2.8379	2.8415	0.1282 %	3.0713	8.2272 %
124	MLII	3.0040	2.9976	-0.2127 %	3.3667	12.0743 %
124	V4	2.9085	2.8985	-0.3452 %	3.3251	14.3217 %
200	MLII	2.5863	2.5873	0.0407 %	2.8534	10.3269 %
200	V1	2.7951	2.8037	0.3041 %	3.1414	12.3890 %
201	MLII	3.0766	3.0705	-0.1978 %	3.4944	13.5802 %
201	V1	2.9226	2.9180	-0.1560 %	3.3709	15.3397 %
202	MLII	2.7859	2.8017	0.5684 %	2.9977	7.6043 %
202	V1	2.8324	2.8401	0.2701 %	3.0689	8.3483 %
203	MLII	2.4028	2.4257	0.9500 %	2.6358	9.6954 %
203	V1	2.4575	2.4646	0.2871 %	2.7059	10.1060 %
205	MLII	3.1601	3.1564	-0.1155 %	3.5017	10.8090 %
205	V1	3.1198	3.1096	-0.3255 %	3.5459	13.6574 %
207	MLII	2.7499	2.7477	-0.0812 %	3.0320	10.2582 %
207	V1	2.7282	2.7469	0.6855 %	3.0412	11.4727 %
208	MLII	2.5417	2.5397	-0.0775 %	2.7705	9.0007 %
208	V1	2.6977	2.6888	-0.3315 %	2.9489	9.3116 %
209	MLII	2.5238	2.5424	0.7345 %	2.7693	9.7260 %
209	V1	2.6752	2.6775	0.0854 %	2.9426	9.9932 %
210	MLII	2.7849	2.7819	-0.1047 %	3.0988	11.2744 %
210	V1	2.8478	2.8305	-0.6082 %	3.1559	10.8198 %
212	MLII	2.3895	2.3835	-0.2496 %	2.6257	9.8849 %
212	V1	2.4628	2.4971	1.3941 %	2.7324	10.9481 %
213	MLII	2.5659	2.4414	-4.8512 %	2.7184	5.9435 %
213	V1	2.6097	2.5549	-2.0998 %	2.7832	6.6474 %
214	MLII	2.6205	2.6253	0.1850 %	2.8403	8.3875 %

		THR_{static}	$THR_{adaptive}$		$THR_{adaptive}$ s úpravou distribúcie	
Záznam	Zvod	CF	CF	PR	CF	PR
214	V1	2.5768	2.5883	0.4452 %	2.8140	9.2058 %
215	MLII	2.4627	2.4639	0.0470 %	2.6943	9.4042 %
215	V1	2.6397	2.6647	0.9493 %	2.8790	9.0677 %
217	MLII	2.5207	2.5089	-0.4668 %	2.7319	8.3793 %
217	V1	2.9093	2.8856	-0.8122 %	3.1807	9.3286 %
219	MLII	2.8668	2.8418	-0.8713 %	3.1930	11.3791 %
219	V1	2.8709	2.8567	-0.4928 %	3.2222	12.2391 %
220	MLII	2.8146	2.8088	-0.2045 %	3.1410	11.5990 %
220	V1	2.9484	2.9416	-0.2314 %	3.4107	15.6774 %
221	MLII	2.7503	2.7684	0.6604 %	2.9556	7.4648 %
221	V1	2.7617	2.7769	0.5488 %	2.9929	8.3712 %
222	MLII	2.7489	2.7632	0.5210 %	2.9787	8.3603 %
222	V1	2.8007	2.8080	0.2631 %	3.0128	7.5743 %
223	MLII	2.8985	2.8809	-0.6050 %	3.2184	11.0385 %
223	V1	2.9424	2.9297	-0.4320 %	3.3448	13.6755 %
228	MLII	2.6590	2.6718	0.4838 %	2.8843	8.4755 %
228	V1	2.6034	2.6130	0.3679 %	2.8279	8.6228 %
230	MLII	2.8378	2.8185	-0.6783 %	3.1446	10.8125 %
230	V1	2.8348	2.8047	-1.0633 %	3.1571	11.3674 %
231	MLII	2.9466	2.9438	-0.0931 %	3.2710	11.0104 %
231	V1	3.0822	3.0689	-0.4320 %	3.5152	14.0469 %
232	MLII	2.9610	2.9490	-0.4059 %	3.3126	11.8726 %
232	V1	2.8481	2.8344	-0.4826 %	3.1960	12.2122 %
233	MLII	2.7013	2.6459	-2.0507 %	2.9234	8.2207 %
233	V1	2.6904	2.6605	-1.1093 %	2.9094	8.1410 %
234	MLII	2.8638	2.8565	-0.2562 %	3.1849	11.2120 %
234	V1	2.7491	2.7749	0.9406 %	3.0157	9.7004 %