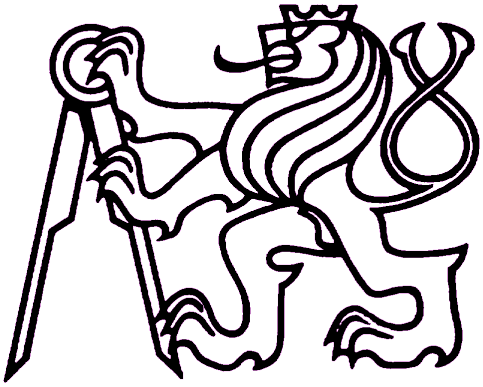
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

KATEDRA ŘÍDICÍ TECHNIKY



Bc. Milan Cejnar

Zpracování biomedicínských signálů na platformě Android

Biomedical Signal Processing on the Android Platform

Diplomová práce

Praha 2015

Autor práce:	Bc. Milan Cejnar
Studijní program:	Otevřená informatika
Studijní obor:	Počítačové inženýrství
Vedoucí práce:	Ing. Václav Gerla, Ph.D.
Pracoviště vedoucího práce:	Katedra kybernetiky

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Milan Cejnar**

Studijní program: Otevřená informatika
Obor: Počítačové inženýrství

Název tématu: **Zpracování biomedicínských signálů na platformě Android**

Pokyny pro vypracování:

Cílem práce je ověřit možnosti zpracování biomedicínských záznamů na zařízení s operačním systémem Android. Hlavní motivací je snížení objemu dat při snímání dlouhodobých signálů. Navržený postup bude experimentálně ověřen sestavením aplikace, která bude sloužit pro odhad klinicky významných parametrů a usnadní zpracování dlouhodobých záznamů.
1. Definice protokolu a sestavení aplikace, která umožní emulovat měřící zařízení a zajistí bezdrátový přenos dat z PC do zařízení s OS Android.
2. Segmentace signálu – navržený postup by měl umožnovat použití různých velikostí segmentů pro různé typy signálů.
3. Extrakce klinicky významných parametrů z různorodých biomedicínských signálů (typicky např. EKG, EEG, EOG, EMG).
4. Výběr a implementace vhodných vizualizací (např. zobrazení měřených signálů a/nebo zobrazení vypočtených trendů).
5. Ověření možností aplikace nad reálnými klinickými daty. Potřebná data dodá studentovi vedoucí práce.

Seznam odborné literatury:

[1] Jerome J.F. DiMarzio, Android a programmer´s guide, McGraw-Hill Osborne Media, 2008, online: http://www.e-reading.ws/bookreader.php/142063/Android_-_a_programmers_guide.pdf
[2] Václav Gerla, Automated Analysis of Long-Term EEG Signals. Ph.D. thesis, CTU-FEE, Prague, 2012, online: http://bio.felk.cvut.cz/psglab/disertace/disertace-2012-02-29.pdf
[3] Lopes da Silva Fernando, Niedermeyer Ernst, Electroencephalography – Basic principles, clinical applications and related field. Philadelphia, 2005.

Vedoucí: Ing. Václav Gerla, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 6. 2014

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil výhradně uvedené citované prameny, literaturu a další odborné zdroje. Současně dávám svolení k tomu, aby má diplomová práce byla zveřejněna a volně používána ke studijním účelům.

V Praze dne 5. 1. 2015

_____

podpis

Abstrakt

Tato diplomová práce pojednává o návrhu a realizaci aplikace pro operační systém Android, která je určena ke zpracování, záznamu a vizualizaci vybraných biomedicínských dat v reálném čase. Část pojednávající o zpracování dat sestává zejména z návrhu vhodných segmentačních a filtračních postupů, výběru algoritmů určených k výpočtu významných parametrů vybraných signálů a popisuje také způsoby záznamu a vizualizace získaných výsledků. Součástí práce je také implementace aplikace emulující funkci biomedicínského zařízení, jež slouží jako zdroj biomedicínských dat, a návrh komunikačního protokolu pro bezdrátový přenos dat.

Abstract

This diploma thesis describes an implementation and a design of an application for Android operating system which is capable of realtime processing and vizualization of selected biomedical signals. Parts of this thesis also deal with a realization of a separate application posing as an emulated biomedical device or sensor and describe a design of wireless communication protocol which is used by both applications to exchange data. A significant portion of this thesis also discusses a selection and a relalization of suitable signal segmentation and filtration methods as well as other means of signal processing and extraction of relevant data from selected biomedical signals.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Předmětem této práce je návrh a implementace mobilní aplikace pro systém Android, která bude schopná zpracovávat vybrané biomedicínské signály v reálném čase. Účelem práce je demonstrovat možnosti platformy Android v analýze biomedicínských signálů, jejichž zpracování zřejmě bude v blízké budoucnosti běžnou součástí chytrých telefonů a jiných asistivních, sportovních a mobilních zařízení nebo nositelné elektroniky.

Součástí práce je kromě toho také realizace aplikace, která slouží jako zdroj biomedicínských signálů a která umožňuje tyto signály odesílat v reálném čase pomocí bezdrátového spojení. S tím souvisí také navržení vhodného komunikačního protokolu, který umožní oběma aplikacím spárování, výměnu metadat a následné zasílání požadovaných biomedicínských dat.

Součástí práce je dále realizace univerzálního postupu, jenž umožňuje nezávisle pro každý signál nastavit metodu zpracování, způsob segmentace, podvzorkování a podle typu signálu případně i číslicovou filtraci signálu. Dalším krokem zpracování je extrakce významných parametrů vybraných signálů, jejich uložení pro pozdější analýzu. Navržené řešení musí být také během měření schopné průběžně aktualizovat a zobrazovat uživateli názornou formovou získané parametry, čímž se rozumí zobrazení nejen formou číselného výstupu, ale také vhodný způsob grafické vizualizace měřených průběhů. Navržená aplikace je nakonec demonstrována za použití reálných klinických dat z vybraných klinických studií.

Konkrétními metodami, které byly v rámci práce implementovány a které slouží k demonstraci možností aplikace na reálných biomedicínských signálech, jsou výpočet srdeční frekvence na základě vstupního EKG signálu a automatická klasifikace spánku na základě jednoho EEG a jednoho EOG kanálu. Aplikace dále umožňuje extrakci dalších statisticky zajímavých parametrů signálů, jako je střední a efektivní hodnota signálu, výpočet směrodatné odchylky, sledování minima a maxima měřených hodnot nebo analýzu výkonů ve zvolených frekvenčních pásmech signálu. Způsob zpracování signálů je navíc navržen modulárně tak, aby bylo možné snadno implementovat nové funkce nebo rozšiřovat stávající metody dle konkrétních požadavků daných signálů, čehož je dosaženo pomocí možnosti rozšíření předem definovaných abstraktních tříd a rozhraní.

Výstupem práce je kromě výše popsané aplikace pro operační systém Android také serverová aplikace pro PC v jazyku Java s vlastním grafickým uživatelským rozhraním, která umožňuje pomocí vlastního navrženého komunikačního protokolu odesílat vybraná klinická data ze zdrojových souborů ve formátu EDF [7]. V práci je dále využito několik skriptů pro prostředí MATLAB, které realizují popsané metody analýzy signálů a které slouží k učení a vyhodnocování výsledků klasifikátoru stádií spánku, který byl v rámci práce také navržen.

2 Motivace práce

Zpracování biologických signálů bylo dlouhou dobu výsadou pouze profesionálních medicínských a výzkumných pracovišť nebo univerzit. Profesionální medicínská elektronická zařízení bývají často velice nákladná a jejich obsluha složitá, což znemožňovalo jejich širší rozšíření do domácností mezi laickou veřejnost. V posledních letech ale dochází k prudkému vývoji a miniaturizaci elektronických zařízení a díky technologii MEMS je nyní možné realizovat celou řadu miniaturních elektromechanických součástek a senzorů v mikroskopickém provedení. Spolu s tím se vyvíjejí a miniaturizují také vestavěné systémy, což umožňuje redukovat náklady a rozměry spousty jednoúčelových zařízení, mezi nimiž lze nalézt i spoustu biomedicínských systémů. Díky zmíněným faktorům je možné postupně přibližovat tato specializovaná zařízení i oblasti spotřební elektroniky s možným využitím mezi laickými uživateli přímo v jejich domácnostech. Výsledkem jsou rozličná zařízení z oblasti asistivních technologií, nositelné elektroniky, ale i běžné spotřební elektroniky, jež kombinují a integrují poznatky z oboru elektrotechniky, biomedicínské techniky, zpracování digitální informace, obrazu, biomedicínských signálů a mnoha dalších.

V domácnostech tak dnes lze nalézt mnoho vybavení, jež bylo dříve dostupné pouze ve zdravotnických zařízeních, jako jsou například domácí inhalátory, manžetové tlakoměry, glukometry, oxymetry a další vybavení. Metody inspirované zdravotnickou technikou a zpracováním biomedicínských signálů se také postupně přesouvají do oblasti běžné spotřební elektroniky, jako jsou rozličné sportovní a rehabilitační doplňky, a do oblasti nositelné elektroniky a chytrých telefonů.

Budoucí nositelná zařízení nejspíše budou například běžně měřit srdeční frekvenci. Měření srdeční frekvence je již standardní funkce snad všech fitness trackerů, ale postupně se rozšiřuje i do oblasti chytrých telefonů. Příkladem chytrého telefonu vybaveným touto funkcí je Samsung S5, který využívá optického měření formou fotopletysmografie (zkráceně PPG). Ještě dokonalejší zařízení, které se v době psaní této práce nachází teprve ve stavu vývoje, představuje Samsung Simband, který kromě PPG nabízí také sadu elektrod a senzorů, jež slouží například k bioimpedanční analýze, či měření vodivosti kůže (zkráceně GSR) a jež nabídne možnost kontinuálně měřit srdeční tlak a frekvenci, EKG, saturaci kyslíkem nebo tělesnou teplotu. To vše samozřejmě s možností propojení

s ostatními výrobky této společnosti, hlavně tedy tablety a chytrými telefony, které budou moci tyto údaje dále vyhodnocovat.

Příkladem dalších široce rozšířených senzorů ve spotřební elektronice jsou MEMS gyroskopy a akcelerometry, které slouží k měření úhlové rychlosti a lineárního zrychlení. Ty jsou často používány k vyhodnocování pohybu a slouží tak v aplikacích jako jsou krokoměry a nebo v zařízeních nazývaných jako fitness tracker nebo také activity tracker jako například Fitbit Flex, Nike+ FuelBand nebo Samsung Gear Fit. Uvedené senzory lze nalézt také v profesionálnějších systémech, které slouží ke snímaní pohybu a lze je využít například jako zpětnou vazbu při rehabilitačních cvičeních, kde příkladem mohou být řešení od společností Xsens nebo Animazoo.

Do oblasti spotřební elektroniky se dokonce dostávají i poměrně složité metody měření a zpracování elektroencefalogramu (zkráceně EEG). EEG je využíváno v metodách jako EEG Biofeedback nebo v aplikacích typu human-machine interface (zkráceně HMI), kde mohou být příkladem produkty firmy Emotiv s názvy EPOC a Insight.

Z rozvoje těchto zařízení je tedy zřejmé, že do budoucna bude mnoho běžně dostupné elektroniky vybaveno sadou biometrických senzorů, které bude možné propojit s chytrými telefony a televizory nebo tablety. Z toho důvodu lze do budoucna pro tato zařízení také očekávat rozvoj mnoha softwarových aplikací, které budou schopny vyhodnocovat tato data v reálném čase a dále je analyzovat nebo uchovávat. To je ostatně tématem této diplomové práce, která si klade za cíl vyhodnotit a vyzkoušet možnosti současných zařízení pro zpracování biomedicínských signálů na platformě Android, jelikož tyto metody se v blízké době stanou novým standardem a běžnou součástí každodenního používání zařízení jako jsou chytré telefony nebo nové nositelné elektroniky.

3 Cíl práce

Cílem práce je navrhnout a implementovat aplikaci pro operační systém Android, jež bude schopna v reálném čase zpracovávat vybrané biomedicínské signály a jež také bude umožňovat vyhodnocovat v těchto signálech zvolené klinicky významné parametry. Cílem aplikace je také pomocí extrakce významných parametrů omezit objem vstupních dat, což usnadní zpracování a ukládání dlouhodobých biomedicínských záznamů.

Ke splnění stanovených cílů práce je nutné nejprve navrhnout a implementovat aplikaci pro PC schopnou emulovat funkci biomedicínského zařízení, která zajistí bezdrátový přenos zdrojových dat mezi PC a cílovou aplikací na zařízení s OS Android. Součástí tohoto kroku je také definice a implementace vlastní aplikační protokolové sady, který zajistí robustní a spolehlivý způsob párovaní zařízení a přenos zdrojových dat.

Dalším bodem práce je samotné zpracování biomedicínských signálů, s čímž souvisí zajištění vhodné segmentace signálů, přičemž aplikace musí v jednom měření umožňovat zpracování signálů s různou vzorkovací frekvencí a různě dlouhými segmenty. Dále je nutné navrhnout a implementovat vhodné metody extrakce klinicky významných parametrů z různorodých biomedicínských signálů jako jsou například EKG, EEG, EOG nebo EMG.

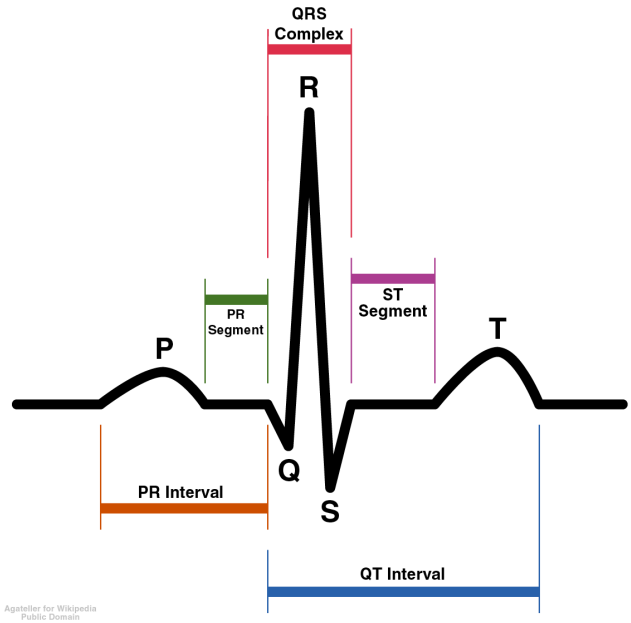
Součástí práce je i výběr a implementace vhodných metod vizualizace dat a zobrazení základních údajů o zpracovávaných signálech a extrahovaných parametrech v reálném čase. Posledním bodem práce je také ověření aplikace a navržených metod nad reálnými klinickými daty.

4 Úvod do použitých biomedicínských signálů

4.1 Elektrokardiogram

Základní funkcí lidského srdce je uvádět do pohybu krev, což se děje díky rytmickému střídání kontrakcí a relaxací srdečního svalu, jehož činnost je řízena pomocí bioelektrických signálů vznikajících v sinoatriálním uzlu. [1]

Elektrokardiogram (zkráceně EKG) je diagnostická metoda, která zaznamenává časový průběh elektrických potenciálů, které vznikají při srdeční činnosti. EKG představuje neinvazivní a ekonomickou metodu jak odhalovat a vyhodnocovat různé patologické stavy srdečního rytmu jako například arytmie, předčasné stahy, raménkové blokády, fibrilace a další. [2]



Obrázek 1: Ideální EKG záznam jednoho srdeční cyklu. Převzato z [27]

Ideální průběh EKG záznamu jednoho srdečního cyklu je znázorněn na obrázku č. 1. Každý z uvedených segmentů a intervalů má svůj fyziologický význam a pomocí jejich analýzy pak lze odhalovat různé patologické stavy. Tato práce se ale v oblasti zpracování EKG signálu zaměřuje pouze na vyhodnocování srdeční frekvence v reálném čase, k čemuž je obzvláště výhodný tzv. QRS komplex, který je způsoben depolarizací srdečních komor a v časové oblasti se projevuje jako výrazný trojúhelníkový kmit s dobou trvání přibližně 50 až 110 ms. [2] Metoda výpočtu tepové frekvence využívající algoritmu založeného na filtraci signálu a detekci QRS komplexů je popsána níže v kapitole č. 5.

4.2 Elektroencefalogram

Elektroencefalogram (zkráceně EEG) je neinvazivní diagnostická metoda založená na měření elektrické aktivity centrálního nervového systému pomocí elektrod umístěných na povrchu skalpu. Měřené elektrické pole je indukováno excitačními synaptickými proudy, které vznikají při excitaci neuronů. Touto metodou samozřejmě není možné zachytit přímo akční potenciály jednotlivých neuronů a měřené elektrické pole na povrchu skalpu reprezentuje proto pouze sumu mnoha superponovaných elektrických potenciálů jednotlivých neuronů. Výsledné elektrické pole je navíc tlumeno mozkovými obaly a lebkou a výsledný potenciál na povrchu skalpu dosahuje řádově typicky pouze desítek mikrovoltů a je proto nutné ho dále zesilovat. [3]

Za počátek EEG záznamu lze považovat pokusy Carla Matteucciho a Emila Du Bois-Reymonda ve čtyřicátých letech devatenáctého století, kteří pomocí galvanometrů poprvé zaznamenali změnu elektrického potenciálu spojenou s nervovou aktivitou při svalových kontrakcích. Zřejmě první opravdové měření EEG pak roku 1875 provedl anglický vědec Richard Caton, který za pomoci galvanometru a dvou elektrod zaznamenal lidskou mozkovou aktivitu ve formě elektrického signálu. [20]

Dnes je EEG využíváno v širokém spektru klinických aplikací nebo dokonce i v oblasti spotřební elektroniky, kde nachází využití v zařízeních typu *brain-machine interface* (zkráceně BMI). V klinické praxi slouží k diagnostice a výzkumu v oblasti neurologie a v dalších souvisejících oborech, kde se nejčastěji setkáváme s pojmenováním a umístěním elektrod podle systému 10-20, který standardně sestává z devatenácti aktivních a dvou referenčních elektrod. [3]

4.2.1 Signál EEG

Vzhledem k povaze signálu EEG, který je projevem sumy různých potenciálů mnoha neuronů, neposkytuje tato diagnostická metoda prostorově detailní obraz elektrické aktivity mozku. Na druhou stranu má toto měření velice dobré časové rozlišení a pomocí EEG lze proto sledovat různé významné tranzientní děje, které jsou přímým projevem mozkové aktivity.

V časové oblasti je možné pozorovat projevy evokovaných potenciálů, které představují bioelektrickou aktivitu při zpracování a odpovědi mozku na rozličné vnější podněty. V EEG lze tedy na vybraných elektrodách po daném vizuálním, motorickém,

sluchovém nebo obecně jakémkoliv senzorickém podnětu pozorovat bioelektrickou odezvu s odhadnutelným časovým průběhem. Problémem evokovaných potenciálů je jejich nízká amplituda oproti běžné EEG aktivitě a pro jejich detekci a zpracování je proto potřeba využít speciálních filtračních metod a průměrování signálu.

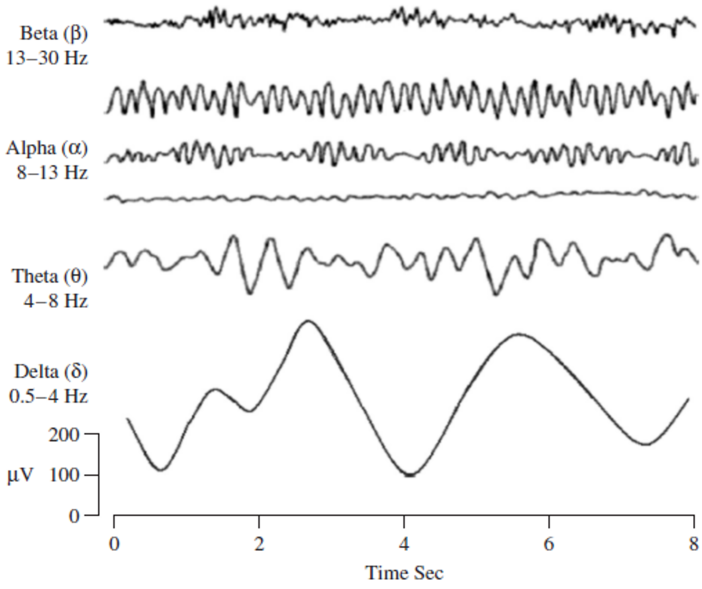
Základním problémem EEG v časové oblasti je také odlišení artefaktů neboli rušení od opravdové mozkové aktivity. Hlavními artefakty jsou artefakty z přístroje a prostředí, jako jsou například artefakty elektrostatické a napájecí sítě, elektrodové artefakty a podobně. Významné artefakty jsou také způsobeny mechanickou a fyziologickou činností pacienta, kde můžeme sledovat třeba oční artefakt, pulzový artefakt, artefakt z pocení a další.

V EEG lze sledovat samozřejmě i mnoho dalších významných fyziologických i patologických tranzientů a grafoelementů. Například při studiu spánkového EEG lze pozorovat pomalé výrazné vlny s vysokou amplitudou nazývané K komplexy nebo krátké nápadné úseky signálu s frekvencí okolo 9 až 14 Hz označované jako spánková vřetena. Mezi patologické průběhy naopak patří děje jako třeba výrazné hroty v signálu, případně hrot zakončený pomalou vlnou (spike waves) nebo několik hrotů zakončených vlnou (polyspike and waves). [22]

4.2.2 Rytmy EEG

Zásadním objevem v oblasti EEG záznamu bylo objevení přítomnosti EEG rytmů, které poskytují informaci o stavu mozkové aktivity pacienty a s jejichž pomocí lze také odhalit řadu patologických neurologických projevů.

Existuje pět hlavních rytmů, které je možné v EEG pozorovat. Nejpomalejší vlny delta spadají do frekvenčního pásma mezi 0,5 a 4 Hz a jsou spojovány zejména se stavem hlubokého spánku. Theta vlny reprezentující pásmo mezi 4 a 8 Hz se objevují zejména při usínání případně ale také při hluboké meditaci, různých kreativních nebo intuitivních činnostech. Tyto vlny jsou také běžně přítomné u dětí ale během dospívání postupně mizí. Zvýšená přítomnost vln theta u dospělého člověka může být projevem různých patologických stavů.



Obrázek 2: Rytmy EEG. Převzato z [23].

Následují vlny alfa v pásmu přibližně 8 až 13 Hz, které jsou typickým projevem prosté bdělosti a relaxovaného stavu bez většího mentálního úsilí nebo soustředění. Tyto vlny jsou také spojovány se zrakem, jelikož zavření očí se typicky na některých elektrodách projeví objevením alfa vln, zatímco při otevření očí, stejně jako při intenzivním soustředění nebo intenzivní mentální činnosti, úzkosti a podobně dochází obvykle k zeslabení nebo úplnému vymizení těchto vln. Beta vlny pak odpovídají elektrické aktivitě v pásmu přibližně mezi 13 a 30 Hz a u dospělého člověka jsou v bdělém stavu běžně přítomny. Jsou spojovány s aktivním přemýšlením, koncentrací a řešením problémů, ale mohou odrážet také některé psychologické stavy jako stres, panika a podobně. [23]

Zmíněné mozkové rytmy jsou ilustrovány na obrázku č. 2 a jedná se o hlavní čtyři rytmy, které jsou u člověka běžně přítomné. Páté a poslední pásmo připadá na frekvence větší než přibližně 30 Hz souhrnně označované jako vlny gama. Horní hranice pásma je přibližně 45 Hz, ale teoreticky mohou být významné i vyšší frekvence a například u zvířat byla zaznamenána aktivita i v pásmu mezi 200 až 300 Hz, nicméně takto vysokým frekvencím již není přikládán žádný větší klinický význam. Gama vlny se vyskytují u člověka poměrně vzácně, i když mají i svůj fyziologický význam a při určitých převážně koordinačních činnostech je lze lokálně pozorovat, jejich zvýšený výskyt může sloužit jako ukazatel při diagnostice některých onemocnění. [23]

4.3 Elektrookulogram

Ve dvacátých letech minulého století bylo objeveno, že přiložením elektrod na kůži v oblasti okolo očí je možné zaznamenat elektrickou aktivitu, která koreluje s pohybem oči. Později bylo zjištěno, že tato napětí ve skutečnosti sledují potenciál mezi rohovkou a sítnicí, který v klidovém stavu dosahuje asi 10 až 30 mV. Při rotaci oka tak dochází k rotaci vektoru tohoto elektrického pole a nepřímo tak dochází i ke změně potenciálu v okolních tkáních, díky čemuž je možné z těchto potenciálů vyčíst i samotný pohyb oka. Pomocí umístění elektrod na víčka a poblíž očních koutků je tak možné sledovat zvlášť vertikální a horizontální pohyby očí. [21]

Elektrookulogram (zkráceně EOG) je významný při klasifikaci spánkových stádií, kde je možné sledovat dva snadno rozlišitelné průběhy. Prvním průběhem jsou pomalé oční pohyby (zkráceně SEM), které jsou typickým projevem usínání a prvního stádia spánku. Druhým výrazným průběhem jsou rychlé oční pohyby (zkráceně REM), které je sice možné pozorovat i během bdění, ale které jsou hlavně klíčovým ukazatelem fáze spánku REM. [19]

4.4 Elektromyogram

Elektromyogram (zkráceně EMG) je diagnostická metoda sloužící ke sledování elektrických potenciálů vznikajících při svalové aktivitě a hraje tak významnou roli při diagnostice celé řady poruch nervosvalové soustavy. Používají se klasické povrchové elektrody, které se přikládají na kůži v oblasti sledovaného svalu, nebo podkožní jehlové elektrody. Signál získaný povrchovými elektrodami představuje sumu akčních potenciálů motorických jednotek s frekvenčním pásmem v rozsahu od několika desítek až do několik stovek Hz. Podpovrchové jehlové elektrody jsou invazivní metodou měření EMG, ale umožňují sledovat velice přesně potenciály jednotlivých motorických jednotek a navíc s menším šumem a s frekvenčním pásmem dosahujícím až ke 2 kHz. [3]

Při využití EMG během polysomnografie se často sleduje svalová aktivita svalstva brady. Sledování svalové aktivity během spánku může být významné pro rozlišení přechodu mezi stádiem bdělosti a stádiem N1, kdy dochází mírnému útlumu svalové aktivity, pro stádia spánku N2 a N3 však není EMG významné, i když i zde může být sledován další útlum. Zásadní je signál EMG pro určení fáze spánku REM, kdy dochází k ochabnutí svalstva a k výrazné celkové redukci sledované svalové aktivity. [19]

4.5 Polysomnografie

Polysomnografie (zkráceně PSG) označuje simultánní záznam několika různých fyziologických funkcí organismu během spánku. Obvykle zahrnuje alespoň několik kanálů EEG se sledovaným frekvenčním pásmem v rozsahu 0,3 až 35 Hz nebo více, dále obsahuje záznam pohybu očí (horizontální a vertikální), s nastavením filtrace frekvenčního pásma shodně jako u signálů EEG, a obvykle také záznam EMG zachycující aktivitu svalstva brady se sledovaným frekvenčním pásmem mezi 10 až 100 Hz. Tyto signály mají při záznamu PSG také dobře definované další vlastnosti, jako například impedanci elektrod, časové a amplitudové rozlišení záznamu a podobně, tak aby je mohl jednoduše hodnotit zdravotnický personál ideálně zcela nezávisle na původu PSG záznamu.

Z uvedených signálů je nejdůležitější záznam EEG kanálů, které umožňují poměrně dobře rozlišit mezi jednotlivými spánkovými stádii. Signály EMG a EOG slouží jako zpřesňující informace, díky které je na základě měřené svalové aktivity možné lépe rozlišit mezi bdělostí, spánkem REM a ostatními stádii nebo díky pomalým a rychlým očním pohybům (SEM a REM) možné lépe rozlišovat mezi spánkem REM a ostatními stádii.

PSG záznam bývá také doplněn řadou dalších záznamů jako je saturace arteriální krve kyslíkem, dechová aktivita, pohyb hrudníku a břicha nebo záznam EKG. Tyto dodatečné signály mohou poskytovat další informaci o průběhu spánku a celkovém stavu pacienta nebo mohou sloužit při detekci různých patologických projevů během spánku (neurologická nebo kardiovaskulární onemocnění, spánková apnoe a jiné.). Požadavky na vlastnosti průběhů a způsobu měření těchto signálů nejsou striktně definovány a mohou se mezi různými pracovišti výrazně lišit.. Vyjma EKG se jedná o signály, u kterých je často sledována stejnosměrná složka signálu a u nichž je významné pásmo často v rozsahu 0 nebo 0,1 až 15 Hz. [19]

Saturace kyslíkem je obvykle měřena pomocí optických senzorů, o dýchací aktivitě také vypovídá nepřímo pohyb břicha a hrudníku měřený pomocí pásů s piezoelektrickými

senzory, případně se užívá měření ezofageálního tlaku. Samotný dech je také možné sledovat několika způsoby, přičemž nejednodušší je společné měření dechu v oblasti úst a nosu pomocí termistoru, ale používá se také pneumotachograf, tedy měření celkového průtoku vzduchu, nebo tlakové senzory pro měření nazálního tlaku. [19]

PSG záznam je možné ohodnotit a rozdělit jej do několika spánkových stádií, přičemž je záznam obvykle pomyslně rozdělen do jednotlivých epoch o délce třiceti vteřin. [8] Pro klasifikaci záznamu již existuje řada automatických metod ale rozhodujícím standardem je stále vizuální hodnocení prováděné doktorem nebo jiným zdravotnickým personálem.

4.6 Spánek a jeho stádia

Vědecké zkoumání spánku a jeho fází vznikalo jako samostatná disciplína postupně a z počátku bylo provázáno s rozvojem měřících metod pro záznam a analýzu biologických signálů, z nichž nejvíce výzkum spánku závisel na postupném vývoji a zkoumání EEG záznamů. Prvním vědcem, který v roce 1929 popsal rozdíly mezi průběhy EEG během bdělého stavu a během spánku, byl německý psychiatr Hans Berger, který také zároveň jako první identifikoval alfa vlny.

Postupně docházelo k objevování dalších závislostí mezi spánkem a jeho projevy v EEG i dalších biologických záznamech a spolu s tím docházelo také postupně k popsání různých stádií spánku. Až v roce 1968 ale došlo k ustanovení první ucelené metodiky hodnocení spánku a jeho stádií, jež sestavila skupina vědců, kterou vedli Allan Rechtschaffen a Anthony Kales, podle nichž se tato metodika obvykle zkráceně označuje jako R&K. [8]

Záznam PSG je dle hodnocení R&K rozdělen do epoch o délce 20 nebo 30 sekund a na základě vizuální analýzy minimálně jednoho kanálu EEG, obvykle kanál C3 nebo C4, a za využití záznamů EOG a EMG svalstva brady je možné jednotlivým epochám přiřadit jednu ze šesti fází spánku, z čehož jedna z těchto fází připadá na stav bdělosti a zbylých pět tak připadá na samotný spánek. Klasifikační kritéria dle R&K jsou pro zajímavost shrnuta v tabulce č. 1.

Výzkum spánku samozřejmě pokračoval intenzivně i po vydání R&K manuálu a spolu s novými poznatky vznikla potřeba revidovat a rozšířit stávající hodnocení spánku, obzvláště s ohledem na rozdílnost spánku u dětí a dospělých. Významným moderním

manuálem pro klasifikaci spánku se po svém vydání na konci roku 2006 stal manuál, který vznikl pod záštitou organizace *American Academy of Sleep Medicine* (zkráceně AASM) a který mírně pozměnil a rozšířil kritéria jednotlivých spánkových stádií. Tento manuál navíc oddělil jednotlivá pravidla pro vizuální hodnocení PSG u dětí a dospělých. [17]

Tabulka 1: Klasifikační kritéria pro vizuální hodnocení PSG podle metodiky klasifikace spánku Rechtschaffen a Kales. Převzato z [8].

Fáze spánku	Kritérium
Bdění	Ve více než 50% epochy převládá rytmus alfa (8-13 Hz), anebo aktivita s nízkou amplitudou v pásmu 2-7 Hz.
NREM, stádium 1	Smíšená aktivita 2-7 Hz v asi 50% epochy a zároveň méně než 50% rytmu alfa. Lze zaznamenat pomalé valivé pohyby očí v délce několik sekund.
NREM, stádium 2	Přítomnost spánkových vřeten a K-komplexů s trváním alespoň 500 ms. Méně než 20% epochy může také obsahovat pomalé signály a vysoké amplitudě (>75 μV, <2 Hz).
NREM, stádium 3	20%-50% epochy obsahuje pomalé signály s vysokou amplitudou (> 75 μV, < 2 Hz).
NREM, stádium 4	Více než 50% epochy delta aktivity - pomalé signály s vysokou amplitudou (> 75 μV, < 2 Hz).
Spánek REM	Smíšená aktivita nízké amplitudy v pásmu 2-7 Hz. Epizody rychlých očních pohybů. Výrazný pokles nebo vymizení svalové aktivity brady.

Klasifikace podle AASM a podle R&K nejsou zcela kompatibilní a nejvýznamnějším rozdílem je v novém manuálu sloučení stádií S3 a S4 do jednoho stádia N3. Jinak jsou jednotlivá stádia v zásadě porovnatelná, přičemž stádia S1 a S2 byla přejmenována na N1 a N2. V mnoha aspektech, jako je latence spánku, celková délka trvání spánku a podobně, se oba manuály v zásadě shodují. Na druhou stranu dochází mezi manuály ke změně poměrů mezi trváním jednotlivých stádií spánku a proto není možné hodnocení libovolně zaměňovat.

Detailněji zkoumá rozdíly mezi oběma manuály článek [24]. Samotná klasifikační kritéria podle AASM nejsou pro tuto práci podstatná, ale manuál je v prvním vydání dostupný k nahlédnutí i online viz. [17].

5 Metody zpracování biomedicínský signálů

5.1 Předzpracování signálu

5.1.1 Segmentace

Před samotnou analýzou signálu je nutné signál vhodně předzpracovat a segmentovat. Signál lze segmentovat buď za pomoci konstantní segmentace, kdy je signál rozdělen do bloků o konstantní velikosti, nebo lze použít adaptivní segmentaci, kdy je velikost a pozice signálových bloků určována na základě zvolených kritérií za pomoci vhodných segmentačních algoritmů. [6]

Výhoda adaptivní segmentace je především v možnosti rozdělit signál dle sledovaných parametrů tak, aby v rámci jednoho signálového bloku nedocházelo k výrazné kvalitativní změně parametrů daného signálu, což umožňuje dosáhnout lepších výsledků v dalších krocích zpracování signálu. Nevýhodou adaptivní segmentace je kromě dodatečného výpočetního výkonu pro analýzu parametrů signálu také potřeba pracovat s dostatečně velkými úseky signálu tak, aby bylo vůbec možné signál do jednotlivých oken smysluplně rozdělit. Oba tyto nedostatky jsou obzvláště výrazné u zpracování signálu v reálném čase, protože mohou zavádět do systému dodatečné špatně předvidatelné zpoždění.

Konstantní segmentace oproti tomu rozděluje signál do předem definovaných datových bloků o konstantní délce. Data jsou navíc připravena k analýze ihned, jakmile je k dispozici dostatek vstupních dat pro naplnění segmentu. Nevýhodou tohoto přístupu je, že může docházet k výrazným změnám charakteristiky signálu v rámci jednoho segmentu nebo naopak k rozdělení některých tranzientních dějů nevhodným způsobem do více po sobě jdoucích bloků, což může vést k horším výsledkům v následujících fázích zpracování signálu.

V této práci byla vzhledem k omezeným výpočetním zdrojům a k požadavku na zpracování signálu v reálném čase zvolena právě konstantní segmentace, která usnadňuje realizaci kroku předzpracování signálu a zároveň umožňuje přenechat více výpočetního výkonu pro samotnou analýzu signálu.

5.1.2 Filtrace signálu

Výběr filtru a návrhové metody

Důležitou součástí předzpracování signálu je také filtrace signálu, která slouží k odstranění nežádoucích frekvencí, rušení nebo šumu. Důvodem pro filtrování signálu může být i požadavek na jeho převzorkování, kdy před decimací nebo po interpolaci signálu musí být omezeno frekvenční spektrum signálu.

Vzhledem k omezeným možnostem a dostupnosti open-source DSP knihoven pro platformu Android bylo nutné v této práci implementovat vlastní způsob návrhu číslicových filtrů a také vlastní proces samotné filtrace. Předem jsem vyřadil možnost využití IIR filtrů, které sice běžně dosahují nižšího řádu než filtry typu FIR, ale mohou být nestabilní a obvykle mají nelineární fázi. Stabilitu IIR je možné zajistit pomocí vhodných návrhových pravidel, ale nelineární fázi nelze spolehlivě ošetřit například použitím nekauzální filtrační metody jako např. *Zero-phase IIR filtering* [5, str. 627] kvůli nutnosti filtrace v reálném čase a použitím nevhodných IIR filtrů by tedy mohlo dojít k významnému zkreslení výsledného signálu.

Alternativou je tedy návrh a využití vhodných filtrů typu FIR, které sice dosahují několikanásobně vyšších řádů než jejich IIR ekvivalenty, ale za pomoci vhodných algoritmů je možné je efektivně nasadit při zpracování signálů v reálném čase.

Vzhledem k šířce a obecnosti zadání této práce není možné předem definovat jaké filtry bude aplikace potřebovat a není proto možné vygenerovat předem nějakou konečnou množinu filtrů, která by pokryla potřeby této práce. Všechny filtry a jejich koeficienty je proto nutné generovat dynamicky přímo v aplikaci dle konkrétních vlastností daných signálů. Z toho důvodu bylo nutné definovat a implementovat univerzální postup pro návrh filtrů založený na tzv. metodě oken.

Návrh filtrů FIR pomocí metody oken

Návrh číslicových filtrů FIR typu dolní propust metodou oken sestává z následujících kroků: [5, str. 564]

1. Formulace zadání filtru a určení hodnoty maximální velikosti zvlnění v propustném i závěrném pásmu a určení středu přechodového pásma dle vzorce č. 1 a šířky přechodového pásma pomocí vzorce č. 2.

$$\omega_c = \frac{\omega_p + \omega_s}{2} \tag{1}$$

$$\Delta\omega = \omega_s - \omega_p \tag{2}$$

2. Na základě výše uvedených vlastností a tabulky z obrázku č. 3 vybrat vhodnou funkci okna a dostatečný řád filtru.
3. Určit impulsní odezvu $h_d(n)$ ideálního filtru, který je ve frekvenční oblasti popsán přenosovou funkcí $H_d(\omega)$ dle vztahů č. 3 a č.4. Výslednou impulsní odezvu lze popsat vzorcem č. 5, který reprezentuje obdélníkem oříznutou a zpožděnou verzi analyticky získané impulsní odezvy z původní přenosové funkce tak, aby výsledný filtr splňoval požadavky linearity fáze, konečné odezvy a kauzality.

$$H_d(\omega) = 1, \quad |\omega| \leq \omega_c \tag{3}$$
$$H_d(\omega) = 0, \quad \omega_c < |\omega| < \pi \tag{4}$$

$$h_d(n) = \frac{\sin(\omega_c(n - M/2))}{\pi(n - M/2)} \tag{5}$$

4. Posledním krokem je získání impulsní odezvy navrhovaného filtru pomocí vynásobení odezvy $h_d(n)$ s funkcí zvoleného signálového okna $w(n)$ dle vztahu č. 6, tato operace odpovídá konvoluci obou signálů ve frekvenční oblasti. Posledním bodem by mělo být ověření návrhu a splnění požadovaných vlastností navrženého filtru.

$$h(n) = h_d(n)w(n) \tag{6}$$

Window name	Side lobe level (dB)	Approx. $\Delta\omega$	Exact $\Delta\omega$	$\delta_p \approx \delta_s$	A_p (dB)	A_s (dB)
Rectangular	−13	$4\pi/L$	$1.8\pi/L$	0.09	0.75	21
Bartlett	−25	$8\pi/L$	$6.1\pi/L$	0.05	0.45	26
Hann	−31	$8\pi/L$	$6.2\pi/L$	0.0063	0.055	44
Hamming	−41	$8\pi/L$	$6.6\pi/L$	0.0022	0.019	53
Blackman	−57	$12\pi/L$	$11\pi/L$	0.0002	0.0017	74

Obrázek 3: Tabulka s vlastnostmi vybraných oken. L = M + 1, M označuje řád filtru. Převzato z [5, str. 563].

Daný postup se týká pouze filtru typu dolní propust, pokud ale máme možnost vytvořit filtr typu dolní propust, je možné ukázat, že se dá daným postupem složit libovolný vícepásmový filtr. Například filtr typu horní propust s danou mezní frekvencí lze získat vztahem č. 7, kde výrazem $1_{(-\omega_c,\omega_c)}(\omega)$ se v tomto kontextu míní přenosová funkce $H_d(\omega)$ navrhovaného filtru typu dolní propust dle vztahů č. 3 a č.4. Jedná se tedy vlastně o rozdíl jednotkového přenosu v celém pásmu a frekvenčního přenosu ideálního LP filtru.

$$H_d(\omega) = 1_{(-\pi,-\omega_c)\cup(\omega_c,\pi)}(\omega) = 1_{(-\pi,\pi)}(\omega) - 1_{(-\omega_c,\omega_c)}(\omega) \tag{7}$$

Pomocí skládání přenosových funkcí lze získat libovolný předpis ideální filtru. Rozdílem dvou LP filtrů lze snadno získat filtr typu pásmová propust, součtem přenosu LP a HP filtrů lze naopak získat filtr typu pásmová zádrž.

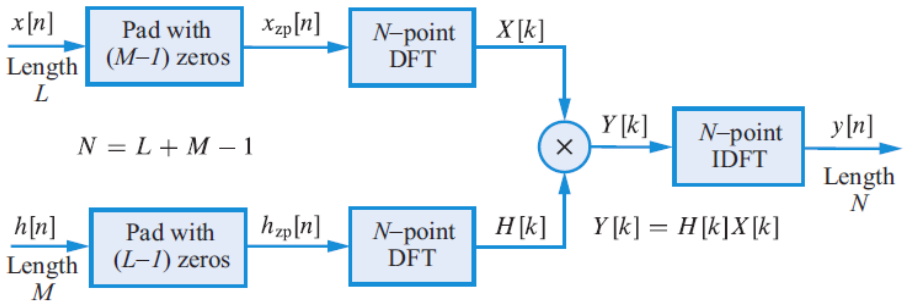
Metoda filtrace

Pokud máme signál $x(n)$ o délce L a impulsní odezvu filtru $h(n)$ o délce M, pak je možné filtrovaný signál získat jako lineární konvoluci těchto dvou průběhů. Konvoluce v časové oblasti ale kvůli asymptotické složitosti $O(LM)$ není při vysokých řádech filtru právě efektivní a je proto potřeba nalézt lepší způsob jejího výpočtu.

Možným řešením tohoto problému je převedení lineární konvoluce na konvoluci cyklickou, kterou lze efektivně provést ve frekvenční oblasti pomocí doplnění obou signálů nulami na společnou délku $L+M-1$ vzorků, provedení diskrétní Fourierovy transformace,

vynásobení obou obrazů a převedení výsledku zpět do časové oblasti tak, jak znázorňuje obrázek č. 4. Tento proces je samozřejmě složitější a s jeho realizací je spojená jistá režie, ale pro vyšší řády filtru je jednoznačně efektivnější, jelikož asymptotická složitost diskrétní rychlé Fourierovy transformace stejně jako její inverze náleží do třídy složitosti $O(n \log(n))$.

Další komplikaci představuje použití této metody pro filtraci dlouhých segmentovaných signálů, jelikož výsledný signál je kvůli zpoždění způsobenému konvolucí posunut a filtrovaný segment proto vždy závisí na výstupu z předešlého segmentu. Obvyklou metodou, jak filtrovat dlouhé diskrétní segmentované signály ve spektru, je metoda uložení přesahů nebo metoda sčítání přesahů. [5, str. 395]



Obrázek 4: Výpočet lineární konvoluce y(n)=x(n) ⁎ h(n) dvou diskrétních posloupností pomocí diskrétní Fourierovy transformace. [5, str. 392]

V aplikaci byla využita metoda sčítání přesahů, která nevyžaduje částečný překryv segmentů, ale místo toho si mezi segmenty udržuje v paměti *M-1* posledních filtrovaných vzorků z předchozího segmentu, které jsou po filtraci následujícího segmentu přičteny k jeho prvním *M-1* vzorkům. Platných je vždy prvních *L* vzorků výsledného filtrovaného signálu, které se postupně spojují za sebe, čímž vzniká výsledný filtrovaný signál ekvivalentní lineární konvolucí impulsní odezvy filtru a originálního nesegmentovaného signálu.

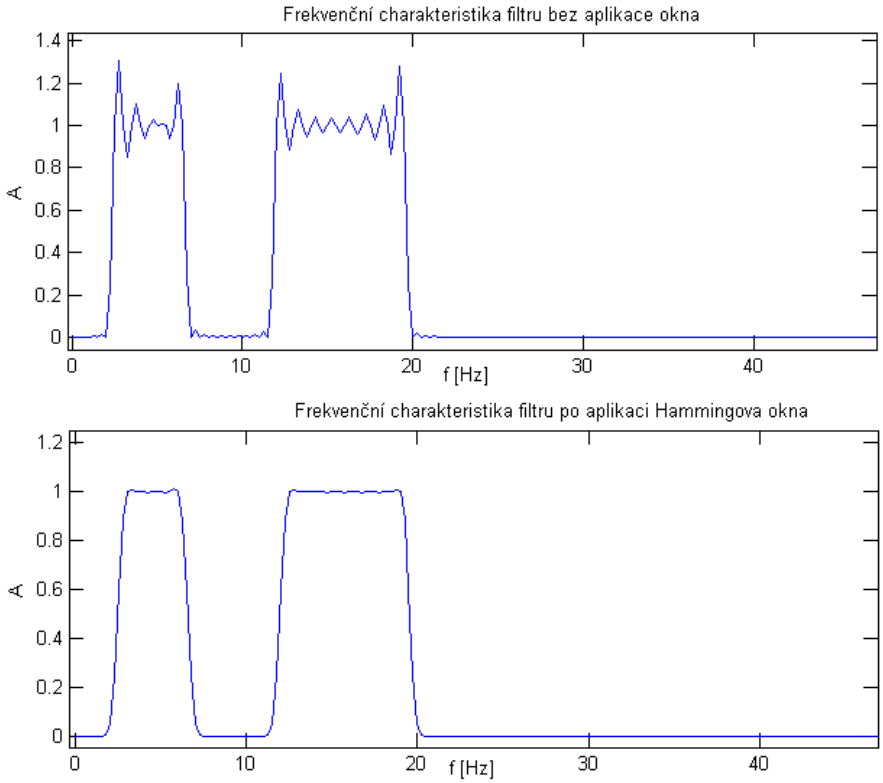
Implementace a ukázka návrhu filtru

Při implementaci návrhu filtru bylo získání ideální přenosové funkce zjednodušeno tak, že platí $H_d(\omega)=1$ pokud ω leží v propustném pásmu a $H_d(\omega)=0$ pokud ω leží ve filtrovaném pásmu. Datové pole odpovídající jednotlivým vzorkům přenosové funkce je tak možné vyplnit jedničkami a nulami pomocí jediného lineárního průchodu tímto polem, nicméně korektnost této operace je založena na myšlence sčítání a odčítání různých ideálních LP filtrů vzniklých dle postupu uvedeného výše. Dalším problémem

bylo určení parametrů a řádu filtru. Vzhledem k tomu, že nebyly určeny žádné konkrétní parametry a požadavky na filtrování signálu, bylo pro návrh filtru zvoleno Hammingovo okno s funkčním předpisem dle vztahu č. 8.

$$w(n) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{M}\right) \tag{8}$$

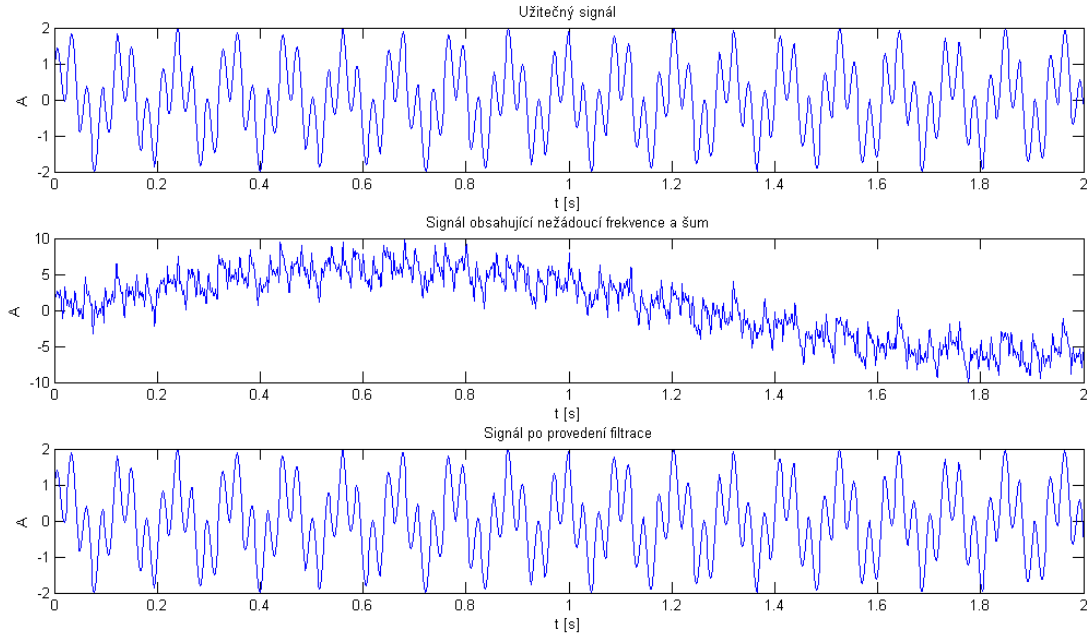
Hammingovo okno představuje dobrý kompromis mezi šířkou přechodového pásma, zvlněním signálu v propustném pásmu a odstupem potlačeného pásma. Obecně pracují požadavky na strmost přechodu filtru a minimalizaci zvlnění v propustném pásmu proti sobě, což je vidět obzvláště u obdélníkového okna, kde se kvůli Gibbsovu jevu [5, str. 149] objevují poblíž hraniční frekvence filtru výrazné oscilace, které vznikají kvůli snaze aproximovat náhlé nespojitosti funkce pomocí konečné řady spojitých funkcí. Gibbsovy oscilace, které nelze bohužel omezit ani zvyšováním řádu filtru, je možné redukovat právě pomocí zmíněných oken, která nespojitost zmírňují, ale za cenu rozšíření přechodového pásma. Hammingovo okno proto představuje dobrou alternativu k obdélníkovému oknu a zároveň nabízí i dostatečnou strmost přechodu filtru, což přispívá k redukci řádu filtru. Ilustrace rozdílu frekvenční charakteristiky syntetizovaného filtru před a po aplikaci Hammingova okna je znázorněna na obrázku č. 5.



Obrázek 5: Ilustrace rozdílu ve frekvenční charakteristice filtru před a po aplikaci Hammingova okna.

Při implementaci filtru v aplikaci jsou jako parametry zadány vzorkovací frekvence, definice pásem a požadovaná maximální šířka přechodového pásma. Dle požadované šířky pásma je poté spočítán řád filtru a jsou vygenerovány průběhy $H_d(\omega)$ a $w(n)$. Hodnoty $h_d(n)$ nejsou získány analyticky ale jsou aproximovány za pomoci zpětné Fourierovy transformace obrazu $H_d(\omega)$. Poté jsou pomocí vzorce č. 6 spočítány výsledné koeficienty filtru $h(n)$, které jsou následně pomocí Fourierovy transformace opět převedeny do frekvenční oblasti do obrazu $H(\omega)$, který je nakonec použit při filtraci samotné.

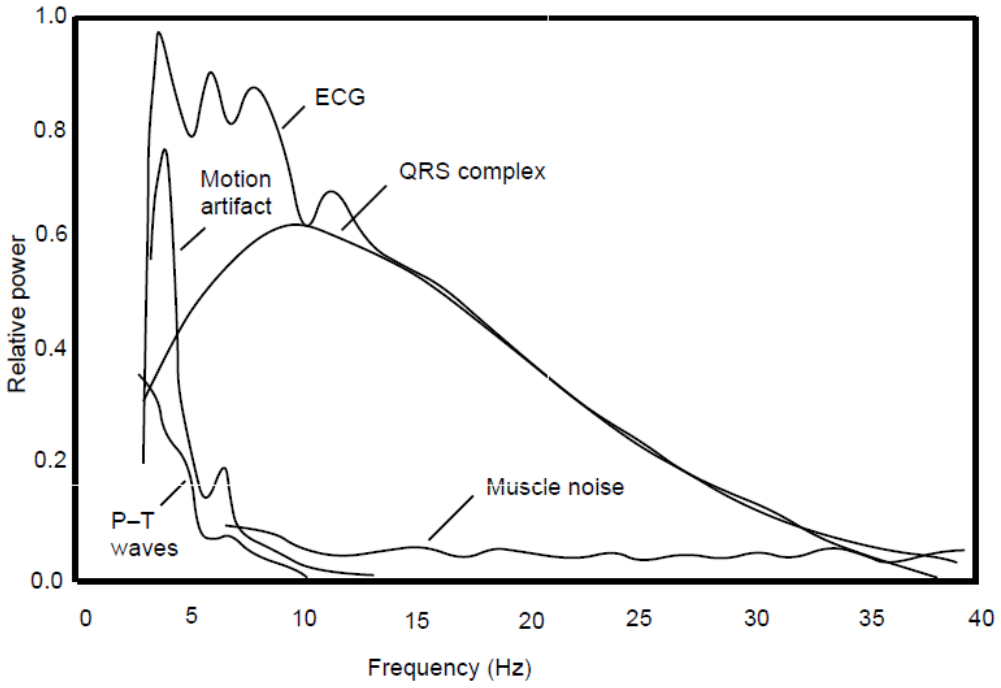
Na obrázku č. 6 jsou demonstrovány výsledky filtrace pomocí navržené metody na syntetizovaném signálu o délce šestinásobku řádu filtru. Užitečný signál vznikl složením dvou sinusových průběhů s jednotkovou amplitudou a frekvencemi 9,31 Hz a 34,2 Hz. Zarušený signál, který je zobrazen jako druhý signál na obrázku č. 6, obsahuje bílý šum s poměrem SNR o hodnotě 15 dB a dále další sinusové průběhy o jednotkové amplitudě a frekvencích 0,4 Hz, 19 Hz, 50 Hz, 100 Hz a 150 Hz. Poslední zobrazený průběh představuje zrekonstruovaný signál za použití filtru navrženého výše popsanou metodou, jehož frekvenční charakteristika je znázorněna na obrázku č. 5.



Obrázek 6: Ilustrace výsledků testu navrženého filtru a filtrace metodou sčítání přesahů.

5.2 Výpočet srdeční frekvence ze signálu EKG

Vyhodnocení srdeční frekvence ze signálu EKG je relativně snadno řešitelný a dobře prozkoumaný problém, který je možné řešit bez větších obtíží i v reálném čase. V časové oblasti je velmi výraznou součástí EKG signálu QRS komplex, který má dobře definovaný tvar, průběh i dobu trvání a je proto obvyklým cílem algoritmů pro určování srdeční frekvence. Existují i obecné algoritmy na bázi vlnkové transformace nebo autokorelace EKG signálu, které také mohou být využity k výpočtu tepové frekvence, ale pokud je srdeční frekvence jediným cílem výpočtu, pak stačí využít i starší ověřené algoritmy pracující v časové oblasti, jež využívají dobře definovaných vlastností QRS komplexu a jež obvykle pomocí filtrace a prahování nebo vzájemné korelace s předem definovaným vzorem, dokáží dostatečně přesně lokalizovat čas výskytu QRS komplexu ve vstupním signálu. [4]



Obrázek 7: Normalizované výkonové spektrum EKG signálu s vyznačením QRS komplexu, P a T vln a rušení způsobeného pohybem a svalovou aktivitou. [4]

EKG signály jsou obyčejně vzorkovány s frekvencí alespoň 200 Hz a obvyklou frekvenční charakteristiku EKG znázorňuje obrázek č. 7. Z obrázku je patrné, že pro určování srdeční frekvence je možné se omezit na poměrně úzké spektrum. Typicky je vhodné odfiltrovat nízké frekvence, kde v pásmu pod 1 Hz je signál ovlivněn například dýcháním, kolísáním reference elektrod a dalšími elektrodovými artefakty. Dále je z obrázku je patrné, že v nízkých frekvencích se také může objevovat rušení způsobené pohybem pacienta a svalovými artefakty. Pásmo nad 30 Hz už pro určování srdeční frekvence nenese mnoho relevantní informace a některé algoritmy může šum z těchto vyšších frekvencích významně rušit. Proto je vhodné odfiltrovat i tyto vyšší frekvence, čímž zároveň dojde k potlačení případného rušení elektrické sítě na 50 nebo 60 Hz a vyšších harmonických.

Často používaným algoritmem pro určování srdeční frekvence je Pan-Tompkinsův algoritmus popsaný již v roce 1985. Algoritmus využívá kombinace filtrace přes dolní a horní propust, diferenci, umocnění a průměrování a nakonec několika pravidel pro prahování výsledného signálu. V této práci nicméně používám o něco starší alternativní algoritmus, který je více náchylný na vysokofrekvenční šum, ale v kombinaci s filtrováním nabízí spolehlivou a hlavně rychlou metodu výpočtu tepové frekvence. Tento algoritmus popsaný v roce 1983 Ahlstromem a Tompkinsem je založen na diferenci signálu a následnému prahování. [4]

Základem použitého algoritmu, který je popsán pomocí vztahů č. 9 až č. 11, je vážený součet první a druhé diference vstupního signálu. Výsledný signál je poté prahován pomocí zvolené konstanty. Jelikož vstupní signál není nijak normalizován, byla empiricky zvolena konstanta prahu jako půlka maxima výstupního signálu, přičemž maximum je dynamicky aktualizováno přes celý zpracovávaný signál.

$$y_0(nT) = |x(nT) - x(nT - 2\mathrm{T})| \tag{9}$$

$$y_1(nT) = |x(nT) - 2\mathrm{x}(nT - 2\mathrm{T}) + x(nT - 4\mathrm{T})| \tag{10}$$

$$y_2(nT) = 1.3\, y_0(nT) + 1.1\, y_1(nT) \tag{11}$$

Algoritmus byl testován pro vstupní vzorkovací frekvence 256 a 512 Hz, pro vyšší frekvence je potřeba signál vhodně podvzorkovat. Algoritmus při překročení daného prahu vyhodnotí aktuálně zpracovávaný vzorek jako novou polohu QRS komplexu a přeskočí několik následujících vzorků, které zhruba odpovídají 100 ms intervalu v daném signálu.

Algoritmus je dle [4] možné při detekci překročení prahu dále vylepšit pomocí kontroly následujících 8 vzorků, přičemž je nová pozice označena jako QRS komplex pouze tehdy pokud alespoň šest z těchto osmi vzorků také překročilo daný práh. Díky použití diference prvního a druhé řádu dochází k filtraci nízkofrekvenční části spektra signálu, ale na druhou stranu je daný postup citlivý na vysokofrekvenční rušení, které může vést k falešně pozitivní detekcím, a pro zvýšení odolnosti je proto možné algoritmus doplnit o vhodně zvolený filtr typu dolní propust.

Samotná srdeční frekvence v jednotkách srdečních akcí resp. úderů za minutu je pak určena po každé detekci QRS komplexu pomocí vztahu č. 12, kde Δn_{QRS} představuje počet vzorků mezi předchozím a aktuálním QRS komplexem.

$$BPM = \frac{60 f_{vz}}{\Delta n_{QRS}} \tag{12}$$

5.3 Spánek a klasifikace stádií spánku

5.3.1 Analýza problému

Automatická analýza PSG signálů a klasifikace stádií spánku je poměrně aktivně zkoumané téma a existuje řada přístupů počínaje využitím standardizovaných definic pro vizuální hodnocení jednotlivých spánkových stádií v časové oblasti, přes využití analýzy ve frekvenční oblasti, až po komplexní přístupy využívající vlnkové transformace, neuronové sítě a jiné pokročilé metody signálové analýzy a strojového učení. Příkladem komplexního přístupu a analýzy může být například knihovna PSGLab pro MATLAB vyvíjená přímo na Fakultě elektrotechnické ČVUT v Praze.[13]

Problémem komplexnějších přístupů je jejich výpočetní náročnost a omezená možnost aplikace v reálném čase. Z toho důvodu jsem se zaměřil spíše na jednodušší jednokanálové způsoby klasifikace. Příkladem přístupu využívajícího jednokanálové EEG je například článek [14] nebo články [11], [12], které s úspěchem využívají frekvenční obraz EEG signálu pomocí analýzy vlastností několika vybraných frekvenčních pásem.

Další zajímavý článek [10], kterým se tato práce inspiruje, popisuje klasifikaci spánku v reálném čase pomocí analýzy EOG s velice dobrou přesností přesahující 80%. Ani tento klasifikátor nicméně nepracuje opravdu v reálném čase, jelikož není kauzální a obsahuje řadu pravidel, která zpětně upravují minulá rozhodnutí klasifikátoru a tedy ve výsledku může být finální hodnocení dané epochy závislé na budoucích epochách. Článek [9] nakonec popisuje možnosti využití EOG a EMG signálů pro rozlišení mezi stádii bdělosti a spánku REM.

Vzhledem k nutnosti klasifikace v reálném čase a možnosti využití kombinace EEG, EOG a EMG signálů jsem se nakonec rozhodl pro navržení vlastního klasifikátoru, který využívá poznatky z výše uvedených článků. Zároveň bylo snahou této práce vytvořit klasifikátor, který nebude spoléhat na předem dané rozsahy amplitud jednotlivých signálů a který bude odolný vůči rušení v časové oblasti. Rozhodl jsem se proto využít vlastního přístupu, který spoléhá na frekvenční analýzu jednotlivých signálů a který je založený na porovnávání hladin normalizovaných energií v sousedících pásmech daného signálu a zároveň rozpadu klasifikačního problému do několika postupných kroků pomocí rozdělení problému do více disjunktních klasifikačních skupin.

5.3.2 Zdroj dat

Jako zdroj reálných klinických dat pro učení klasifikátoru a jeho vyhodnocení byly použity PSG záznamy z databáze *The Sleep EDF Database (Expanded)* ze serveru projektu PhysioNet. [15] Jedná se o sbírku 61 PSG záznamů, ke které jsou navíc přiloženy i hypnogramy ohodnocené nezávislými experty. Každý záznam je uložen ve formátu EDF a sestává ze dvou EEG kanálů, elektrod Fpz-Cz a Pz-Oz, o vzorkovací frekvencí 100 Hz, jednoho EOG kanálů (horizontální) taktéž vzorkovaného frekvencí 100 Hz, jednoho EMG kanálu (svaly brady) a některé záznamy dále obsahují záznam dechu a tělesnou teplotu. Jednotlivé průběhy zaznamenávají kompletní spánek pacientů a pocházejí ze dvou různých studií, kterých se zúčastnili muži i ženy ve věku od 25 do 101 let.

Fakt, že data pocházejí ze dvou různých studií umožňuje lépe otestovat klasifikátor, jelikož tak klasifikace není vázaná pouze na jeden specifický zdroj dat. Kvalitativní rozdílnost dat ale představuje jistý problém při učení klasifikátoru, jelikož data ze druhé studie obsahují poměrně málo epoch se stádiem bdělosti, a navíc jsou EMG signály zaznamenány v těchto studiích rozdílným způsobem.

V první studii procházelo EMG postupně filtry typu horní a dolní propust a poté z něj byla počítána efektivní hodnota, jež byla zaznamenávána s frekvencí 1 Hz. Druhá studie obsahuje EMG signál vzorkovaný frekvencí 100 Hz bez použití filtrace. Aby tedy bylo možné použít EMG z obou studií, byly EMG signály z druhé studie v rámci testování filtrovány pásmovou propustí s propustným pásmem 15-40 Hz a z tohoto filtrovaného signálu byla poté vypočítána efektivní hodnota, čímž bylo dosaženo podobných průběhu EMG u obou studií.

Poskytnuté hypnogramy jsou ručně ohodnoceny lidskými experty dle manuálu Rechtschaffen and Kales [16] a jednotlivá stádia jsou označena jako W*, R, 1, 2, 3* a *4*. Dále také hypnogram obsahuje hodnocení *M* a *?,* která přísluší označení pohybu pacienta a nebo v případě hodnocení *?* jakékoliv jiné nehodnotitelné epoše z důvodu přítomnosti artefaktů nebo jiných příčin.

Pro účely této práce bylo těchto 61 záznamů podle svého názvu sestupně lexikograficky seřazeno a rozděleno dle pořadí na sudé a liché záznamy. Liché záznamy byly použity pro učení klasifikátoru, přičemž se v průběhu práce ukázalo, že jeden z těchto záznamů není možné vůbec přečíst a další obsahují v hypnogramu neoznačené artefakty (dočasné úplné výpadky elektrod), které sice algoritmus detekoval, ale tyto záznamy byly

i tak z učení pro jistotu vynechány. Nakonec tak na učení připadlo dvacet vstupních záznamů. Sudé záznamy pak byly určeny jako testovací k vyhodnocení přesnosti klasifikátoru, testování klasifikátoru tedy probíhalo na jiných datech, která nebyla k dispozici v době učení.

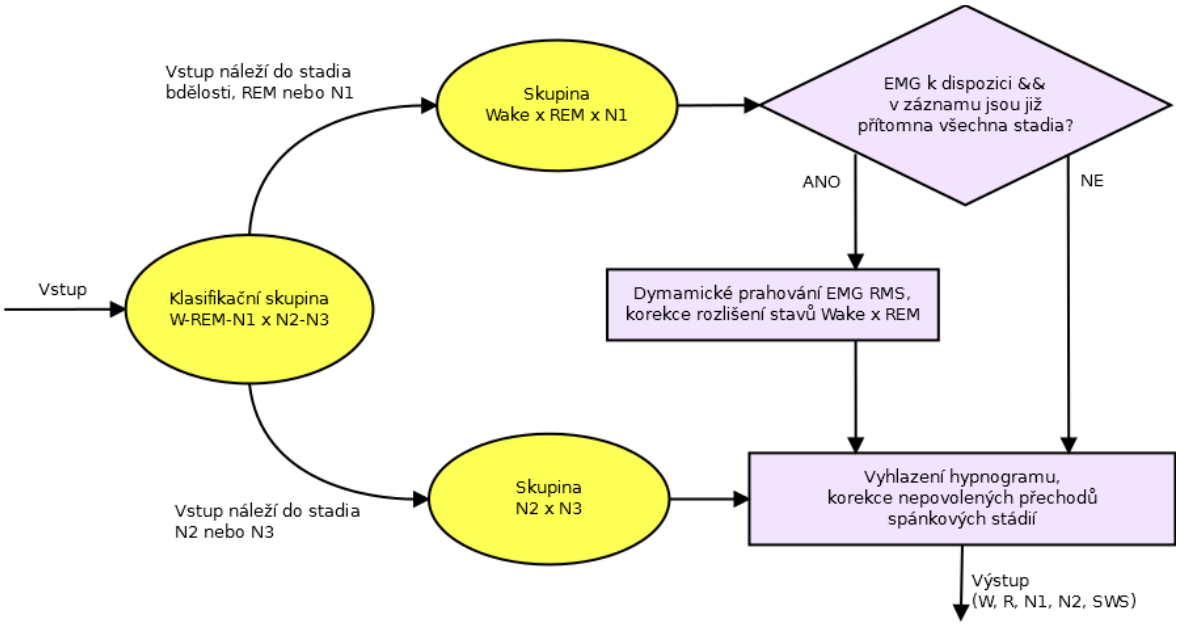
5.3.3 Klasifikátor

Obecný popis

Navržený klasifikátor předpokládá na vstupu jeden EEG kanál (Pz-Oz), jeden kanál EOG (horizontální) a volitelně také jeden EMG kanál zaznamenávající svalovou aktivitu brady. Jeho výstupem je pak hypnogram s pěti výstupními stádii REM, N1, N2, hlubokého spánku SWS a také stádiem bdělosti. Klasifikace byla testována na epoše o trvání 30 sekund, která se často používá i při ručním hodnocení [16], ale v principu může fungovat na epochách o libovolné délce.

Klasifikace u signálů EEG a EOG probíhá čistě ve frekvenční oblasti, zatímco doplňková klasifikace EMG je založena na dynamickém prahování efektivní hodnoty signálu v časové oblasti. Klasifikace probíhá v několika krocích, kdy je nejprve vyhodnoceno, zda vstup spadá do klasifikační skupiny, jež obsahuje stádia bdělosti, spánku REM a spánku N1, anebo zda spadá do druhé skupiny, která reprezentuje stádia N2 a SWS (SWS odpovídá zhruba sjednocení stádií S3 a S4 dle [16] resp. stádiu N3 dle [17]). V rámci těchto dvou skupin je poté vstupu přiřazeno již konkrétní spánkové stádium.

Posledním krokem algoritmu je kontrola několika pravidel, kdy například není dovoleno přecházet ze stavu bdělosti přímo do spánku REM, a hlavně také vyhlazování průběhu, kdy k opravdovému přechodu výstupu do nového stádia dochází až v okamžiku, kdy bylo toto stádium detekováno v současné a zároveň i předchozí epoše. Klasifikátor proto obsahuje vnitřní paměť s vnitřním hodnocením minulých stádií, která se nemusejí shodovat s výstupem klasifikátoru v předešlých epochách. Klasifikátor navíc také obsahuje paměť pro maximální a minimální efektivní hodnoty EMG signálu, které jsou v průběhu měření dynamicky aktualizovány.



Obrázek 8: Schéma funkce a rozhodování klasifikátoru.

Pokud je navíc vstup označen jako fáze bdělosti nebo spánek REM a je k dispozici EMG kanál, tak ještě před samotným vyhlazováním signálu dochází také k prahování efektivní hodnoty EMG pro danou epochu vůči maximální a minimální hodnotě efektivních hodnot, jež byly pozorovány během aktuálního záznamu. Dodatečnou podmínkou použití efektivní hodnoty EMG je navíc to, že v aktuálním záznamu musí být po dobu alespoň jedné epochy přítomno každé z hodnocených stádií, jinak je prahování přeskočeno.

Prahování je nastaveno tak, že pokud leží efektivní hodnota v horních 20 % procentech intervalu omezeným minimem a maximem z naměřených efektivních hodnot, je stádium označeno jako bdělost. Pokud naopak spadá efektivní hodnota do spodních 10 % je stádium označeno jako spánek REM. Pokud efektivní hodnota nepřekročí ani jeden práh je ponecháno ohodnocení z předchozího kroku. Postupný proces rozhodování klasifikátoru je znázorněn na obrázku č. 8.

Klasifikace dle EOG a EEG signálu pak spočívá na převodu signálů do frekvenční oblasti, rozdělení amplitudového spektra signálu do pásem, normalizaci těchto pásem pomocí součtu amplitud v celém použitém spektru, výpočtu podílů mezi sousedícími pásmy a nakonec výpočtu Euklidovské vzdálenosti těchto vektorů podílů od pevně daných předem naučených vektorů, získaných během fáze učení.

Učení

Učení probíhá na dvaceti vstupních PSG souborech z obou zdrojových studií ze vstupních signálů EEG (Pz-Oz) a horizontálního EOG. Do učení není nijak zahrnut signál EMG, který je vyhodnocován dynamicky až při samotné fázi klasifikace. Dalším vstupem učícího algoritmu jsou také definice pásem pro jednotlivé klasifikační skupiny a manuálně ohodnocené hypnogramy pro vstupní PSG. Výstupem je pro každou klasifikační skupinu matice, jejíž jednotlivé řádky sestávají z vektorů podílů sousedících pásem pro jednotlivé klasifikační třídy použitých klasifikačních skupin. Algoritmus učení lze popsat následujícími kroky:

1. Časové průběhy EEG i EOG signálů pro danou epochu jsou pomocí rychlé Fourierovy transformace převedeny do frekvenční oblasti.
2. Z frekvenčního obrazu je získáno amplitudové spektrum dle vztahu č. 13.

$$A(\omega)=\sqrt{S^{*}(\omega)S(\omega)} \tag{13}$$

3. Dále je spočítána suma amplitud v použité části spektra vzorce č. 14.

$$A_{\text{sum}}=\sum_{\omega_{min}}^{\omega_{max}} A(\omega) \tag{14}$$

4. Následuje výpočet dílčích sum v definovaných pásmech jednotlivých klasifikačních tříd dle předpisu č. 15.

$$A_{bandsum}(n)=\sum_{\omega_{band}(n)}^{\omega_{band}(n+1)} A(\omega), n=\{1, 2 \dots L_b\}, \tag{15}$$
kde L_b je počet pásem klasifikační skupiny

5. Jednotlivé sumy jsou poté normalizovány pomocí sumy amplitud celého pásma dle vztahu č. 16.

$$A^{rel}_{bandsum}(n)=\frac{A_{bandsum}(n)}{A_{\text{sum}}}, n=\{1, 2 \dots L_b\}, \tag{16}$$
kde L_b je počet pásem klasifikační skupiny

6. Nakonec je vypočten vektor podílů sousedních pásem dle vzorce č. 17.

$$V(n)=\frac{A_{bandsum}^{rel}(n)}{A_{bandsum}^{rel}(n+1)}, n=\{1,\ 2\ \ldots\ L_b-1\}, \quad (17)$$
$$\textit{kde } L_b \textit{ je počet pásem klasifikační skupiny}$$

7. Kroky č. 1 až č. 6 jsou opakovány nezávisle pro EEG i EOG kanál dokud nejsou takto vyhodnoceny všechny epochy PSG záznamu. Poté algoritmus pokračuje krokem č. 8.
8. Pro každou epochu záznamu je sestaven nový vektor podílů *V(n)*, který vznikne dle předpisu č. 18 konkatenací původních *V(n)* vektorů EEG a EOG kanálů.

$$V_E(n)=\left[V_E^{EEG}(n)\ \ V_E^{EOG}(n)\right], n=\{1,\ 2\ \ldots(\ L_b^{EEG}+L_b^{EOG}-2)\},$$
$$\textit{kde } L_b \textit{ je počet pásem EEG nebo EOG dané klasifikační skupiny}, \quad (18)$$
$$E \textit{ označuje index epochy}, E=\{1\ldots N_E\}, N_E \textit{ značí počet epoch v PSG záznamu}$$

9. Vektory podílů jsou poté na základě manuálně ohodnoceného hypnogramu po jednotlivých epochách zařazeny do příslušejících klasifikačních tříd dané klasifikační skupiny (krok 9a). Nakonec je z vektorů podílů v rámci těchto tříd vypočten aritmetický průměr jednotlivých složek těchto vektorů (krok 9b).
   a. Každé epoše je na základě manuálně hodnoceného hypnogramu přiřazen index, který označuje, do které klasifikační třídy patří. Do této klasifikační třídy je pak zařazen i vektor podílů dané epochy. Přiřazení indexu se děje dle mapovací funkce dle předpisu č. 19.

$$\forall E: \vec{I}_E=F_{cls}(H(E)),\ \ E\in\{1,\ldots,N_E\},$$
$$F_{cls}:\{W,S1,S2,S3,S4,R\}\rightarrow\{1,\ \ldots,N_{cls}\}, \quad (19)$$
$$N_{cls} \textit{ označuje počet klasifikačních tříd}$$

   b. Pro každou třídu je pomocí aritmetického průměru dle vzorce č. 20 vypočten nový vektor podílů, takový vektor je optimální pokud je kritériem minimalizace sumy kvadrátů Euklidovské vzdálenosti vektorů. [18]

$$\forall E\,\forall c: \vec{V}_c^m=\frac{\sum_{I_E^c}\vec{V}_E}{\left|I_E^c\right|},\ I_E^c=\{I_E; I_E=c\},\ c\in\{1,\ \ldots,\ N_{cls}\} \quad (20)$$

10. Formálně jsou získané vektory pro každou klasifikační skupinu seřazeny do matice způsobem popsaným vztahem č. 21

$$M = \begin{bmatrix} \vec{V}_1^m \\ \vec{V}_2^m \\ \cdots \\ \vec{V}_{N_{cls}}^m \end{bmatrix} \tag{21}$$

11. Kroky 1 až 10 je nutné opakovat pro všechny klasifikační skupiny. Matice všech klasifikačních skupin jsou nakonec výstupem samotného algoritmu učení.

Rozdělení frekvenčních pásem jednotlivých klasifikačních skupin a klasifikační vektory použité při vyhodnocování výsledků v této práci jsou uvedeny v syntaxi jazyku Java v příloze č. 1.

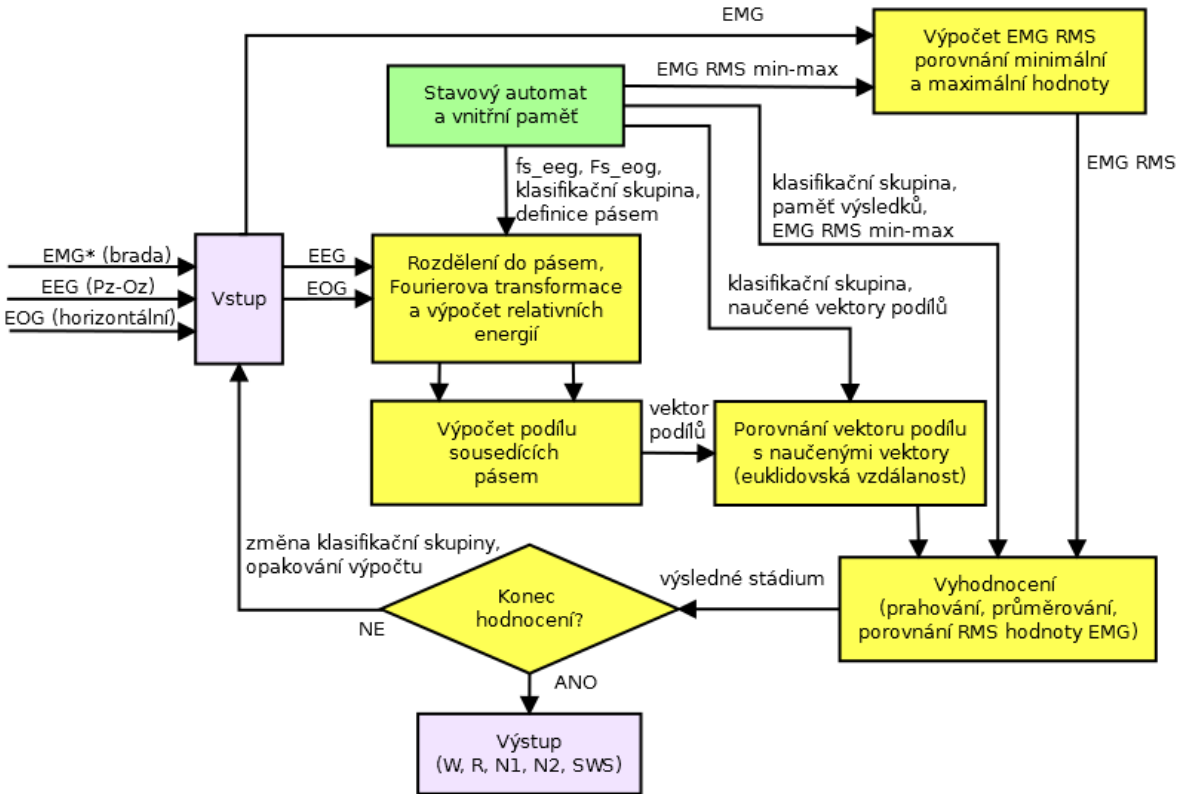
Klasifikace

Blokové schéma klasifikace jedné epochy PSG záznamu navrženým algoritmem znázorňuje obrázek č. 9. Ze schématu je vidět, že vstupem algoritmu je jeden signál EEG, jeden signál EOG a jeden kanál EMG, který ale není povinný a algoritmus je navržen tak, že může pracovat i bez jeho přítomnosti. Parametrem algoritmu může být také definice rozdělení pásem klasifikačních skupin a jejich klasifikační matice, ale vzhledem k relativně časově náročnému procesu učení jsou tyto údaje v aplikaci definovány přímo v kódu algoritmu.

Výstupem algoritmu je ohodnocení dané epochy jako jednoho z pěti možných stádií, kterými jsou bdělost, spánek REM, spánek N1, N2 nebo spánek SWS, který odpovídá sjednocení stádií S3 a S4 podle klasifikačního manuálu R&K.

Jak je vidět na obrázku č. 9, jsou signály EOG a EEG nejprve převedeny do frekvenční oblasti pomocí rychlé Fourierovy transformace a následně rozděleny do pásem dle definic pásem pro aktuální klasifikační skupinu. Za zmínku stojí, že na signály není aplikováno před transformací žádné signálového okno pro potlačení prosakování ve spektru, protože pokusy s násobením signálu pomocí Hammingova okna vedly obecně spíše ke zhoršení výsledků klasifikace.

Dále následuje několik kroků, které jsou identické s kroky č. 1 až č. 6 z fáze učení. Z frekvenčního obrazu signálů je vypočteno amplitudové spektrum a z tohoto spektra je vypočtena celková suma amplitud použitého spektra a také dílčí sumy amplitud z jednotlivých frekvenčních pásem. Tyto dílčí sumy jsou poté vyděleny celkovou sumou amplitud spektra. Následně jsou ještě z těchto normalizovaných součtů jednotlivých pásem získány podíly mezi sousedícími spektry. Vektory podílů sousedících spekter získané ze signálu EEG i EOG jsou poté sloučeny do jednoho vektoru, jenž je dále porovnáván s klasifikačními vektory v rámci dané klasifikační skupiny. Podobnost vektorů je vyhodnocena pomocí Euklidovské vzdálenosti a klasifikační třída s níž má vektor podílů dané epochy minimální vzdálenost je prohlášena za výslednou.



Obrázek 9: Blokové schéma klasifikace jedné epochy PSG.

Výsledek porovnání s klasifikačními vektory je dále zpracováván vnitřní logikou aplikace. Pokud je k dispozici EMG záznam, je vypočtena jeho efektivní hodnota za danou epochu, která je poté porovnána s minimem a maximem EMG efektivních hodnot, které byly naměřeny od počátku měření až do současné epochy. Pokud už během současného záznamu došlo k průchodu všech stádií spánku, dochází k prahování efektivní hodnoty tak, jak je popsáno výše v obecném popisu klasifikátoru, a případné úpravě hodnocení stádia

dané epochy. Ke změně výstupu dochází navíc pouze tehdy, pokud se výsledné stádium současné epochy shoduje s výsledkem epochy minulé, čímž je dosaženo efektu vyhlazení výsledného hypnogramu, i když takové vyhlazování zavádí při použití v reálném čase dodatečné zpoždění o délce jedné epochy.

Popsané hodnocení navíc probíhá v několika krocích, jak znázorňuje obrázek č. 8, za použití několika odlišných klasifikačních skupin. Nejdříve dochází k rozlišení zda vstup náleží do skupiny stádií bdělosti, spánku REM a spánku N1, anebo zda náleží do skupiny obsahující stádia spánku N2 a SWS. Poté následuje další iterace klasifikace, kdy již dochází k přiřazení konečného stádia v rámci těchto podskupin. Výsledek klasifikace je nakonec průměrován a případně dochází k prahování efektivní hodnoty EMG podle výše popsaného postupu.

5.3.4 Výsledky

Klasifikační algoritmus byl testován ve dvou verzích a to nejprve bez použití EMG kanálu a následně i s použitím prahování efektivní hodnoty EMG. Testování probíhalo na testovacích instancích vstupních PSG záznamů smíšených z obou použitých zdrojových studií tak, jak je uvedeno v popisu zdrojových dat.

Průměrné výsledky shrnují tabulky č. 2 a č. 3, které ukazují úspěšnost algoritmu pro jednotlivá spánková stádia. Pro zajímavost byla přiložena i tabulka č. 4, která shrnuje výsledky algoritmu při vynechání EOG i EMG záznamů během klasifikace. Hodnoty tabulky reprezentují relativní četnost ohodnocení stádia daného řádku jako stádia popsaného daným sloupcem, hodnota na prvním řádku v posledním sloupci tedy například reprezentuje relativní četnost, kdy byla epocha označená v manuálně hodnocených hypnogramech jako bdělost klasifikována algoritmem jako stádium spánku REM. Relativní četnosti na hlavní diagonále tabulky tak představují shodu hodnocení zdrojových hypnogramů a navrženého klasifikačního algoritmu v jednotlivých spánkových stádiích.

Tabulka 2: Průměrná úspěšnost klasifikačního algoritmu bez použití EMG.

	Bdělost	N1	N2	SWS	REM
Bdělost	64,25%	19,82%	3,97%	0,94%	11,03%
N1	13,66%	40,65%	17,02%	2,72%	25,94%
N2	1,32%	5,75%	77,38%	9,26%	6,29%
SWS	0,47%	0,50%	20,41%	78,05%	0,57%
REM	6,61%	7,31%	7,46%	0,54%	78,08%

Shoda celkem	**73,87%**

Z tabulek vyplývá, že přesnost algoritmu se pohybuje okolo 73,5 %, což bohužel nedosahuje přesnosti shody hodnocení mezi manuálně hodnocenými hypnogramy, která se například podle článku [25] obyčejně pohybuje okolo 76 %. Z tabulek je dále vidět, že navržený postup je poměrně úspěšný při rozlišení skupiny se stádii N2 a SWS proti zbytku hodnocených stádií. Na druhou stranu nejslabším výsledkem je přesnost hodnocení spánku N1, který algoritmus nedokáže dobře rozlišit a přiřazuje toto stádium často do jiných stádií. Chybné hodnocení spánku N1 navíc není konzistentní a v různých testovacích instancích má tendenci být přiřazováno do různých stádií, což vysvětluje rozprostření četností tohoto hodnocení do více stádií a což navíc znemožňuje tento problém nějak výrazněji ošetřit. S tímto stádiem nicméně mají problémy i citované automatické klasifikátory [11], [12] nebo [14], a tak se dalo toto chování do jisté míry očekávat.

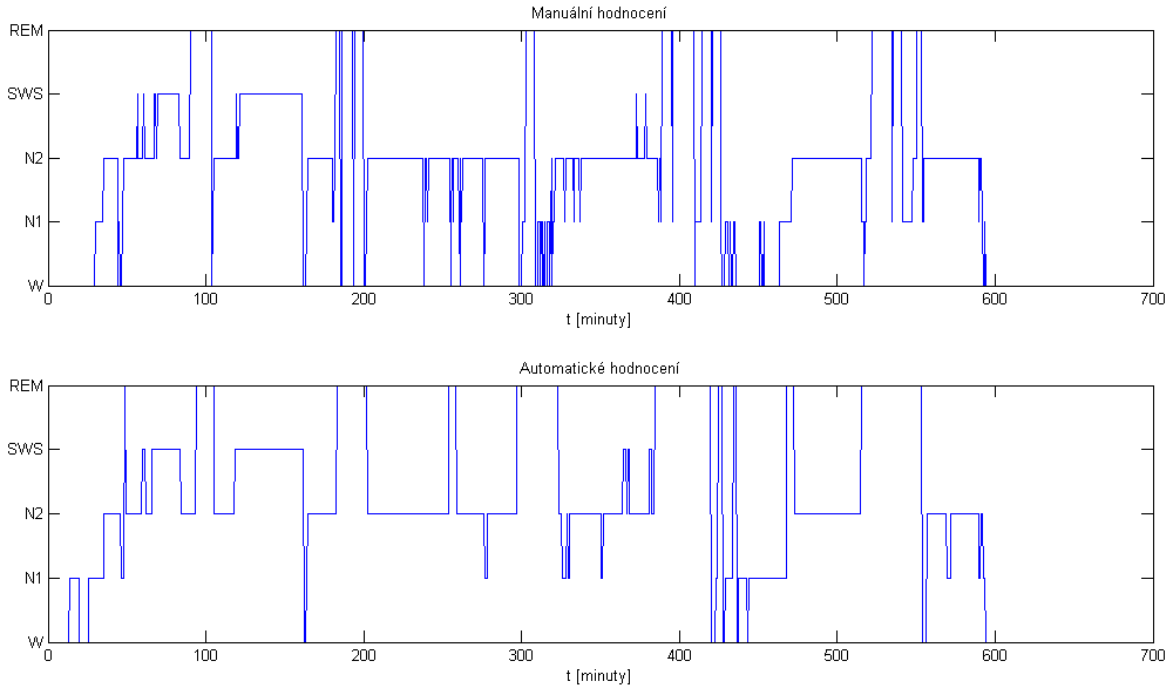
Tabulka 3: Průměrná úspěšnost klasifikačního algoritmu s použitím EMG kanálu.

	Bdělost	N1	N2	SWS	REM
Bdělost	70,10%	13,01%	3,39%	0,87%	12,62%
N1	30,50%	23,95%	15,50%	2,49%	27,56%
N2	3,93%	3,51%	77,09%	9,16%	6,31%
SWS	1,04%	0,42%	20,45%	77,71%	0,38%
REM	12,00%	1,09%	6,40%	0,41%	80,11%

Shoda celkem	**73,37%**

Z tabulek dále vyplývá, že algoritmus má určité obtíže s klasifikací mezi stádii bdělosti a spánkem REM. Tento jev je dobře vidět i na obrázku č. 10, který ukazuje srovnání výsledného a zdrojového hypnogramu pro jednu z testovacích instancí, kde občas v průběhu spánku dochází k záměně uvedených dvou stádií. Algoritmus nicméně většinou neměl problémy s určením bdělosti před nástupem spánku a ke konci záznamu po úplném probuzení.

Dle očekávání se při zařazení EMG záznamu zlepšilo rozlišení stádií bdělosti a spánku REM, jelikož fáze bdělosti se vyznačuje zvýšenou svalovou aktivitou oproti spánku a naopak pro spánek REM je typická svalová ochablost a významné redukce aktivity EMG. Překvapivé ale je, že místo celkového vylepšení klasifikace došlo paradoxně průměrně k mírnému zhoršení kvůli dalšímu zhoršení přesnosti klasifikace stádia N1.



Obrázek 10: Ukázka hypnogramů se shodou 75,81 %. Spodní hypnogram byl generován navrhovaným algoritmem bez použití EMG signálu.

Tabulka 4: Průměrná úspěšnost klasifikačního algoritmu při vynechání EOG i EMG.

	Bdělost	N1	N2	SWS	REM
Bdělost	63,44%	23,85%	3,49%	1,75%	7,46%
N1	26,17%	23,82%	17,44%	3,49%	29,08%
N2	2,68%	2,24%	73,39%	12,25%	9,45%
SWS	0,32%	0,29%	24,37%	74,14%	0,88%
REM	3,19%	9,79%	6,82%	0,74%	79,47%

Shoda celkem	**70,61%**

Využití verze algoritmu s kanálem EMG tedy obecně nepřináší celkové vylepšení výsledků, ale nabízí možnost vylepšit klasifikaci stádií bdělosti a spánku REM na úkor stádia N1. Navržený algoritmus celkově nepřinesl dostatečně spolehlivé výsledky, aby jej bylo možné použít pro opravdu spolehlivé automatické hodnocení PSG záznamů. Významnou vlastností navrženého algoritmu je ovšem možnost použití v reálném čase a pro orientační automatické hodnocení spánku v průběhu měření poskytuje poměrně dobré výsledky. Algoritmus navíc dokáže pracovat i bez přítomnosti EMG a dokonce i EOG kanálu a v krajním případě tedy nabízí přesnost okolo 70 % při hodnocení v reálném čase na základě jediného EEG kanálu.

6 Architektura a implementace

6.1 Společná část serverové a klientské aplikace

6.1.1 Komunikační rozhraní

Implementační část této práce lze rozdělit na dvě oddělené aplikace. První aplikace zajišťuje funkci serveru, který emuluje biomedicínské zařízení sloužící jako zdroj dat, a hlavní klientská aplikace napsaná pro operační systém Android je pak určená k analýze biomedicínský signálů. Obě aplikace byly realizovány v jazyce Java a přestože serverová část je určena k běhu pod Java SE Runtime Environment, zatímco klientská část je určena k běhu na virtuálním stroji Dalvik platformy Android, sdílejí obě aplikace část kompatibilního kódu realizujícího aplikační část protokolové sady, kterou obě aplikace využívají při vzájemné komunikaci po bezdrátové síti.

Obě aplikace využívají k přenosu dat TCP/IP sokety, jež slouží k výměně dat pomocí obyčejných bajtových proudů. Nedošlo tedy k využití sofistikovanějších možností Javy, jako přenos celých objektů díky mechanismu serializace objektů, což ale umožňuje snadnější zpřístupnění aplikace i jiným programovacím jazykům nebo případně i reálným bezdrátovým zařízením implementujícím shodný komunikační protokol.

Aplikace s výhodou využívá některých vlastností TCP/IP soketů zejména, že je již na síťové vrstvě zajištěno spojované a spolehlivé spojení, které by tedy mělo zajistit doručení všech paketů a navíc ve stejném pořadí, v jakém byly odeslány.

Tabulka 5: Formát komunikačního rámce používaného aplikací.

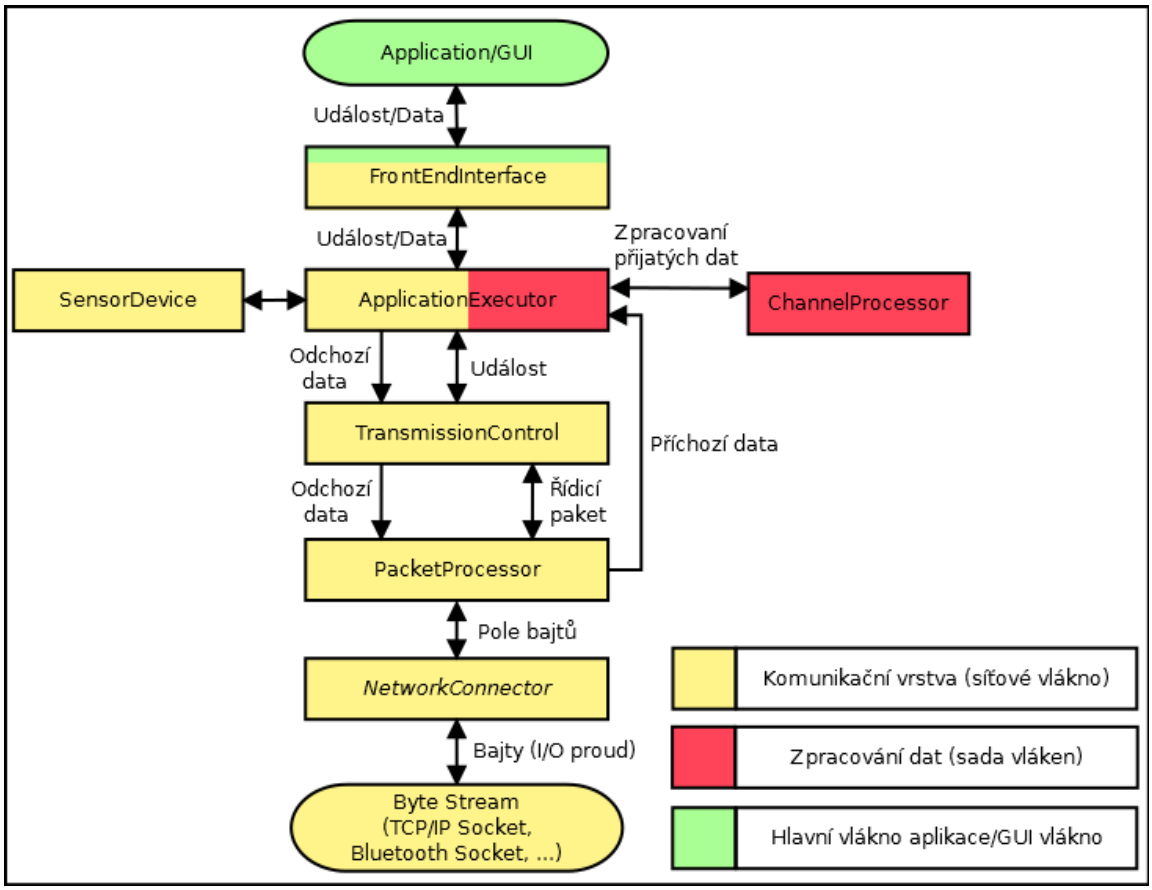
	Začátek rámce	Primární ID	Sekundární ID	Délka datové části	Data	EOF
Délka	2 B	2 B	2B	3 B	...	1 B

Komunikace mezi serverem a klientskou aplikací je v aplikační vrstvě realizována pomocí aplikačních rámců, jejichž formát je popsán v tabulce č. 5. Každý rámec začíná dvěma synchronizačními bajty s hexadecimální hodnotou *FF5A*. Následují dvě pole, každé o velikosti dvou bajtů, určující primární a sekundární identifikační číslo zprávy. Primární identifikační číslo určuje jednoznačně druh zprávy, zatímco sekundární identifikační číslo představuje doplňující informaci k primárnímu ID jako je například číslo kanálu, typ datového přenosu a podobně. Následují tři bajty, které určují délku datové oblasti rámce

v bajtech. Poté, pokud byla délka datové oblasti nenulová, následuje datová oblast, která je určena k přenosu informací o zařízení, nastavení kanálů a primárně také k zasílání naměřených dat. Po odeslání všech dat je rámec zakončen jedním bajtem s hexadecimální hodnotou *63,* po jehož přijetí je interní přijímací stavový automat uveden do výchozího stavu a je připraven přijmout další rámec. Zprávy, které neodpovídají předepsanému formátu jsou ignorovány a vnitřní stavový automat v takovém případě přechází do chybového stavu a čeká na zachycení nového rámce pomocí přijetí dvou synchronizační bajtů.

6.1.2 Komunikační protokol

Úroveň aplikačních rámců je sice základem použitého komunikačního protokolu, ale tvoří pouze nejnižší část uvnitř aplikační části protokolu. Aplikační rámce jsou ve skutečnosti vyhodnocovány pomocí dalších aplikačních vrstev a dále sestavovány do vyšších logických objektů. Pro lepší přehled je na obrázku č. 11 znázorněno blokové schéma tříd zpracovávajících příchozí a odchozí komunikaci.



Obrázek 11: Funkční blokové schéma tříd realizujících komunikační protokol použitý serverovou a klientskou aplikací.

Nejnižší vrstvu tvoří abstraktní třída *NetworkConnector,* která obsahuje jen deklarace vybraných funkcí pro odesílání dat, odpojení soketu a podobně. Třída také obsahuje referenci na instanci objektu *PacketProcessor*. Serverová i klientská aplikace pak musí z třídy *NetworkConnector* dědit a implementovat vlastní způsob příjmu a odesílání jednotlivých bajtů pomocí fyzického síťového rozhraní. Tato vrstva komunikačního protokolu bohužel nejde implementovat obecně, jelikož přímo závisí na knihovnách operačního systému zařízení, na kterém běží, a obě aplikaci navíc mohou potřebovat v chybových stavech přímo přistupovat na tuto úroveň nebo případně přímo k síťovým soketům.

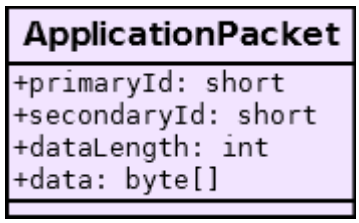


Diagram 1: Pole aplikačního paketu

Dalším důvodem oddělené implementace je možnost rozdělit funkce vrstv*y* definované objektem *NetworkConnector* do více programových vláken dle potřeb aplikace. Jelikož tato vrstva předpokládá komunikaci ve formě prostého bajtového vstupu a výstupu je také výhodou tohoto řešení možnost snadno zaměnit použité síťové rozhraní, kterým by mohl být například soket bezdrátové sítě Bluetooth nebo nějaké jiné komunikační rozhraní, bez toho aniž by bylo nutné zasahovat do dalších vrstev navrženého protokolu.

Následuje třída nazvaná *PacketProcessor,* která z příchozích bajtů sestavuje aplikační rámce a z těchto rámců dále konstruuje řídicí pakety nebo v případě příjmu naměřených dat předává tato data bez úpravy vyšším vrstvám. Tato třída obsahuje implementaci stavového automatu, který zodpovídá za sestavení a integritu příchozích aplikačních rámců, které už byly výše popsány a jejichž formát je znázorněn v tabulce č. 5. Samozřejmě tato třída podporuje i opačný směr a z požadavků a dat z vyšších vrstev umí sestavovat aplikační rámce a odesílat je přes síťové rozhraní.

Řídicí nebo také aplikační paket je logickým obrazem aplikačního rámce, který již obsahuje reprezentaci dat aplikačního rámce ve formě datových typů jazyku Java a se kterými již lze v aplikaci přímo pracovat. Struktura aplikačního paketu je proto takřka identická s již definovaným aplikačním rámcem a je vidět na diagramu č. 1. Metody třídy aplikačního paketu nejsou na diagramu znázorněny, ale samozřejmě i tato třída obsahuje standardní množinu set a get funkcí, i když je z důvodu urychlení zpracování k proměnným objektu díky veřejnému přístupu možné přistupovat i bez užití těchto funkcí.

Tato třída také obsahuje seznam veřejných statických konstant, které reprezentují známé hodnoty primárních ID aplikačních rámců i paketů a které využívají i ostatní třídy komunikačního protokolu. Tok příchozích dat se na úrovni *PacketProcessoru* dělí a dle typu míří výše do třídy *TransmissionControl* nebo v případě naměřených dat přímo do třídy *ApplicationExecutor*.

Třída *TransmissionControl* slouží ke zpracování řídicích paketů a řízení transakcí mezi serverem a zařízením a uchování aktuálního stavu síťové vrstvy. Tato třída má na starost zejména párování zařízení a aplikace po bezdrátové síti, enumeraci nabízených kanálů a jejich vlastností, inicializaci nového měření, výběr kanálů, o jejichž přenos má aplikace zájem, a nakonec i spouštění nebo ukončení samotného záznamu.

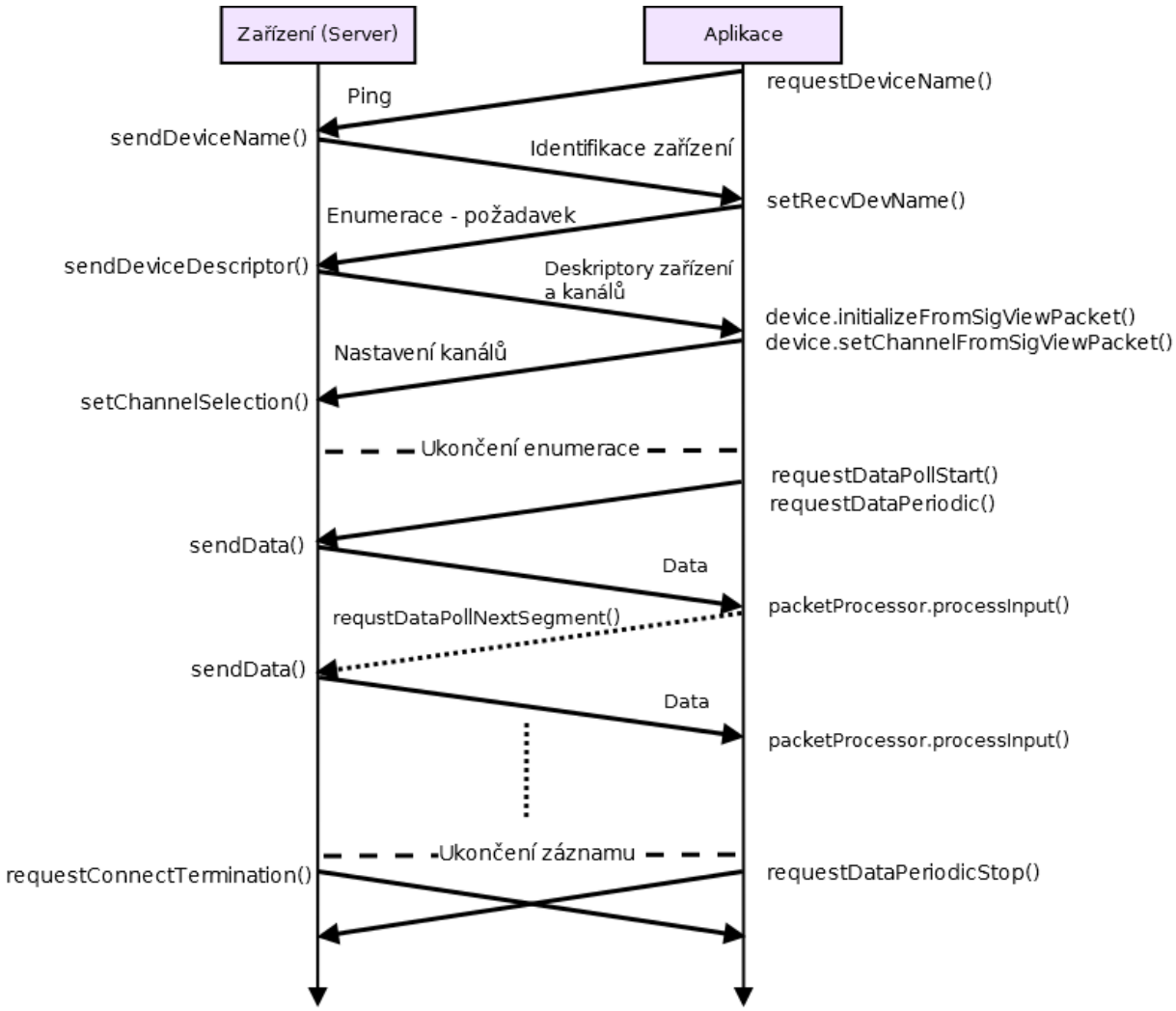
Nad třídou *TransmissionControl* se nachází vrstva *ApplicationExecutor*, která slouží k příjmu a zpracování přijatých dat měření. V této třídě se také nacházejí funkce zodpovědné za rozdělení přijatých dat do jednotlivých kanálů a přiřazení těchto dat jednotlivým výpočetním vláknům, která následně zajišťují zpracování těchto dat. Tato třída také v průběhu sezení spravuje přístup k instanci třídy *SensorDevice*, která představuje logický obraz přípojeného zařízení a která slouží zbytku aplikace jako popis vlastností tohoto zařízení.

Mezi sadou aplikačního protokolu a zbytkem aplikace se nakonec nachází rozhraní s názvem *FrontEndInterface*, které musí aplikace sama dle své vlastní funkce implementovat. V případě PC serveru toto rozhraní implementuje přímo grafické aplikační rozhraní, zatímco v případě aplikace pro Android toto rozhraní implementuje systémová služba *CoreService*, která po vyhodnocení příchozích událostí šíří tyto události dále do aktivit aplikace pomocí asynchronní fronty zpráv.

6.1.3 Enumerace kanálů a navázání spojení

Hlavním úkolem komunikačního protokolu je spolehlivá výměna dat mezi zařízením a aplikací. Než je ale tato data vůbec možné začít přenášet, musí obě komunikační strany úspěšně projít sérií několika kroků, které jsou zde souhrnně označeny jako enumerace. Jedná se o výměnu metadat, která slouží k nalezení zařízení v lokální síti, výměnu deskriptorů popisujících zařízení a jeho jednotlivé kanály, zvolení požadovaných kanálů a nakonec spuštění periodického nebo asynchronního dotazovacího režimu zasílání dat.

Běžný průběh komunikace mezi zařízením a aplikací je zobrazen na obrázku č. 12. Komunikaci iniciuje klient, který do celé lokální sítě zašle zprávu dotazující se na jméno zařízení. Pokud se v síti nachází zařízení s použitým komunikačním protokolem, odešle jako odpověď zprávu se svojí identifikací a IP adresou. Poté může uživatel započít enumeraci a vyžádat zaslání všech dostupných údajů o zařízení a jeho kanálech pomocí dohodnuté struktury deskriptorů.



Obrázek 12: Schéma znázorňující běžný průběh síťového sezení mezi aplikací a zařízením.

Po přijetí deskriptorů čeká aplikace na uživatele, který v rámci nastavení záznamu musí zvolit, jaké kanály se mají přenášet. Uživatel dále během procesu enumerace může zvolit délku segmentů jednotlivých kanálů, případně i úroveň decimace, typ kanálu a podobně. Tyto funkce jsou ale všechny řešeny pomocí interní vyrovnávací paměti a algoritmů na straně aplikace a z pohledu komunikačního protokolu, jsou v dalším kroku opravdu zvoleny pouze jednotlivé kanály určené k přenosu, s tím, že vlastní segmentace a vzorkovací frekvence jsou převzaty od zdrojového zařízení. Po zvolení použitých kanálů je nakonec toto nastavení posláno zpět zařízení, které je tím uvedeno do stavu, kdy je připraveno ke spuštění záznamu.

Začátek záznamu iniciuje klient, který odešle zařízení požadavek o periodické nebo asynchronní zasílání dat. Při periodickém zasílání, začne zařízení odesílat data podle své vzorkovací frekvence a možnosti svých interních segmentových bufferů, jedná se tedy o režim měření v reálném čase. Asynchronní režim je možný pouze při emulaci zařízení, kdy jsou data záznamu čtena ze zdrojového souboru a je tak možné je posílat podle potřeb aplikace. V tomto případě si aplikace žádá pomocí speciálního požadavku o další segment ihned po zpracování segmentu předchozího a může tak data ve výsledku zpracovávat v průměru rychleji než při periodickém režimu.

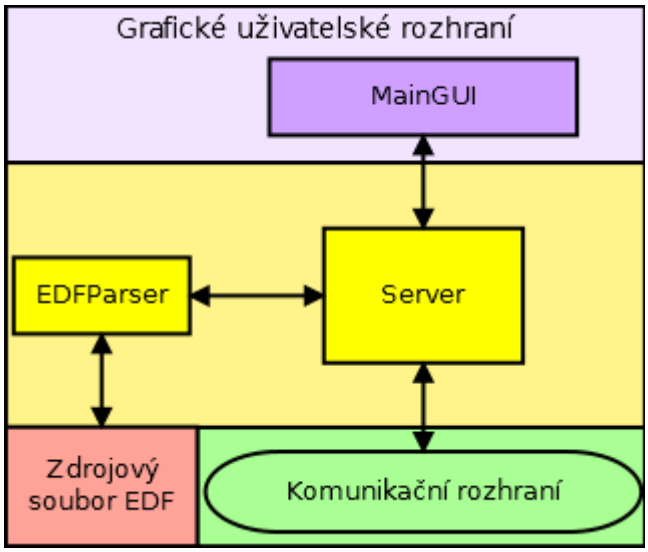
K ukončení záznamu dochází na žádost aplikace pomocí požadavku na ukončení zasílání dat, anebo ze strany serveru, kdy je v případě potřeby ukončit jednostranně službu možné klientovy odeslat oznámení o ukončení činnosti serveru. To zaručuje řádné ukončení záznamu i v případě přerušení ze strany serveru, přestože aplikace dokáže na poruchy serverového soketu po určité prodlevě bezpečně reagovat i bez tohoto oznámení.

6.2 Serverová část aplikace

Před popisem hlavní části práce, kterou je návrh aplikace pro platformu Android, je nutné popsat krátce také serverovou část práce, která je tvořena samostatnou aplikací. Úkolem serverové části této práce je emulace funkce biomedicínského zařízení určeného k měření a přenosu biomedicínských dat v reálném čase. Takové zařízení by, co se týče řídicí části, bylo pravděpodobně realizováno jako relativně jednoduché zařízení s ještě omezenějšími zdroji než jsou přístroje s operačním systémem Android. Z toho důvodu nelze obecně předpokládat, že takové zařízení bude schopné nějakých sofistikovanějších činností kromě sběru dat, jejich dočasného uchování a odesílání po bezdrátové síti. Proto je také serverová část realizována poměrně jednoduše jako pouhá brána mezi zdrojem dat a bezdrátovou sítí, která ale disponuje alespoň na aplikační vrstvě vlastní protokolovou sadou, jež zajišťuje odesílání dat předem dohodnutým způsobem v jednoduché podobě definované jako přenos předem dohodnutých binárních sekvencí.

Serverová část aplikace byla napsána v jazyku Java za použití standardních knihoven a je určena pro běh na osobních počítačích s operačními systémy MS Windows, GNU/Linux, Mac OS nebo Solaris. Pro běh aplikace musí mít zařízení dostupné nějaké úložiště pro uchování zdrojových souborů, obvykle tedy pevný disk, obrazovku pro práci s grafickým prostředím aplikace a přístup k bezdrátovému rozhraní sítě Wi-Fi pomocí protokolu TCP/IP.

Hlavním a v podstatě jediným úkolem serverové aplikace je tedy zprostředkování zdrojových dat po bezdrátové síti aplikaci běžící na zařízení s OS Android. Blokové schéma architektury serverové aplikace je znázorněno na obrázku č. 13.



Obrázek 13: Schéma bloků serverové aplikace.

Třída programu označená jako *MainGUI* slouží jako vstupní bod programu a implementuje zároveň grafické uživatelské prostředí. Grafické rozhraní programu je velice jednoduché a sestává z textového terminálu, který vypisuje uživateli změny stavu aplikace, tlačítka, které otevírá dialogové okno umožňující výběr a otevření zdrojového souboru, a nakonec tlačítka, které spouští samotný server jako síťovou službu.

Třída *EDFParser z*odpovídá za načtení zdrojového souboru a poskytuje serveru informace o vlastnostech dostupných kanálů a na vyžádání zprostředkovává vlastní zdrojová data. Jak název třídy napovídá, zdrojové soubory musí splňovat formát podle definice EDF. [7] Výhodou této třídy je, že implementuje kompletní logiku pro čtení EDF souborů v jazyce Java v rámci jediného souboru bez závislosti na externích knihovnách, a je proto velice snadno přenositelná a dobře aplikovatelná jako modulární rozšíření i do jiných aplikaci. Důležitou vlastností této třídy je také schopnost vyčítání pouze vyžadovaných krátkých částí zdrojového souboru a jejich uložení do mezipaměti, takže aplikace nevyžaduje načtení všech zdrojových dat do operační paměti, jelikož velikost takových dat často pro dlouhodobá měření dosahuje stovek megabajtů a někdy ještě více.

Centrálním prvkem aplikace je třída *Server*. Tato třída je vytvořena na vyžádání grafickým uživatelským rozhraním a jako parametr jejího konstruktoru jí musí být dodán objekt *EDFParser* s načtenými zdrojovými daty. Server se po vytvoření spustí jako služba ve vlastním vlákně a na vytvořeném TCP/IP soketu naslouchá a čeká na připojení klientské aplikace. Server dokáže obsloužit vždy pouze jednoho klienta a případné další klienty odmítá, dokud není ukončeno sezení s právě připojeným klientem. Komunikace pak probíhá podle již popsaného komunikačního protokolu až do ukončení spojení klientem nebo serverem, anebo dokud nejsou přečtena všechna zdrojová data.

6.3 Programové prostředí systému Android

Operační systém Android je open-source operační systém určený primárně pro mobilní zařízení, jako jsou chytré mobilní telefony a tablety, ale také pro další vestavěné systémy jako jsou například chytré televizory, multimediální přehrávače nebo miniaturní počítače.

Z uživatelského hlediska není asi potřeba operační systém Android představovat, jelikož v době psaní této práce běží na asi osmdesáti procentech všech moderních chytrých telefonů a tabletů a s jeho postupným prosazováním v multimediálních aplikacích je již uživatelská znalost OS Android poměrně běžná i mezi netechnickou veřejností.

Z pohledu programátora je platforma OS Android postavena nad linuxovým jádrem upraveným pro konkrétní hardware použitého zařízení a nativními linuxovými knihovnami jako je například databázový stroj SQLite, OpenGL ES, WebKit a SSL a další. Klíčovým prvkem systému je Android Runtime Environment, jenž sestává z jádrových systémových knihoven a hlavně z virtuálního stroje Dalvik, který je upravenou implementací Java Virtual Machine vzniklou pod záštitou společnosti Google.

Vyšší vrstvy systému Android jsou psané v jazyce Java a poté interpretovány pomocí VM Dalvik, což činí OS Android poměrně snadno přenositelný a nezávislý na použité hardwarové platformě. Aplikační framework postavený nad Android Runtime Environment sestává z široké sady objektů, nejčastěji s názvem s příponou Manager nebo Provider, které poskytují uživatelským aplikacím služby jako například bezdrátové sítě, správu zdrojů a paměťového úložiště, telefonní a SMS služby a podobně.

Na aplikační vrstvě pak běží uživatelské aplikace, které jsou svojí architekturou a životním cyklem poměrně odlišné od klasických PC aplikací. Základními stavebními kameny aplikační vrstvy jsou objekty služeb *(Service)*, aktivit *(Activity)* a speciálních systémových zpráv s názvem *Intent*, které slouží k vyvolávání systémových událostí nebo k předávání zpráv.

Aktivita je aplikační komponenta reprezentující jednu obrazovku grafického uživatelského rozhraní. Aktivita tedy zobrazuje výstupní data v grafické podobě a zároveň umožňuje uživateli interagovat s aplikací a zadávat do ní vstupní data. Aplikace v OS Android se obvykle skládá z několika aktivit, které většinou dohromady vytvářejí pomyslnou hierarchickou strukturu všech zobrazení. Po startu aplikace se spouští hlavní

aktivita a ta poté zobrazuje další aktivity, jež samy dále mohou spouštět další aktivity až do libovolného stupně zanoření. Nicméně hierarchie aktivit není nikde explicitně definována a všechny aktivity se mohou mezi sebou volat naprosto bez omezení a velice snadno lze i v konfiguračním souboru aplikace (tzv. manifest) určit jako vstupní aktivitu jakoukoliv z definovaných aktivit dané aplikace.

Mnoho aplikací si vystačí pouze s aktivitami, nicméně existují i problémy, které jsou náročnější na výpočetní čas a mohly by blokovat hlavní vlákno aplikace určené pro vykreslování GUI, případně jsou i činnosti, jako jsou například síťové operace, jejichž spuštění uvnitř aktivit z hlavního GUI vlákna OS Android přímo zakazuje. Proto systém Android nabízí kromě aktivit i služby *(Service)*, které slouží k opakovanému spouštění kódu mimo hlavní vlákno, např. periodický zápis dat, automatická synchronizace aplikace se serverem a podobně. Služby mohou také sloužit k dlouhodobému nepřerušovanému běhu aplikačního kódu mimo GUI vlákno, kde typickým příkladem je třeba přehrávání multimédií, u kterého je obvyklá i funkčnost, aby přehrávání pokračovalo i v době, kdy uživatel uzavře aktivitu dané přehrávací aplikace, což právě služby umožňují.

Dalším důležitým objektem v OS Android jsou speciální systémové zprávy označované jako *Intent*. *Intent* je objekt určený pro systémovou signalizaci událostí a přenos zpráv a obvykle tak slouží k vyvolání žádosti o provedení nějaké akce cílovou aplikační komponentou. Tyto zprávy tak slouží k přenosu žádostí mezi komponentami a navíc mohou nést i další data a parametry pro provedení požadované akce a mohou tak sloužit také k přenosu dat mezi komponentami v rámci jednoho systémového procesu nebo i k přenosu mezi různými procesy.

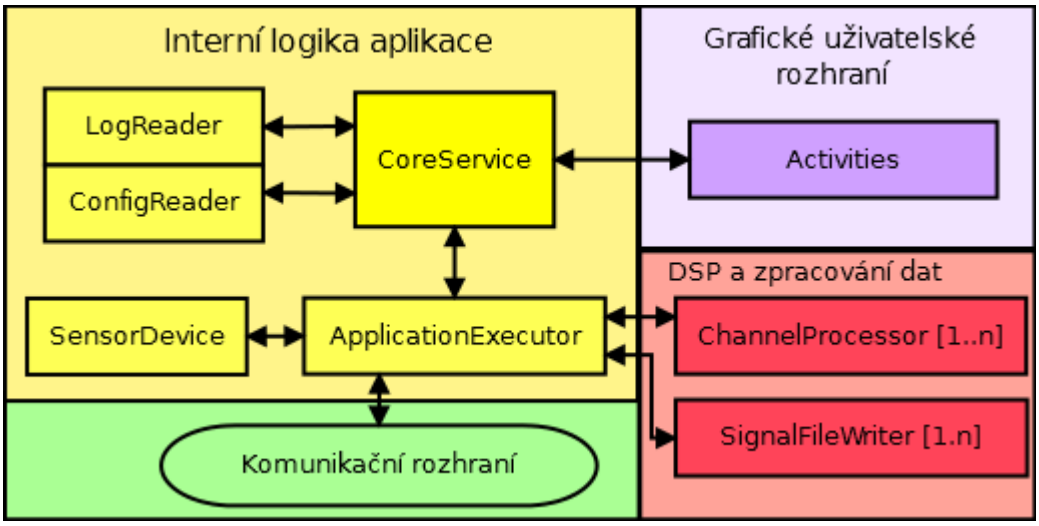
Častým použitím těchto zpráv je spouštění aktivit a služeb nebo zasílání systémových broadcast zpráv, což jsou zprávy šířící se v systému do všech spuštěných procesů informující obvykle o důležitých systémových událostech. Každá aplikace navíc může svým komponentám přiřadit filtry událostí *(Intent Filter)* které určují, jaké zprávy má komponenta zachytávat a na druhé straně slouží i systému, který registruje, že daná aplikace dané typy zprávy zachytává. Příkladem projevu těchto filtrů může být pro uživatele situace, kdy nějaká aplikace žádá o vytvoření nové emailové zprávy. Aplikace kvůli tomu vytvoří novou zprávu a odešle ji systému. Systém poté dle registrovaných filtrů určí, která aplikace umí daný požadavek obsloužit a uživateli nabídne okno se seznamem podporovaných aplikací. Po zvolení aplikace je nakonec systémem tato aplikace spuštěna a daný požadavek je jí poté zpracován. [26]

6.4 Popis architektury aplikace

6.4.1 Přehled bloků aplikace

Architekturu aplikace lze zhruba rozdělit na čtyři relativně nezávislé celky. Hlavním blokem je jádro aplikace, obsahující hlavní službu aplikace, vlákna pro zpracování výpočetních úloh, deskriptory zařízení a záznamu, deskriptory kanálů a metody pro zpřístupnění záznamů v úložišti zařízení a pro řízení komunikačního protokolu a stavu aplikace.

Dalším blokem je již popsaný komunikační protokol, který je definován v samostatném Java balíčku a který je sdílený i se serverovou aplikací. Zvláštní blok pak tvoří třídy pro zpracování signálů a pro zápis výsledků do úložiště zařízení. Poslední blok nakonec tvoří aktivity aplikace, které definují podobu a chování grafického uživatelského rozhraní.



Obrázek 14: Přehled hlavních funkčních bloků klientské aplikace.

6.4.2 Jádro aplikace

Jádro aplikace je tvořeno několika poměrně komplexními třídami, které spravují prostředky aplikace a reprezentují její interní stav. Nejdůležitějším prvkem architektury aplikace je systémová služba *CoreService*. Aplikace je okolo této služby vystavěna a vrstva aktivit tak zajišťuje pouze zprostředkování výstupů a vstupů mezi aplikací a uživatelem s minimem vlastní aplikační logiky a funkcí. Důvodem pro tuto architekturu je předpoklad, že aplikace je určena i pro dlouhodobé záznamy, kdy uživatel s telefonem vůbec nepracuje nebo jej případně používá pro jiné účely a aplikace tak musí nenápadně běžet na pozadí systému.

Stav aplikace je tedy udržován uvnitř třídy *CoreService* a je nezávislý na stavu aktivit a lze v zásadě redukovat do tří skupin – nečinnost *(idle)*, prohlížení záznamu a aktivní režim měření. V případě nečinnosti nebo režimu prohlížení záznamu je možné aktivity aplikace či dokonce i službu kdykoliv ukončit bez dopadu na integritu aplikace. V případě aktivního měření by ale při nečekaném ukončení služby mohlo dojít k poškození dat, služba je proto v tomto režimu přenesena do popředí a je zviditelněna v notifikační liště, což zabraňuje systému tuto službu nečekaně ukončit a navíc je této službě přidělen pomocí správce napájení speciální zámek *(WakeLock)*, který znemožní úplné uspání telefonu dokud je měření aktivní.

Služba *CoreService* dále obsahuje sadu funkcí pro interakci s komunikačním protokolem a zprostředkovává vybrané funkce síťového rozhraní systému spojené s inicializací spojení a hledáním kompatibilních zařízení v lokální síti. Implementuje také rozhraní *FrontEndInterface*, díky čemuž může odchytávat a vyhodnocovat události komunikačního protokolu, které jsou, pokud uživatel právě pracuje s grafickým uživatelským rozhraním aplikace, dále šířeny do právě zobrazované aktivity.

Aktivity mohou využívat metody této služby pomocí přímého volání veřejných metod případně i přístupu k veřejným proměnným. Naopak tato služba s aktivitami komunikovat přímo nesmí a ke sdílení dat je využit mechanizmus asynchronních zpráv systému Android pomocí systémové třídy *Handler.* Služba je od aktivit takto oddělena z důvodu, že se většina událostí šíří z jiných vláken než je hlavní GUI vlákno, které má v systému Android speciální úlohu a je rezervováno pro aktivity aplikace. Tímto oddělením je tedy dosaženo toho, že události z vnitřních vrstev aplikace nemohou nikdy blokovat uživatelské rozhraní.

V jádru aplikace se dále nacházejí další třídy, které přímo vlastní služba *CoreService*, jež slouží k logickému zabalení vybraných funkcí a díky čemuž je vnitřní architektura aplikace lépe strukturovaná.

V tomto prostoru se nachází třída *LogReader*, která obaluje funkce pro čtení a procházení záznamů. Tato třída nabízí aplikaci řadu služeb pro získávání informací o záznamech z datového úložiště zařízení a oprošťuje zbytek aplikace od nutnosti přímého procházení a práce se souborovým systémem operačního systému. Třída nabízí řadu statických funkcí pro čtení záznamů a jejich popisů. Tati třída nenabízí žádný konstruktor a její instanci je možné získat způsobem typickým pro návrhový vzor *factory method pattern*, kdy je instance získána pomocí volání statické metody, která pro zvolený záznam provede načtení všech metadat o záznamu a jednotlivých kanálech z úložiště do operační paměti. Instance třídy pak nabízí pole objektů *ChannelReader,* které obsahují informace o jednotlivých kanálech a nabízí zbytku aplikace metody k vyčítání dat záznamu z úložiště zařízení.

Služba *CoreService* dále obsahuje reference na instance tříd *ApplicationExecutor* a *SigViewWifiConnector*, která je implementací abstraktní třídy *NetworkConnector*. *SigViewWifiConnector* implementuje v rámci komunikačního protokolu přístup k reálnému síťovému rozhraní bezdrátové sítě Wi-Fi na systému Android a zároveň nabízí některé další funkce hlavní službě, jež tuto třídu využívá ke spravování síťových soketů a dalších prostředků spojených se síťovým rozhraním.

Třída *ApplicationExecutor* pak leží na rozhraní aplikace, tříd komunikačního protokolu a modulů pro zpracování signálů. Tato třída sama spravuje sadu běhových vláken určených k výpočtům a zpracování jednotlivých kanálů a přímo těmto vláknům přerozděluje příchozí data bez nutnosti řízení nebo zásahu hlavní služby nebo vyšších vrstev aplikace.

6.4.3 Datové úložiště

V jádru aplikace se nacházejí také třídy *LogReader* a *ConfigReader*, které slouží ke čtení záznamů a nastavení aplikace a třída *SignalWriter*, která naopak slouží jako rozhraní aplikace pro zápis dat do externího úložiště zařízení.

Aplikace ukládá zpracovaná data ve formě binárních souborů přímo do externího úložiště zařízení, tedy na dostupnou paměťovou kartu nebo integrovanou flash paměť. Je použit vlastní proprietární formát ukládaných dat tak, aby bylo dosaženo vysoké hustoty zápisu a nedocházelo ke zbytečnému plýtvání diskovým místem, k čemuž by při použití textového formátu nebo strukturovaných souborů typu XML nebo podobných formátů zcela jistě docházelo. K uchování metadat měření by mohl být také zvlášť využit databázový stroj systémů Android, ale výpočetní a paměťové nároky takového řešení zdaleka přesahuje míru jeho přínosu, a proto byla metadata nakonec uložena spolu s měřenými daty v rámci jednoho souboru.

Měření je definováno pomocí názvu měření a seznamu uživatelem zvolených měřících kanálů. V hlavním adresáři aplikace je pak pro každé měření vytvořen zvláštní adresář, jehož název je shodný s názvem měření, ve kterém se nalézají soubory jednotlivých kanálů, jež pro každý kanál obsahují kombinaci metadat a dat měření.

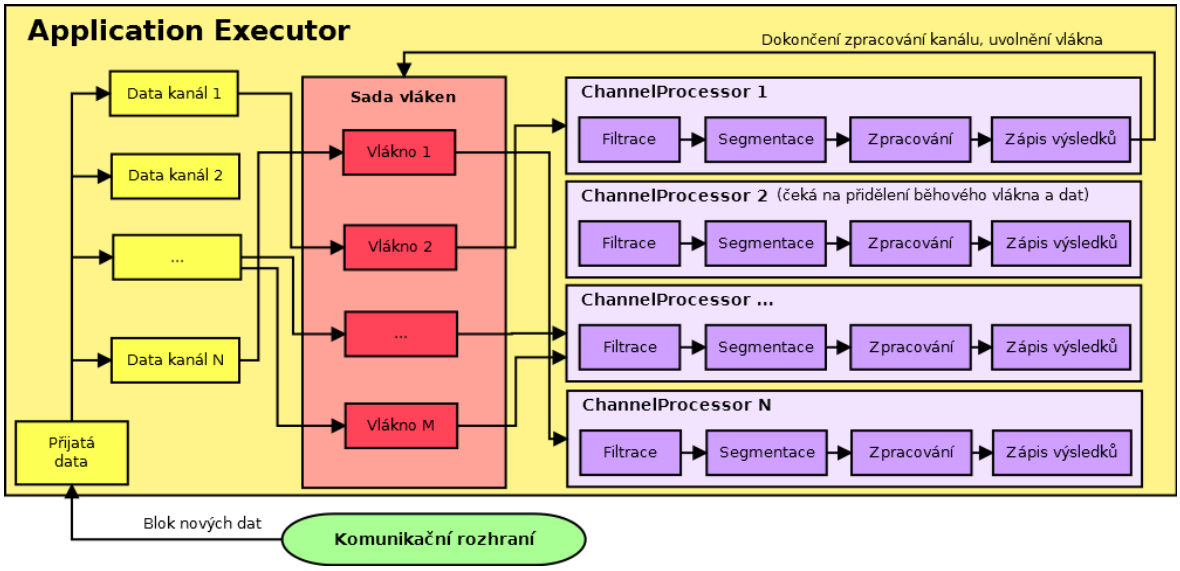
Soubor kanálu je vždy pojmenován názvem daného kanálu a vždy obsahuje alespoň hlavičku o délce 512 B kombinující binární a textová data obsahující informace o fyzikálním rozměru dat, typu kanálu, o délce dat a jejich časovému rozlišení, počtu obsažených signálů a dále obsahuje navíc několik rezervovaných polí pro případné další zpřesňující údaje.

Po hlavičce následují samotná data kanálu, uložená dle definice hlavičky do segmentů o konstantní délce ve formátu pohyblivé řádové čárky s velikostí 32 b dle standardu IEEE754. Formát hlavičky a binárních dat je inspirován formátem EDF a jednotlivá pole hlavičky je možné vyčíst ve třídě *SignalFileWriter* ve zdrojovém kódu projektu.

6.5 Zpracování signálů

6.5.1 Obecný programový postup zpracování dat

Samostatnou část aplikace dále tvoří několik tříd pro zpracování a analýzu signálů. Tyto třídy zaobalují postupy popsané v kapitole č. 5 a propojují je se zbytkem architektury aplikace. Několik tříd také využívá metody knihovny edu.emory.mathcs.jtransforms ve verzi 2.4, která je využita pro výpočet rychlé Fourierovy transformace.



Obrázek 15: Blokové schéma příjmu dat, jejich přiřazení výpočetním vláknům a jejich zpracování.

Klíčovým modulem je třída s názvem *ApplicationExecutor*, která zprostředkovává řízení výpočetních vláken aplikace a slouží do jisté míry jako oddělení komunikačního protokolu od uživatelské části aplikace.

Při běžném provozu jsou navzorkovaná data měřených signálů periodicky zasílána pomocí použitého komunikačního protokolu a ihned po příjetí je reference na buffer s těmito daty předána pomocí blokující fronty instanci třídy *ApplicationExecutor*. Ta pak data přerozdělí jednotlivým kanálům a připravené kanály zařadí do fronty k přidělení výpočetního času a zpracování. Základní model zpracování dat je znázorněn na obrázku č. 15. Na obrázku je znázorněno chování třídy *ApplicationExecutor*, která vystupuje jako producent dat a výpočtová vlákna pak tato data konzumují. Jedná se tedy o synchronizační problém producent-konzument.

Jednotlivé kanály jsou reprezentovány objekty třídy *ChannelProcessor,* které obsahují metody pro vykonávání a zpracování přijatých dat. Reference na tyto objekty jsou po rozdělení dat řazeny do blokující fronty, ze které je postupně odebírají dedikovaná výpočetní vlákna, jež nad objekty pomocí unifikovaného rozhraní vykonávají požadované výpočty a umožňují zápis výsledků do externího úložiště zařízení.

Základním modulem pro zpracování signálů je tedy třída *ChannelProcessor*, resp. odvozená třída *MultiChannelProcessor,* které obalují logiku pro zpracování jednoho resp. několika měřených kanálů. Výpočetní část je pomocí těchto objektů naprosto oddělena od zbytku aplikace a nové druhy kanálů je tak možné definovat pouze úpravou nebo děděním od těchto tříd bez nutnosti zasahovat do jiné části aplikace s výjimkou uživatelského rozhraní, kde je podle typu kanálu možné dále individualizovat formu vizualizace dat. Obě zmíněné třídy očekávají na vstupu pole vstupních dat z jednoho nebo několika kanálů a na svém výstupu pak nabízejí výsledné extrahované parametry dle zvoleného typu signálu.

6.5.2 Typy kanálů a jejich zpracování

Elementární třídou, která obsahuje konkrétní předpis zpracování příchozích dat je třída *ChannelProcessor*. Ta pak sama obsahuje několik pomocích tříd pro číslicovou filtraci, segmentaci signálů a následné zpracování tak, jak je znázorněno vyznačenými bloky na obrázku č. 15. První blok pro číslicovou filtraci realizuje programově třída *FIRFilter*, která obsahuje metody pro syntézu číslicového filtru dle definovaných pásem a pro filtraci pomocí metody sčítání přesahů. Filtraci je také možné pro urychlení zpracování dat obejít v případě, že ji daný typ kanálu nevyžaduje a nedochází k podvzorkování signálu. Následuje blok pro podvzorkování a segmentaci signálu, který zajišťuje zejména přípravu segmentů o zvolené délce pro další kroky zpracování daného signálu.

Krok zpracování signálu je programově realizován polem objektů odvozených ze třídy *DSPPipeElement.* Tyto objekty umožňují dále zapouzdřit použité výpočetní algoritmy, jež je poté možné libovolně kombinovat a řadit jako nezávislé bloky za sebe v libovolném pořadí. Různé typy kanálů tak mohou tyto výpočetní moduly sdílet mezi sebou a libovolně je kombinovat nebo rozšiřovat dle požadované funkce. Po zpracování dat dochází nakonec k zápisu výsledků do úložiště mobilního zařízení pomocí metod

přidělených objektů *SignalFileWriter* a dále k jejich šíření do uživatelského rozhraní pomocí definovaného rozhraní.

Díky této architektuře nabízí aplikace možnost jednoduchého rozšíření stávajících postupů nebo definování a zavedení zcela nových typů kanálů, případně i modulů pro analýzu signálů z několika různých fyzických kanálů.

V aplikaci byly na základě postupů z kapitoly č. 5 realizovány čtyři hlavní typy zpracování signálů. Prvním typem je zpracování EKG signálu pro výpočet srdeční frekvence, který kromě samotného algoritmu pro detekci QRS komplexu využívá též číslicovou filtraci a dle vzorkovací frekvence vstupního signálu také automatické podvzorkování a segmentaci signálu. Dalším typem je obecný statistický modul, který umožňuje z libovolného signálu vypočítat hodnoty minima, maxima, směrodatné odchylky, střední hodnoty a efektivní hodnoty jednotlivých segmentů signálu.

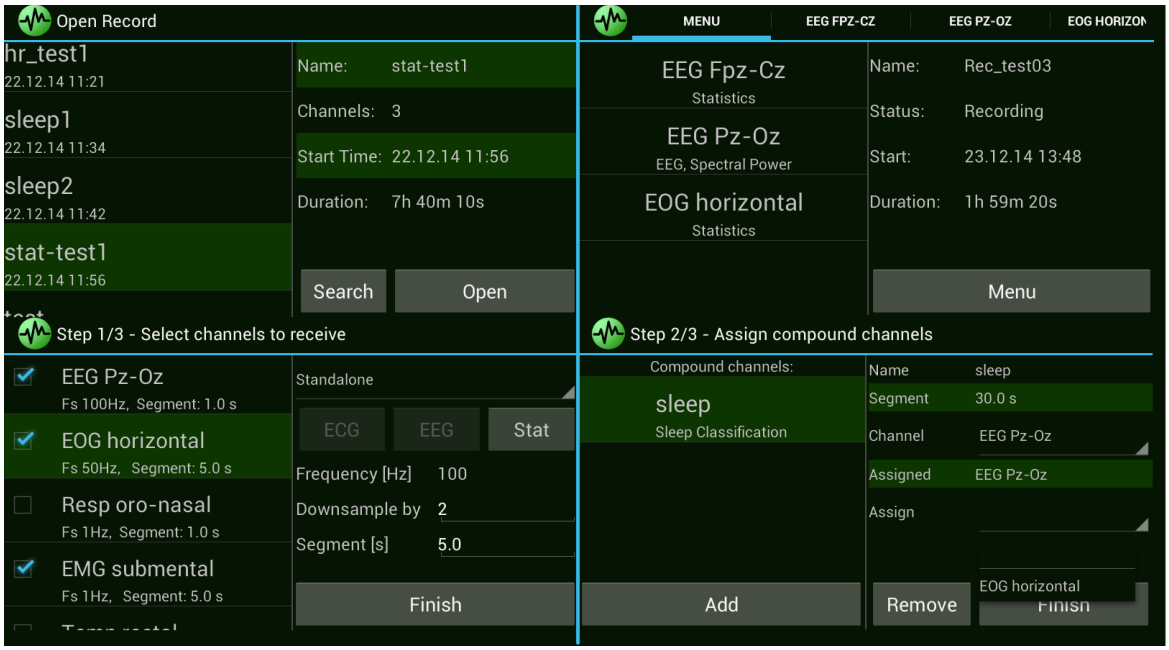
Poslední dva typy vycházejí z metod zpracování EEG a analýzy spánkových stádií. Pro zpracování EEG záznamů aplikace nabízí možnost analýzy ve frekvenční oblasti a výpočtu relativních výkonů definovaných frekvenčních pásem, které je navíc možné libovolně nastavit pomocí konfiguračního souboru aplikace, který se nachází v hlavním adresáři aplikace v úložišti mobilního zařízení. Aplikace nakonec také nabízí možnost analýzy spánkových stádií na základě kombinace jednoho EEG a jednoho EOG kanálu dle postupu v kapitole 5.3.3.

6.6 Grafické uživatelské rozhraní

6.6.1 Součásti uživatelského rozhraní

Uživatelské rozhraní aplikace tvoří několik aktivit a pomocných tříd jako jsou fragmenty, adaptéry seznamů nebo také XML soubory s definicemi statických rozvržení uživatelského rozhraní jednotlivých aktivit a položek seznamů.

Dle funkce lze uživatelské rozhraní zhruba rozdělit na část zodpovídající za procházení uložených záznamů, včetně jejich řazení, třídění a fulltextového hledání, dále na část pro správu vzdáleného zařízení, hlavně hledání zařízení v síti, procesu enumerace a nastavení kanálů a vlastností měření. Samostatný logickou část pak tvoří hlavní aktivita s vizualizačními fragmenty. Uživatelské rozhraní dále obsahuje několik dalších pomocných tříd, převážně různých uživatelským dialogů, sloužících ke vstupu dat od uživatele, případně k upozornění nebo potvrzení nějaké činnosti.



Obrázek 16: Ukázka uživatelského rozhraní pro výběr záznamu a přehled kanálů záznamu (nahoře) a ukázka uživatelského rozhraní během enumerace a nastavování nového měření (dole).

Obrázek č. 16 ukazuje vybrané aktivity uživatelského rozhraní, které slouží k ovládání aplikace. Vlevo nahoře je možné vidět ukázku rozhraní pro výběr záznamu z externího úložiště zařízení. Tato aktivita slouží k procházení uložených záznamů a umožňuje zobrazit základní informace o těchto záznamech. Součástí aktivity je také

dialog, pomocí kterého je možné jména záznamů fulltextově prohledávat podle zvolených slov nebo jejich částí a dále tyto záznamy řadit sestupně i vzestupně podle názvu nebo podle data pořízení záznamu. Po zvolení záznamu se otevře hlavní aktivita, na obrázku vpravo nahoře, která slouží ke zobrazení detailů o jednotlivých kanálech měření a která pomocí záložek v horní liště umožňuje zobrazení vizualizací dostupných kanálů.

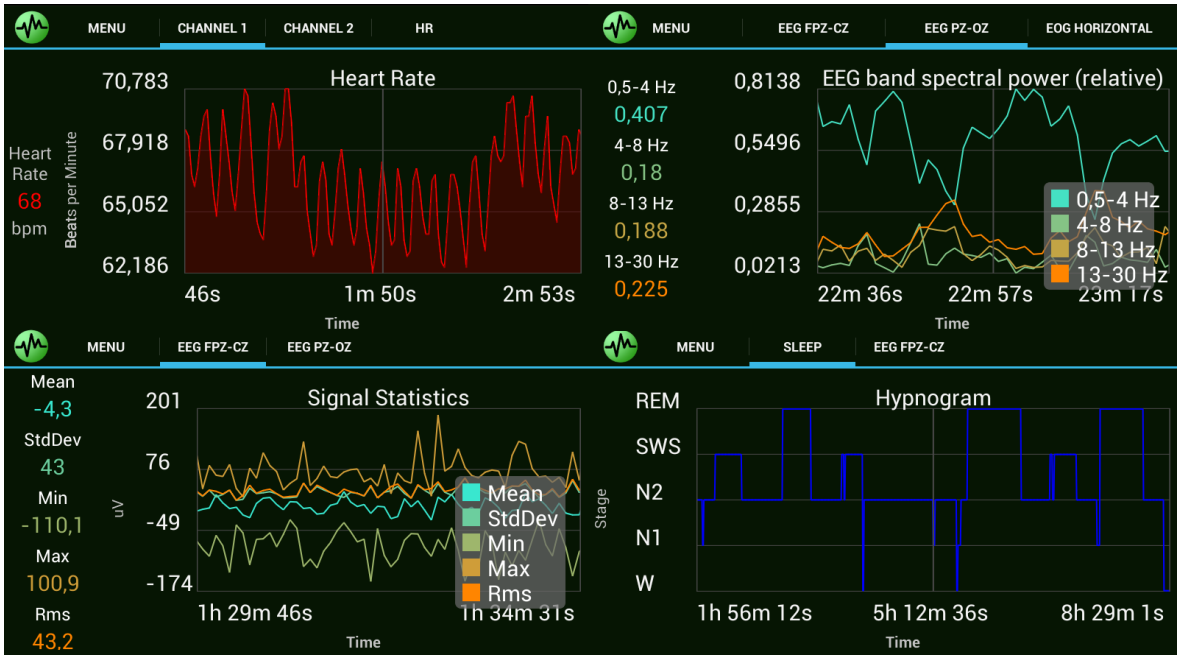
Spodní dvě aktivity z obrázku č. 16 pak zobrazují výběr a nastavení kanálů měření po předchozím úspěšném navázání spojení s bezdrátovým zařízením a vyčtením nabízených kanálů pomocí procesu enumerace. Aktivita vlevo dole slouží k výběru kanálů, které se mají v rámci měření přenášet a dále nabízí možnost nastavení typu a parametrů jednotlivých kanálů. Aktivita vpravo umožňuje zvolené kanály dále sdružovat do složených kanálů, přičemž v aplikace je tato možnost konkrétně využita pro analýzu spánku, která vyžaduje současné zpracování dvou odlišných kanálů.

Aplikace dále obsahuje několik pomocných dialogů sloužících jako upozornění uživateli nebo jako doplňkový vstup informace jako je například název měření nebo případně k potvrzení vybraných akcí uživatelem. Kromě fragmentů souvisejících s vizualizacemi signálů, které jsou popsány níže, obsahuje aplikace dále také například aktivitu pro hledání vzdálených zařízení v lokálních síti a jejich připojení nebo hlavní uvítací okno aplikace.

6.6.2 Vizualizace grafů

Hlavní snahou při implementaci vizualizací bylo, aby grafické rozhraní mohlo k různým typům dat v režimech online i offline prohlížení přistupovat pokud možno jednotným způsobem, ale aby zároveň bylo možné jednotlivé vizualizace lehce upravovat dle typu dat, tak aby uživatelské rozhraní bylo přehledné a umělo se přizpůsobit specifikům jednotlivých typů signálů.

Výsledkem bylo vytvoření rozhraní *DataViewInterface*, které slouží jako aplikační vrstva oddělující vnitřní části aplikace a grafické uživatelské rozhraní. Toto rozhraní implementují třídy LogReader a ApplicationExecutor, čímž dochází k sjednocení přístupu uživatelského rozhraní k datům pocházejícím z uloženého záznamu a datům, která pocházejí z výpočetních modulů právě probíhajícího záznamu.



Obrázek 17: Ukázka vizualizačních režimů aplikace.

Rozhraní kromě metod pro získávání dat obsahuje také metody, díky nimž je možné rozlišit režim záznamu nebo získat pole objektů popisujících jednotlivé kanály záznamu včetně jejich typů. Na základě těchto informací může uživatelské rozhraní zobrazovat a pracovat s daty unifikovaným způsobem, ale v době inicializace nebo zobrazení uživateli může zároveň dle typu záznamu upravovat vybrané grafické prvky zobrazovaného grafu.

Ukázky vizualizací jsou zobrazeny na obrázku č. 17. Hlavním rozdílem mezi vizualizací uloženého záznamu a právě zpracovávaných dat je přítomnost levé svislé informační lišty, která se v režimu online záznamu dynamicky aktualizuje a zobrazuje aktuální hodnoty jednotlivých parametrů vybraného kanálu. Graf samotný je možné také dynamicky aktualizovat, ale s ohledem na výpočetní náročnost přepsání dat a jejich překreslení je graf aktualizován vždy pouze na žádost uživatele po stisku záložky daného kanálu a kapacita grafu je navíc omezena pevně daný maximální počet vzorků. V režimu prohlížení uloženého záznamu je levá lišta neviditelná a graf zobrazuje všechna data, která jsou na daném kanálu k dispozici.

V obou režimech prohlížení je pak samozřejmosti funkce přiblížení na ose *x* a u grafů, kde je obtížné odhadnout rozsah osy *y,* je tato funkce povolena implicitně u obou os. Všechny grafy pak podporují posuv v grafu v obou osách. Tyto funkce jsou uživateli zprostředkovány pomocí gest přímo v okně grafu, s výjimkou přiblížení osy *y*, které se

ovládá na ploše zobrazující popisky osy *y*. Posuv v grafu je ovládán pomocí gesta táhnutí prstu po ploše grafu, funkce přiblížení se pak ovládá pomocí gesta *pinch-to-zoom* a vyžaduje tedy displeje s podporou funkce *multitouch*.

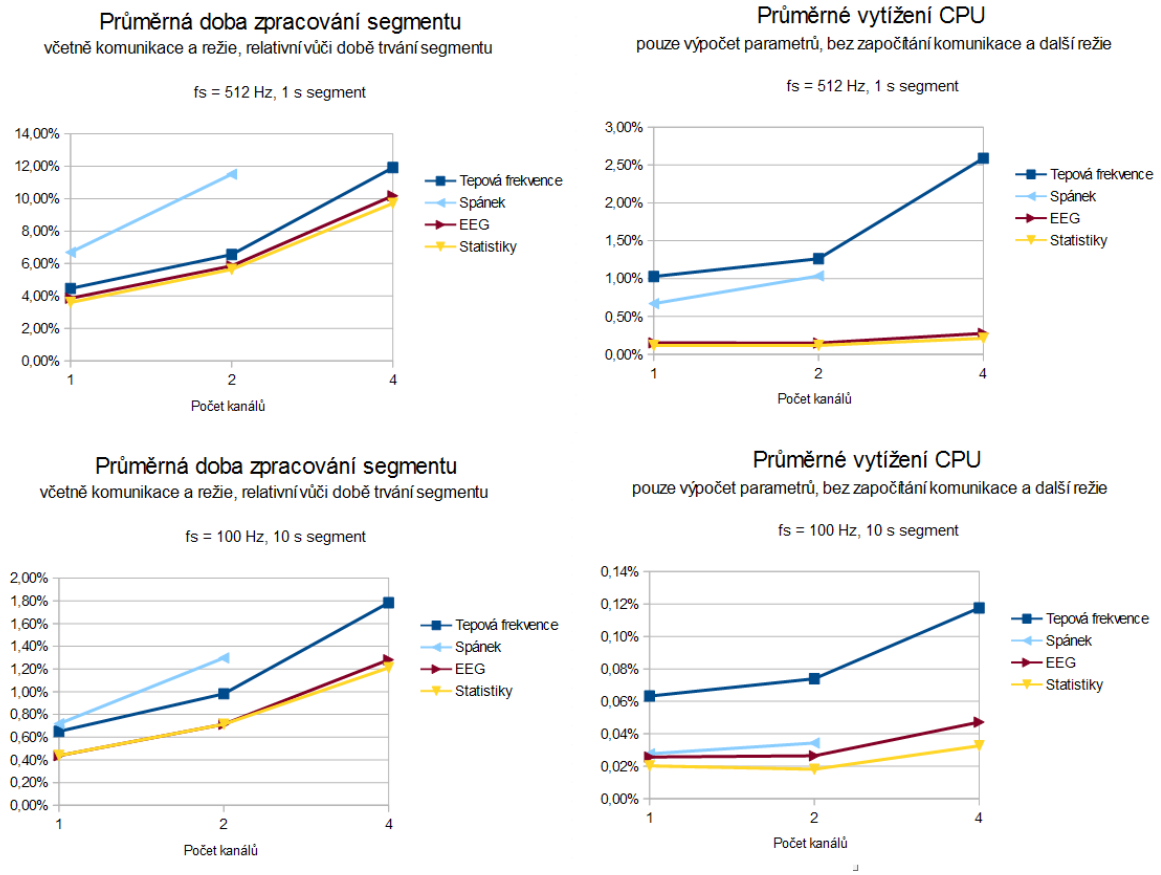
V obou režimech jsou dále dle typu záznamu drobné rozdíly v podobě grafického rozhraní. Dle specifik jednotlivých typů záznamu se mírně odlišuje nastavení rozsahu a pojmenování os, pojmenování grafu, zobrazení legendy, možnosti přiblížení nebo barvy grafu. Rozdílný je také obsah levé lišty, která se přizpůsobuje počtu signálů na daném kanálu a jež také dynamicky upravuje popisky a jednotky, tak aby bylo vždy zřetelné o jaké měření se jedná.

K vizualizaci grafu je využita open-source knihovna com.jjoe64.graphview ve verzi 3.1.4. Tato knihovna byla v rámci práce dále upravena tak, aby podporovala přiblížení a posuv na ose *y* a drobně bylo také upraveno formátování popisků této osy. Ke grafu byly dále ručně přidány popisky os *x* a *y*, které knihovna v aktuální stabilní verzi zatím také nenabízí. Takto upravený graf je i s levou lištou dynamicky inicializován a zaobalen do systémového GUI objektu *Fragment.* Jednotlivé fragmenty obsahující grafy všech kanálů daného měření jsou pak spolu s nultým fragmentem, který obsahuje souhrnné informace o záznamu a hlavní menu, přiřazeny záložkám, jež se zobrazují ve vrchní části hlavní aktivity, díky kterým je tedy možné mezi hlavním menu a grafy přepínat.

7 Zhodnocení výsledků a výkonu aplikace

Kromě ověření chování a správnosti navržených postupů pomocí implementace a kontroly použitých algoritmů v prostředí MATLAB byl na vybraných datech orientačně otestován i výkon aplikace pod systémem Android. Aplikace byla testována na zařízení s procesorem ARM Cortex-A9 se dvěma výpočetními jádry a frekvencí procesoru 1 GHz.

Pro test byly zvoleny dva typy zdrojových signálů, první o vzorkovací frekvenci 512 Hz segmentovaný po 1 sekundě a druhý o vzorkovací frekvenci 100 Hz segmentovaný po 10 sekundách. Tyto dva typy signálů byly použity postupně jako zdroj pro jeden, dva a čtyři kanály a takto byly testovány všechny typy kanálů podporované aplikací. Jistou výjimku představuje testování klasifikátoru spánku, který je možné testovat vždy pouze na sudém počtu kanálů a který byl tedy otestován pouze pro dva a čtyři zdrojové kanály.



Obrázek 18: Vytížení systému a CPU při zpracování vybraných signálů dle počtu kanálů.

Při těchto testech pak byl měřen průměrný čas strávený výpočty na všech výpočetních vláknech a celkový čas zpracování včetně komunikační a systémové režie a započítána navíc byla také doba strávená synchronizací mezi vzdáleným zařízením a aplikací včetně prodlev, které vznikají mezi odesíláním datových paketů. Další známou hodnotou byl také skutečný čas záznamu, který by v režimu reálného času odpovídal také době měření. Testovací měření bylo prováděno v asynchronním dotazovacím režimu, který sice celkově může data zpracovávat průměrně rychleji, ale zavádí oproti režimu reálného času o něco vyšší komunikační režii a navíc dodatečné zpoždění.

Z popsaných hodnot bylo možné vyhodnotit podíl doby zpracování signálů vůči celkovému reálnému trvání záznamu, včetně veškeré komunikační režie a nutnosti čekání na nová příchozí data, a dále také průměrné využití výpočetní kapacity jednoho jádra procesoru při výpočtech, bez uvedené režie, ale s možným zkreslením kvůli přerušení výpočtu systémovými vlákny s vyšší prioritou. Dosažené výsledky jsou shrnuty v grafech na obrázku č. 18.

Z grafů je patrné, že nejnáročnějším kanálem na zpracování je výpočet srdeční frekvence, který v aplikaci využívá číslicovou filtraci a decimaci signálu, ze kterého je poté počítána první a druhá diference, se kterými algoritmus dále pracuje a porovnává je s dynamicky nastavovaným prahem. Přesto ale výpočet bez započítání režie zabral pouze zhruba 3 % výkonu použitého procesoru. Naopak nejméně náročnou metodou je extrakce obecných statistických parametrů, která se děje přímo v časové oblasti bez použití filtrace a která nezabrala ani 0,5 % výkonu procesoru.

Dále je vidět, že redukce vzorkovací frekvence spolu s redukcí objemu dat výrazně snižuje spotřebu výpočetních prostředků a spolu s prodloužením segmentů přispívá k omezení výpočetního výkonu spotřebovaného na režii a komunikační protokol. V oblasti zpracování dat je také vidět, že aplikace umí efektivně využívat dostupná výpočetní jádra, a že průměrná zátěž jednoho výpočetního jádra při výpočtech s jedním nebo dvěma kanály neroste skoro vůbec nebo kvůli vlivu systémové režie alespoň znatelně méně než dvojnásobně. Při výpočtu se čtyřmi kanály je již znát, že kód se vykonával pouze na dvoujádrovém procesoru a vytížení výpočetního vlákna tak již roste lineárně s počtem kanálů.

Poměrně nepříznivě pak vyšla celková rychlost zpracování záznamu včetně komunikací a režie, kde je z grafů patrné, že úzkým místem aplikace je synchronizace mezi zařízeními v dotazovacím režimu a dále také komunikační protokol, který je schopný data zpracovávat pouze na jednom procesorovém jádře.

Navržená architektura je tedy z hlediska výpočetní části zřejmě navržena poměrně efektivně a další vylepšení by tak mohlo směřovat hlavně cestou zefektivnění komunikačního protokolu se snahou co nejdříve rozdělit příjem a zpracování dat na dostupná procesorová jádra. V režimu zpracování v reálném čase navíc odpadá režie spojená se synchronizací a čekáním na nové datové pakety a při komunikaci s reálným bezdrátovým zařízením tak lze očekávat významné snížení režie i při zachování stávajícího protokolu.

Celkově výsledky naznačují značné výpočetní rezervy aplikace na platformě OS Android a možnost tak do ní zařadit složitější algoritmy nebo podporu více kanálu o vyšších vzorkovacích frekvencích. Aplikace je také díky paralelnímu návrhu modulů pro výpočet parametrů signálů schopná plně využít dostupných procesorových jader a nabízí proto možnost dalšího škálování nejen pomocí zvyšování systémové frekvence, ale také pomocí zvyšování počtu jader procesoru.

8 Závěr

Tato práce popisuje návrh a implementaci aplikace pro operační systém Android, která slouží ke zpracování vybraných biomedicínských signálů a extrakci jejich významných parametrů. Všechny metody použité v této aplikaci jsou navrženy s ohledem na práci v reálném čase na předem definovaných segmentech dat. Výsledky použitých metod jsou tak uživateli zprostředkovávány průběžně po dobu trvání měření formou jasně srozumitelných číselných výstupů a grafických vizualizací průběhů signálů. Data jsou zároveň ukládána do nevolatilní paměti mobilního zařízení pro pozdější prohlížení nebo další analýzu dat.

Konkrétně byly v aplikaci implementovány metody pro výpočet srdeční frekvence ze signálu EKG a výpočet dalších vybraných obecných statistických parametrů nebo frekvenčních vlastností signálu. Navržen byl také klasifikátor pro analýzu stádií spánku na základě průběhů jednoho kanálu EEG a EOG, případně s možností doplnění o analýzu EMG signálu z elektrod umístěných na bradě. Ke všem uvedeným metodám byl zároveň implementován individuální způsob vizualizace dat s vhodným zobrazením daného typu výsledků formou grafu a s možností zobrazení výsledných hodnot v reálném čase v průběhu celého měření.

Klasifikátor pro analýzu stádií spánku popsaný v této práci byl naučen a otestován na reálných klinických datech ze dvou vybraných spánkových studií, přičemž učení a vyhodnocení výsledků probíhalo odděleně na dvou nezávislých množinách záznamů. Klasifikátor dosáhl během testování bez kanálu EMG průměrné přesnosti 73,87 %, přičemž přesnost klasifikace stádií spánku N2, SWS a REM dokonce průměrně přesahovala 77 %. Horších výsledků klasifikátor dosáhl u stádia N1, kde přesnost průměrně mírně přesahovala 40 %, u stádia bdělosti se pak shoda pohybovala průměrně okolo 64 %. Zařazením EMG kanálu se celková úspěšnost klasifikace mírně zhoršila s průměrnou mírou shody 73,37 %, nicméně došlo alespoň k dílčímu vylepšení výsledků v rámci stádií bdělosti a spánku REM. Zajímavé jsou i výsledky klasifikátoru založené pouze na použití samostatného kanálu EEG, kdy průměrná přesnost klasifikace dosáhla 70,61 %. Vzhledem k tomu, že míra shody klasifikace záznamu spánku může i mezi lidskými experty kolísat až v řádu desítek procent a že je navržený klasifikační algoritmus navíc schopný pracovat v reálném čase, se tak celkově jedná o poměrně dobré výsledky.

Architektura aplikace byla navrhována tak, aby zohledňovala požadavek práce v reálném čase a vzhledem k cílové platformě proto obsahuje postupy pro vícevláknové zpracování signálů. Zároveň byla navrhována se snahou maximalizovat modularitu řešení pomocí využití sady předem definovaných abstraktních tříd a rozhraní a dalších možností objektového programování. Výsledkem je možnost poměrně snadno rozšířit aplikaci o další metody zpracování signálů, kterými by mohly být například výpočet dechové frekvence, detekce spánkové apnoe, metody pro EEG Biofeedback nebo BCI (Brain Computer Interface), analýza multisvodového EMG a podobně.

Součástí práce jsou dále skripty pro prostředí MATLAB, které slouží k učení a také testování přesnosti použitého klasifikátoru stádií spánku dle zadaných kritérií. V průběhu práce vznikaly také další pomocné skripty k testování metod zpracování signálů popsaných v této práci včetně algoritmu pro výpočet srdeční frekvence, skriptu pro návrh číslicového filtru pomocí metody oken nebo filtrace metodou sčítání přesahů.

Součástí práce je navíc definice použitého komunikačního protokolu s funkcemi automatického hledání kompatibilních zařízení v síti, enumerace, výběrem kanálů připojeného zařízení a samozřejmě také s funkcí přenosu samotných měřených dat v režimu reálného času nebo v režimu s aktivním dotazováním, který umožňuje efektivně využít výpočetní kapacity aplikace při zpracování offline signálů.

V rámci práce také vznikla nezávislá aplikace pro PC, která emuluje funkci bezdrátového zařízení nebo senzoru. Tato grafická aplikace byla realizována v jazyce Java a slouží hlavní aplikaci ke zprostředkování zdrojových klinických signálů uložených ve formátu EDF za využití bezdrátové sítě Wi-Fi a metod navrženého komunikačního protokolu. Komunikační protokol a hlavní aplikace jsou navíc navrženy tak, aby mohl být v budoucnu tento emulovaný zdroj dat snadno nahrazen reálným vlastnoručně navrženým nebo komerčním zařízením.

Literatura

[1] JANČÍK, Jiří, ZÁVODNÁ, Eva a NOVOTNÁ, Martina. *Fyziologie tělesné zátěže – vybrané kapitoly: 5.1. Kardiovaskulární soustava* [online]. [cit. 2014-07-02]. Dostupné z: http://is.muni.cz/elportal/estud/fsps/js07/fyzio/texty/ch05s01.html

[2] SMĚLÝ, Tomáš. *Klasifikace signálu EKG.* Brno, 2009. Diplomová práce. VUT v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství.

[3] PALANIAPPAN, Ramaswamy. *Biological Signal Analysis*. London: Ramaswamy Palaniappan & Ventus Publishing ApS, 2010. ISBN 978-87-7681-594-3.

[4] TOMPKINS, Willis J. *Biomedical digital signal processing: C-language examples and laboratory experiments for the IBM PC.* New Delhi: Prentice Hall of India, 2000. ISBN 81-203-1478-6.

[5] MANOLAKIS, Dimitris G. a INGLE, Vinay K. *Applied digital signal processing: theory and practice.* New York: Cambridge University Press, 2011. ISBN 05-211-1002-5.

[6] GERLA, Václav. *Automated Analysis of Long-Term EEG Signals*. Praha, 2012. Disertační práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra kybernetiky.

[7] KEMP, Bob, VARRI, Alpo, ROSA, Agostinho C., NIELSEN, Kim D. a GADE, John. A simple format for exchange of digitized polygraphic recordings. *Electroencephalography and Clinical Neurophysiology*. 1992, č. 82, s. 391-393.

[8] SILBER, Michael H., ANCOLI-ISRAEL, Sonia, BONNET, Michael H., CHOKROVERTY, Sudhansu, GRIGG-DAMBERGER, Madeleine M., HIRSHOWITZ, Max, KAPEN, Sheldon, KEENAN, Sharon A., KRYGER, Meir H., PENZEL, Thomas, PRESSMAN, Mark R. a IBER, Conrad. The Visual Scoring of Sleep in Adults. *Journal of Clinical Sleep Medicine.* 2007, roč. 3, č. 2, s. 121-131.

[9] ESTRADA, E., NAZERAN, H., BARRAGAN, J., BURK, J. R., LUCAS, E. A. a BEHBEHANI, K. EOG and EMG: two important switches in automatic sleep stage classification. In: *Proceedings of the 28th IEEE EMBS Annual International Conference*. New York City, 2006, s. 2458-2461.

[10] KUO, Chih-En, LIANG, Sheng-Fu, LI, Yi-Chieh, CHERNG, Fu-Yin, LIN, Wen-Chieh, CHEN, Peng-Yu, LIU, Yen-Chen a SHAW, Fu-Zen. *An EOG-based Sleep Monitoring System and Its Application on On-line Sleep-Stage Sensitive Light Control.* Taiwan, 2014. National Cheng Kung University, Tainan, Taiwan a National Chiao Tung University, Hsinchu, Taiwan. Dostupné také z: http://gpl.cs.nctu.edu.tw/liyij/Yi-Chieh_Lee/Project_files/paper.pdf.

[11] BERTHOMIER, C., DROUOT, X., HERMAN-STOICA, M., BERTHOMIER, P., PRADO, J., BOKAR-THIRÉ, D., BENOIT, O., MATTOUT, J. a D'ORTHO, M. P. Automatic Analysis of Single-Channel Sleep EEG: Validation in Healthy Individuals. *Sleep.* 2007, roč. 30, č. 11, s. 1587-1595.

[12] KERKENI, Nizar, ALEXANDRE, Frederic, BEDOUI, Mohamed, BOUGRAIN, Laurent a DOGUI, Mohamed. Automatic classification of Sleep Stages on a EEG signal by Artificial Neural Networks. In: *5th, WSEAS International Conference on SIGNAL, SPEECH and IMAGE PROCESSING.* Corfu, Řecko, 2005.

[13] GERLA, Vaclav, DJORDJEVIC, Vladana, LHOTSKA, Lenka a KRAJCA, Vladimir. PSGLab Matlab toolbox for polysomnographic data processing: Development and practical application. In: *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*. IEEE, 2010, s. 1-4. ISBN 978-1-4244-6559-0. DOI: 10.1109/ITAB.2010.5687737. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5687737

[14] FLEXER, Arthur, GRUBER, Georg a DORFFNER,Georg. A reliable probabilistic sleep stager based on a single EEG signal. *Artificial Intelligence in Medicine*. 2005, roč. 33, č, 3, s. 199-207. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S093336570400079X

[15] GOLDBERGER, A. L., AMARAL, L., GLASS, L., HAUSDORFF, J. M., IVANOV, P. Ch., MARK, R. G., MIETUS, J. E., MOODY, G. B., PENG, C-K. a STANLEY, H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation.* 2000, roč. 101, č. 23, s. 215-220. Dostupné z: http://circ.ahajournals.org/cgi/content/full/101/23/e215]

[16] RECHTSCHAFFEN, Alan a KALES, Anthony [eds.]. *A manual of standardized terminology, techniques and scoring systems for sleep stages of human subjects*. Washington DC: US Government Printing Office, National Instituteof Health Publication, 1968.

[17] IBER, Conrad, ANCOLI-ISRAEL, Sonia, CHESSON, Andrew a QUAN, Stuart F. *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications.* Westchester, IL: American Academy of Sleep Medicine, 2007.

[18] WATIER, Nicholas N., LAMONTAGNE, Claude a CHARTIER, Sylvain. What does the mean mean?. *Journal of Statistics Education* [online]. 2011, roč. 19, č. 2 [cit. 2014-11-13]. Dostupné z: http://www.amstat.org/publications/jse/v19n2/watier.pdf

[19] GREENFIELD, L. John. *Reading EEGs: a practical approach*. Philadelphia, PA: Lippincott Williams, 2010. ISBN 07-817-9344-0

[20] COLLURA, Thomas F. History and Evolution of Electroencephalographic Instruments and Techniques. *Journal of Clinical Neurophysiology.* 1993, roč. 10, č. 4, s. 476-504. DOI: 10.1097/00004691-199310000-00007. Dostupné z: http://content.wkhealth.com/linkback/openurl?sid=WKPTLP:landingpage

[21] *The McGill Physiology Virtual Lab: Biological Signals Acquisition* [online]. 2014 [cit. 2014-11-15]. Dostupné z: http://www.medicine.mcgill.ca/physio/vlab/other_exps/EOG/EOGintro_n.htm

[22] POKORNÝ, Jan. *Evokované potenciály* [online]. ČVUT v Praze, Fakulta biomedicínského inženýrství, 2014 [cit. 2014-11-15]. Dostupné z: http://fbmi.cvut.cz/files/nodes/657/public/EVOKOVAN%C3%89%20POTENCI%C3%81LY.pdf

[23] SANEI, Saeid a CHAMBERS, Jonathon. *EEG signal processing*. Hoboken, NJ: John Wiley, 2007. ISBN 04-700-2581-6.

[24] MOSER, D., ANDERER, P., GRUBER, G., PARAPATICS, S., LORETZ, E., BOECK, M., KLOESCH, G., HELLER, E., SCHMIDT, A., DANKER-HOPFE, H., SALETU, B., ZEITLHOFER, J. a DORFFNER, G. Sleep classification according to AASM and Rechtschaffen & Kales: effects on sleep scoring parameters. *Sleep.* 2009, roč. 32, č. 2, str. 139-149.

[25] NORMAN, Robert G., PAL, Ivan, STEWART, Chip, WALSLEBEN, Joyce A. a RAPOPORT, David. Interobserver Agreement Among Sleep Scorers From Different Centers in a Large Dataset. *Sleep.* 2000, roč. 23, č. 7.

[26] DIMARZIO, J. *Android: a programmer's guide*. New York: McGraw-Hill, 2008. ISBN 978-007-1599-887. Dostupné z: http://www.e-reading.link/bookreader.php/142063/Android_-_a_programmers_guide.pdf

[27] GIBSON, Michael C. QRS Complex. *WikiDoc* [online]. 2014 [cit. 2014-12-30]. Dostupné z: http://www.wikidoc.org/index.php/QRS_complex

A Příloha 1 - Definice pásem a klasifikačních matic použitých při klasifikaci spánku

```
float[][][] cmptable =
{ // class – sleep vs wake
  { { 1.3142f, 1.166f, 1.5582f, 1.622f, 1.1947f, 1.1525f, 0.54356f, 1.0873f,
      2.5419f, 0.89806f },
    { 1.677f, 1.8684f, 1.1395f, 2.98f, 1.361f, 1.1857f, 0.58083f, 0.60014f,
      2.1953f, 0.88193f }
  }, // class N2, SWS
  { { 1.4647f, 1.1174f, 1.8769f, 0.65786f, 0.68633f, 1.2553f, 1.6615f, 1.3169f,
      1.2873f },
    { 2.298f, 1.4235f, 2.2032f, 0.37609f, 0.45961f, 1.6097f, 1.946f, 1.5447f,
      1.4756f }
  },
  { // class wake, REM, N1
    { 1.5341f, 0.89767f, 1.2179f, 1.0455f, 1.3681f, 0.40355f, 0.99142f, 1.6325f,
      1.8907f, 1.4533f },
    { 1.2677f, 1.2368f, 1.4089f, 1.3014f, 1.6501f, 0.67019f, 1.4217f, 1.6307f,
      1.6509f, 1.2985f },
    { 1.2214f, 1.3885f, 1.446f, 1.3891f, 1.6554f, 0.59442f, 1.1117f, 1.5581f,
      1.8289f, 1.4069f }
  }
};

float[][] eegbands =
{ // in Hz
  { 0.5f, 3f, 7f, 11f, 16f, 20f, 24f, 28f },
  { 0.5f, 3f, 7f, 14f, 24f },
  { 0.5f, 3f, 7f, 11f, 17f, 24f, 30f }
};

float[][] eogbands =
{ // in Hz
  { 0.1f, 0.3f, 0.7f, 2f, 3f, 5f },
  { 0.1f, 0.3f, 0.7f, 2f, 4f, 6f, 8f, 10f },
  { 0.1f, 0.3f, 0.7f, 2f, 4f, 6f, 8f }
};
```

B Příloha 2 - Obsah přiloženého CD

- Adresář *text* obsahuje text této diplomové práce a adresář s obrázky a diagramy, které byly v práci použity
- Adresář *skripty* obsahuje skripty pro MATLAB, které byly využity při testování a implementaci vybraných metod číslicového zpracování signálů, a skripty pro učení a vyhodnocení klasifikátoru stádií spánku použitého v této práci. V adresáři je také podadresář s EDF soubory použitými jako zdroj dat pro testování aplikace a spánkového klasifikátoru.
- Adresář *workspace* je kopií pracovního adresáře pro vývojové prostředí Eclipse pro Android a obsahuje zdrojové kódy serverové i klientské aplikace navržené v této práci a dále zdrojové kódy nebo balíčky s externími knihovnami, které byli těmito aplikacemi použity. Adresář také obsahuje spustitelný archiv typu jar pro spuštění serverové aplikace a instalační balíček pro Android s příponou APK, sloužící k instalaci navržené aplikace na zařízení s OS Android.