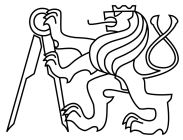
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Nástroj pro zpracování EEG a využití při interakci s aplikacemi |
| **Student:** | Bc. Filip Munzar |
| **Vedoucí:** | Ing. Jaroslav Kuchař |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2016/17 |

## Pokyny pro vypracování

Moderní přístupy jako klasifikace emocí na bázi EEG poskytují širší spektrum nápověd o zájmu a motivaci uživatele než metody založené na zachytávání akcí (kliknutí na objekt, otevření stránky). Cílem diplomové práce je vytvořit aplikaci, která bude získávat signály z EEG senzoru a využívat při interakci se zvolenou aplikací.

- Seznamte se s zařízením pro poskytování EEG – Emotiv EPOC a prostudujte možnosti jeho API.
- Navrhněte, implementujte a otestujte aplikaci v jazyce Java pro komunikaci se zařízením. Aplikace bude umožňovat především záznam čistého EEG a předzpracovaných signálů.
- Navrhněte a implementujte nástroj pro převod záznamů do formátu vhodného pro simulátor EmoComposer. Vytvořte sadu testovacích dat.
- Navrhněte a implementujte rozhraní/architekturu umožňující vzájemnou komunikaci s externími aplikacemi (video přehrávač, prezentace, ...).
- Proveďte experiment se zvolenou aplikací, která bude využívat signály pro odhad zájmu uživatele o informace prezentované aplikací.

## Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 2. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

# Nástroj pro zpracování EEG a využití při interakci s aplikacemi

***Bc. Filip Munzar***

Vedoucí práce: Ing. Jaroslav Kuchař

8. května 2016

# Poděkování

V první řadě děkuji Ing. Jaroslavu Kuchařovi za cenné rady a trpělivost při vedení této práce. Také bych rád poděkoval fakultě informatiky a statistiky Vysoké školy ekonomické v Praze za zapůjčení zařízení Emotiv EPOC, bez kterého by tato práce nemohla vzniknout.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2016 .....

### Odkaz na tuto práci

Munzar, Filip. *0.0.0.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

# Abstrakt

Tato diplomová práce se zabývá návrhem a implementací knihovny umožňující spolupráci s neuroheadsetem Emotiv EPOC a dvou dalších aplikací, které tuto knihovnu budou využívat. Emotiv EPOC je náhlavní souprava umožňující monitorování mozkové aktivity. Práce poskytne stručný úvod do problematiky ovládání počítače pomocí lidských myšlenek. Je zde popsána metoda elekroencefalografie, dále je zde popis Brain-Computer Interface zařízení, tedy zařízení sloužící pro propojení mozku s počítačem a čtenář zde nalezne i charakteristiku přístroje Emotiv EPOC. Hlavní náplní práce je návrh, implementace a testování knihovny, aplikace pro záznam dat získaných z neuroheadsetu a nástroje, který bude určen pro jednoduché ovládání externích aplikací pomocí Emotiv EPOC. Práce je doplněna i o instalační a uživatelskou příručku.

**Klíčová slova**

Elektroencefalografie, EEG, Emotiv EPOC, ovládání počítače pomocí myšlenek, brain-computer interface, BCI

# Abstract

This master thesis describes the design and implementation of library for cooperation with Emotiv EPOC neuroheadset and two applications that will use this library. Emotiv EPOC neuroheadset allows monitoring brain activity. The thesis gives a brief introduction to control a computer using human thoughts. There is described the Electroencephalography, the Brain-Computer Interface, a device used to connect the brain with a computer and finally there is the characteristic of Emotiv EPOC device. The core of this thesis is design, implementation and testing of library and application to recording data acquired usign Emotiv EPOC and tool for basic control of external applications with Emotic EPOC. The work is supplemented by installation and user manual.

**Keywords**

Electroencephalography, EEG, Emotiv EPOC, controlling computer using brain waves, brain-computer interface, BCI

# Obsah

**Úvod** **1**

**1 Mozková aktivita, její záznam a zpracování** **3**
1.1 Elektroencefalografie . . . . . 3
1.2 Brain-Computer Interface . . . . . 7

**2 Analýza a návrh** **11**
2.1 Emotiv EPOC Neuroheadset . . . . . 11
2.2 Model požadavků . . . . . 17
2.3 Návrh aplikací . . . . . 19

**3 Implementace** **31**
3.1 Emotiv EPOC Library . . . . . 31
3.2 Emotiv EPOC Logger . . . . . 36
3.3 EPOC Events Generator . . . . . 40
3.4 Shrnutí . . . . . 46

**4 Testování** **47**
4.1 Emotiv EPOC Library . . . . . 47
4.2 Emotiv EPOC Logger . . . . . 48
4.3 EPOC Events Generator . . . . . 50
4.4 Shrnutí . . . . . 53

**Závěr** **55**

**Literatura** **57**

**A Seznam použitých zkratek** **61**

**B Obsah přiloženého CD** **63**

**C Instalační a uživatelská příručka** **65**
C.1 Nástroje a SDK společnosti Emotiv . . . . . 65
C.2 Emotiv EPOC Library . . . . . 66
C.3 Emotiv EPOC Logger . . . . . 67
C.4 EPOC Events Generator . . . . . 68

**D Emotiv SDK** **71**
D.1 Události EmoEvents . . . . . 71
D.2 Chybové kódy . . . . . 72

**E Diagramy** **73**

**F XSD schéma** **75**

**G Zdrojové kódy** **77**

**H Testovací data a výstupy testování** **81**
H.1 Emotiv EPOC Logger . . . . . 82
H.2 EPOC Events Generator . . . . . 84

**I Ukázky video záznamů z testování** **89**

# Seznam obrázků


1.1 Rozmístění elektrod podle systému 10/20 . . . . . 4
1.2 Princip fungování BCI systému . . . . . 8

2.1 Zařízení NeuroSky MindWave . . . . . 12
2.2 Emotiv EPOC neuroheadset . . . . . 13
2.3 Rozložení elektrod zařízení Emotiv EPOC . . . . . 13
2.4 Emotiv Control Panel . . . . . 15
2.5 Aplikace EmoComposer . . . . . 15
2.6 Princip komunikace s EmoEngine . . . . . 18
2.7 Diagram tříd zajišťující vytvořeení požadovaného typu spojení . . 20
2.8 Diagram template metody pro získání dat ze zařízení . . . . . 21
2.9 Diagram tříd starající se o záznam dat . . . . . 22
2.10 Wireframe aplikace Emotiv EPOC Logger . . . . . 24
2.11 Generování a zpracování událostí nástrojem EPOC Events Generator 28

3.1 Grafické rozhraní aplikace Emotiv EPOC Logger . . . . . 37

4.1 Porovnání zachycených emocí aplikací Emotiv Control Panel a nástrojem EPOC Events Generator - simulovaná data . . . . . 51
4.2 Porovnání zachycených emocí aplikací Emotiv Control Panel a nástrojem EPOC Events Generator - reálná data . . . . . 52
4.3 Graf zachycující uživatelův zájem s vyznačenými úseky, které uživatele zaujaly . . . . . 53

C.1 4 kroky instalačního průvodce nástrojů společnosti Emotiv . . . . 66

E.1 Diagram zachycující třídy zajišťující komunikaci v aplikaci Emotiv EPOC Logger . . . . . 74

I.1 Kontrola správnosti informací o navázeném spojení v aplikace Emotiv EPOC Logger - připojení k EmoComposer . . . . . 90

I.2 Kontrola správnosti informací o navázeném spojení v aplikace Emotiv EPOC Logger - připojení k Emotiv EPOC . . . . . 91
I.3 Záznam EEG dat pomocí Emotiv EPOC Logger (v pozadí je soubor se zaznamenanými EEG daty) . . . . . 92
I.4 Záznam Emo skriptu pomocí Emotiv EPOC Logger (v pozadí je soubor se zaznamenaným Emo skriptem) . . . . . 93
I.5 Ovládání prezentace pomocí klávesových zkratek generovancýh nástroje EPOC Events Generator . . . . . 94
I.6 Odesílání zpráv aplikaci InBeat . . . . . 95

# Úvod

Není to tak dávno, co se k ovládání počítače používala výhradně myš a klávesnice. V současné době se ale čím dál více rozmáhají nové způsoby ovládání. Standardem se pomalu stávají dotykové monitory, díky různým senzorům je umožněno ovládání pomocí řeči nebo snímání pohybu. Můžeme tedy očekávat, že se začnou prosazovat další způsoby, jak ovládat počítač. Jedním z nich může být i ovládání počítače pomocí lidských myšlenek.

Lidská mysl je mocný nástroj a vědci se již mnoho let snaží o její čtení. První úspěšné pokusy se datují již do 20. let 20. století. Od té doby urazil výzkum a vývoj dlouhý kus cesty a dnes jsou různá zařízení umožňující vyšetření mozku pomocí záznamu jeho elektrické aktivity běžnou součástí vybavení větších nemocnic. Dokonce se objevují pokusy o ovládání robotů a protetických paží.

V posledních letech se navíc na trh dostávají zařízení, která jsou díky své relativně nízké ceně dostupná široké veřejnosti. Jedním z nich je zařízení EPOC od společnosti Emotiv. Zařízení je vybaveno 14 senzory pro snímání mozkové aktivity, díky čemuž je možné získat kvalitní záznamy. Hlavní výhoda Emotiv EPOC spočívá v tom, že je k dispozici sada funkcí a nástrojů (tzv. SDK), které umožňují vývoj různorodých aplikací využívající právě toto zařízení.

Cílem práce je návrh a implementace sady funkcí (knihovny), které budou sloužit ke komunikaci se zařízením Emotiv EPOC a budou usnadňovat práci s ním (navazování spojení, sběr dat apod.). Vytvořená knihovna pak bude využita v aplikaci pro záznam elektroencefalografických dat a předzpracovaných signálů, které poskytuje Emotiv EPOC. Elektroencefalografická, neboli EEG data, jsou číselným vyjádřením elektrické aktivity neuronů v mozku, zachycené elektrodami přístroje Emotiv EPOC. Mezi předzpracované signály řadíme základní emoce (nadšení, frustrace apod.) nebo výrazy obličeje (například úsměv, mrknutí nebo zavření očí), které dokáže zařízení z nasbíraných EEG dat odvodit.

Dále bude vytvořen nástroj (opět s využitím vytvořené knihovny), který

bude umožňovat interakci s externími aplikacemi (multimediální přehrávače, prezentace apod.) a poskytovat záznam emocí uživatele.

Kromě samotného návrhu a implementace výše zmíněných aplikací je součástí práce také analytická část. Tato část má za cíl stručně informovat čtenáře o metodách a technologiích, díky kterým je možné zaznamenávat mozkovou aktivitu. Tato část navíc obsahuje kapitolu, která se podrobněji věnuje charakteristice zařízení Emotiv EPOC.

Práce je rozdělena do 4 kapitol. První kapitola se věnuje stručnému úvodu do problematiky ovládání počítače pomocí lidských myšlenek. Je zde popsána metoda elektroencefalografie, která slouží pro monitorování mozkové aktivity. Kapitola také obsahuje sekci věnující se popisu zařízení pro spojení mozku a počítače.

V druhé kapitole naleznete charakteristiku zařízení Emotiv EPOC, seznam požadavků na implementovanou sadu aplikací a v poslední části této kapitoly je popsán návrh těchto aplikací.

Třetí kapitola se pak věnuje popisu implementace knihovny a aplikací Emotiv EPOC Logger a EPOC Events Generator.

Vytvořené aplikace bylo třeba otestovat. Způsob a výsledky testování jsou obsaženy v kapitole 4.

V příloze je k dispozici instalační a uživatelskou příručku, která uživatele provede instalací vytvořených nástrojů a pomůže při jejich používání.

Kapitola 1

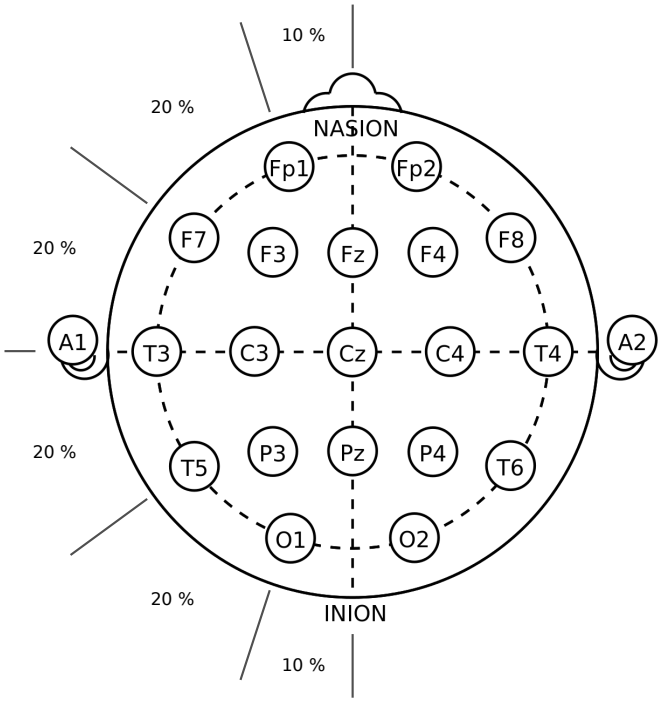
# Mozková aktivita, její záznam a zpracování

Tato kapitola poskytuje stručný úvod do problematiky ovládání počítače pomocí lidského mozku. Je zde popsána metoda elektroencefalografie sloužící pro záznam časových změn mozkových vln. Dále je zde charakteristika rozhraní BCI (Brain–Computer Interface) - tedy rozhraní propojující mozek s počítačem.

## 1.1 Elektroencefalografie

Elektroencefalografie (dále jen EEG) je základní elektrofyziologická metoda pro vyšetření mozkové aktivity[1]. Lidský mozek, konkrétně jeho neurony, vytvářejí elektrické napětí, které je možné zaznamenat[2, 3]. K tomuto účelu se používá přístroj zvaný elektroencefalograf. Jedná se o sadu elektrod, které vyžadují řádné rozmístění po povrchu hlavy[4]. Aby bylo vždy zajištěno stejné (nebo alespoň velice podobné) rozmístění elektrod napříč různými experimenty a bylo tedy možné výsledky těchto experimentů porovnávat, používá se k rozmístění mezinárodní standard označovaný jako 10/20 systém. Ten je založen na vztahu polohy elektrody a pod ní ležící oblasti mozkové kůry. Čísla 10 a 20 odkazují na deseti nebo dvaceti procentní vzdálenost mezi jednotlivými elektrodami (viz obrázek 1.1). Každá pozice senzoru je označena písmenem a číslicí ke snazší hemisferické orientaci. [5]

Písmena F/T/C/P/O označují Čelní (**F**rontal) / Spánkový (**T**emporal) / **C**entrální / Temenní (**P**arietal) a Týlní (**O**ccipital) lalok. Centrální mozkový lalok ve skutečnosti neexistuje, písmeno „C“ (**C**enter) zde slouží pro určení polohy uprostřed hlavy. Sudá čísla jsou přiřazena senzorům na pravé hemisféře, lichá čísla pak senzorům na levé hemisféře. Písmeno „Z“ (zástupný znak pro nulu, anglicky **z**ero) odkazuje na elektrody umístěné na střední linii hlavy. Systém navíc obsahuje dva referenční senzory A1 a A2. [1, 5, 8]




Obrázek 1.1: Rozmístění elektrod podle systému 10/20 (zdroj: [6, 7])

10/20 systém obsahuje 19 elektrod, ale existují i jiné systémy s 64 nebo dokonce 128 elektrodami.

### 1.1.1 Měření EEG signálu

Již v roce 1924 provedl první úspěšné snímání aktivity mozku německý psychiatr Hans Berger. První práci na toto téma ale publikoval až o 5 let později. Přestože se v této době našlo mnoho vědců, kteří výsledkům nevěřili, Berger potvrdil, že lidský nervový systém funguje na bázi elektrických výbojů, které je možné zaznamenávat, lidská pokožka je schopna tento signál přenášet a kůže na hlavě má schopnost reflektovat aktivitu mozku.

Měření mozkové aktivity zajišťuje systém elektrod. Tato technologie je velmi náročná na polohu elektrod a na jejich kontakt s pokožkou. K zajištění co nejlepších vodivých vlastností elektrod se používají materiály s dobrou vodivostí jako například zlato nebo chlorid stříbrný. Pro zlepšení kvality signálu se navíc používají různé vodivé gely nebo přípravky na bázi solného roztoku. Dalším problémem je fakt, že průchodem přes málo vodivou lebku je amplituda signálu zeslabena na úroveň řádově desítek mikrovoltů. Elektroencefalograf tedy musí zaznamenané signály nejprve zesílit a následně odfiltrovat případný šum způsobený rušivými artefakty (více v kapitole 1.1.3). Získaná data zazna-

menává do grafu, který se nazývá elektroencefalogram. [8, 9, 6, 10, 3]

### 1.1.2 Druhy mozkových vln

V EEG signálu vznikají vlny o různých vlnových délkách a různých amplitudách. Existuje pět různých druhů mozkových vln - alfa, beta, gamma, delta a théta. Každá mozková vlna má své vlastnosti, které představují určitý stav vědomí. Každá vlna je charakterizována pomocí frekvence, měřené v cyklech za sekundu a pomocí amplitudy (udává se v $\mu$V). Hodnota amplitudy vyjadřuje výšku vlny.

Každý typ mozkových vln hraje důležitou roli v našem mentálním vývoji během dětství. A když dospějeme, jsou důležité pro naše zdraví a vitalitu. V žádném případě nemůže nastat situace, kdy je vyzařována pouze jedna vlna. Často se ale stane, že na základě konkrétního emocionálního a vědomého stavu pozorovaného jedince právě jedna vlna převyšuje ostatní co se do počtu opakování a síly signálu týče. [11]

Následuje přehled pěti výše zmíněných typů mozkových vln a jejich stručná charakteristika.

#### 1.1.2.1 Delta vlny

Delta vlny se vyznačují frekvencí mezi 0.5 a 4 Hz a amplitudou nepřesahující 100 $\mu$V. Vlny delta jsou nejpomalejší z hlediska frekvence, mají však nejvyšší amplitudu. Této frekvence dosahujeme za hlubokého, bezesného spánku. Delta vlny jsou také dominantní u novorozenců do 24 měsíců věku. [11, 12, 13] Snížením aktivity delta vln roste bdělost, naopak nárůst aktivity navozuje únavu a pocit ospalosti. [8]

#### 1.1.2.2 Théta vlny

Frekvence théta vln se pohybuje v rozmezí 4–8 Hz s amplitudou mezi 20 a 100 $\mu$V. Vlny théta se aktivují během hluboké relaxace a meditace, lehkého spánku nebo snění, včetně REM fáze spánku.

U většiny dětí a dospívajících jsou mozkové vlny théta dominantní. [11, 13]

#### 1.1.2.3 Alfa vlny

Alfa vlny jsou přítomny během bdělého odpočinku se zavřenýma očima. Při otevření očí dochází k potlačení alfa vln. Alfa vlny mají frekvenci v rozsahu od 8 do 13 Hz. Amplituda alfa vln se pohybuje mezi 30 a 50 $\mu$V. [11, 14, 13]

#### 1.1.2.4 Beta vlny

Mozkové vlny beta jsou spojeny s běžným, bdělým vědomím. Jejich frekvence je 13–30 Hz a amplituda je 5–30 $\mu$V. Souvisí se zvýšenou čilostí, logickým

myšlením, schopností řešit problémy, koncentrací, mentální aktivitou. Člověk, který aktivně konverzuje, sportuje nebo něco například prezentuje, se nachází ve stavu beta. Vyšší úrovně beta však vedou ke stresu, úzkosti a neklidu.

Většina lidí tráví většinu bdělého života ve stavu beta, není tedy divu, že mnozí z nás dnes žijí ve velkém stresu. Na druhou stranu je stav beta důležitý pro účinné fungování v rámci každodenního života. [11, 14, 12, 13]

#### 1.1.2.5 Gamma vlny

Vlny gamma jsou nejrychlejší frekvencí, na níž může mozek fungovat. Jejich frekvence se pohybuje v rozsahu od 36 do 44 Hz s amplitudou 3–5 $\mu$V. S převládajícími vlnami gamma máte pocit, že dokážete všechno. [14, 15, 13]

Gamma mozkové vlny byly zpočátku ignorovány, než se vynalezl digitální elektroencefalograf. Jelikož analogové měření nebylo schopno identifikovat oscilace vyšší než 25 Hz. Jeden z nejstarších záznamů byl z roku 1964, kdy byly snímací elektrody implantovány ve zrakovém kortexu u opic při vědomí. [16]

### 1.1.3 Artefakty

Kromě mozkových vln zachytí EEG i jiné rušivé vlny, které se označují jako artefakty. Jako artefakt můžeme označit jakoukoliv vlnu, která má původ jinde než ve zkoumané oblasti mozku.[8]

Artefakty dělíme do dvou skupin – biologické a technické.

#### 1.1.3.1 Biologické artefakty

Do této skupiny řadíme všechny fyziologické signály, které nemají původ v mozku. Jedná se o signály, které jsou spojené s pohyby těla nebo různými tělesnými projevy jako je například pocení. Nejčastějšími biologickými artefakty ze strany uživatele jsou:

- pohyby očí
- srdeční aktivita
- svalová aktivita

Biologické artefakty jsou nejčastějším zdrojem rušení a znečištění EEG signálu. Pro lepší eliminaci těchto artefaktů se proto využívají další senzory pro sledování EMG (vyšetření elektrických signálů vycházející ze svalstva) nebo EKG (snímání elektrické srdeční aktivity). Tyto artefakty lze jednoduše odfiltrovat pomocí automatického zpracování. [8, 17]

#### 1.1.3.2 Technické artefakty

Mezi technické artefakty patří jakékoliv elektrické ruchy z okolních elektronických přístrojů nebo i ze samotného elektroencefalografu díky vadné součástce, špatnému kontaktu senzorů s pokožkou hlavy a nebo kvůli vybití baterie přístroje.

Na rozdíl od biologických artefaktů je obtížné tyto signály správně odfiltrovat pomocí automatizovaného softwaru a jsou tedy častou příčinou špatného vyhodnocení získaných dat. Na druhou stranu jde technickým ruchům poměrně snadno zamezit. [8, 17]

## 1.2 Brain-Computer Interface

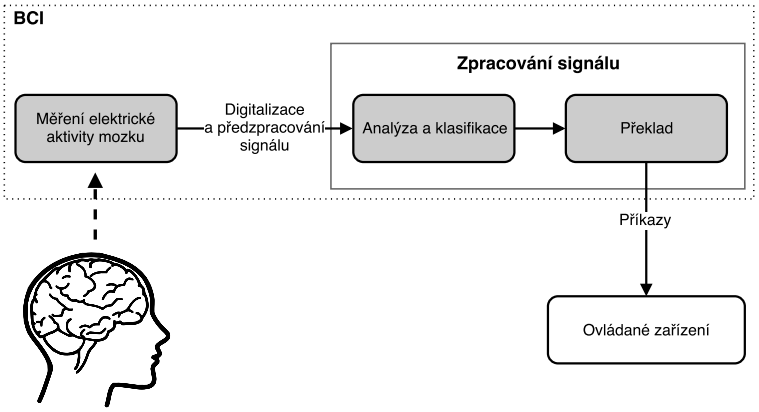
Brain-Computer Interface, zkráceně BCI, je systém, který zajišťuje přímé spojení mezi lidským mozkem a počítačem nebo jiným počítačovým zařízením. Pomocí BCI je možné ovládat připojené zařízení bez nutnosti využívat ovládání vyžadující svalovou aktivitu nebo ovládání pomocí řeči. BCI totiž generuje řídící signály z myšlenek uživatele. Díky tomu jsou tato zařízení využívána u lidí, kteří trpí nějakým druhem postižení pohybového systému. Kromě využití v medicíně lze BCI využít i v oblasti multimediální zábavy (ovládání počítačových her apod.).

Někdy se BCI zaměňuje s neuroprotetikou. V neuroprotetice však dochází ke spojení nervového systému se zařízením, které vykonává funkci protézy. Naproti tomu BCI realizuje spojení mezi centrálním nervovým systémem (mozkem) a počítačem. [18, 19]

BCI zařízení se obvykle skládá z hardwarové a softwarové části. Hardwarová komponenta zajišťuje monitorování a sběr elektrických signálů, které produkuje centrální nervová soustava, software pak vyhodnocuje zachycený signál a hledá signály s požadovanou charakteristikou. Tyto signály poté převádí na obslužné signály pro zařízení, které je pomocí BCI ovládáno. Celý proces, který obsluhuje BCI systém, je naznačen na obrázku 1.2. [20]

### 1.2.1 Historie

Počátek vývoje BCI zařízení se datuje do roku 1970. Jak už tomu bývá u podobných „experimentálních“ projektů, byl i vývoj BCI projektem americké armády. V roce 1998 byl poprvé pořízen záznam mozkových vln ve vysoké kvalitě. V letech 2002 až 2005 se vědci zabávali převážně pokusy na opicích. V roce 2002 byla opice schopna ovládat kurzor myši a v roce 2005 dokonce robotickou paži. V roce 2003 společnost BrainGate představila první veřejně dostupnou hru ovládanou pomocí mozkových vln. V roce 2003 (resp. 2004) na trh vstoupili společnosti Emotiv Systems a NeuroSky, které začali s vývojem BCI zařízení za cenu, díky které se staly BCI zařízení dostupná široké veřejnosti. [22, 10]




Obrázek 1.2: Princip fungování BCI systému (zdroj: [21])

### 1.2.2 Dělění BCI

BCI dělíme podle způsobu, jakým jsou zavedeny do lidského těla. Rozlišujeme 3 základní typy - invazivní, částečně invazivní a neinvazivní způsob. [10, 23, 20]

#### 1.2.2.1 Invazivní BCI

Jak už název napovídá, musí být invazivní BCI chirurgicky zavedeny přímo do mozku. Většinou se jedná o soustavu mikroelektrod, které jsou implantovány do motorického nebo vizuálního centra mozku. Vzhledem ke skutečnosti, že jsou elektrody zavedeny přímo v mozku, poskytuje tato metoda nejlepší kvalitu signálu. Na druhou stranu je ale nutná implantace cizího tělesa do lidského těla.

#### 1.2.2.2 Částečně invazivní BCI

Dalším typem je částečně invazivní BCI. U tohoto typu je také využívána soustava mikroelektrod. Na rozdíl od invazivní metody nejsou tyto elektrody zaváděny přímo do mozku, ale jsou zavedeny pod povrch lebky. Tento způsob využívá například Elektrokortikografie.

#### 1.2.2.3 Neinvazivní BCI

Posledním a nejvyužívanějším typem je neinvazivní BCI. Neinvazivní BCI využívá různé funkční zobrazovací metody, mezi které patří například EEG, Magnetoencefalografie (MEG) nebo funkční magnetická rezonance.

U neinvazivní metody jsou elektromagnetické aktivity mozku zaznamenávány skrze kůži i lebku. Tato metoda tedy nepotřebuje náročný chirurgický

zákrok. Její nevýhoda ale spočívá v omezené citlivosti elektromagnetických vln, a tak má oproti invazivnímu systému nižší rozlišovací schopnost.

Jedná se o převážně různé nasazovací čepice s citlivými senzory. Každý senzor se nachází nad jinou částí mozku.

Kapitola 2

# Analýza a návrh

Tato kapitola obsahuje informace o zařízení Emotiv EPOC, které bylo v rámci této práce použito pro snímání a záznam mozkových aktivit. Dále jsou zde stručně popsány požadavky na vytvářené aplikace a část věnující se jejich návrhu.

## 2.1 Emotiv EPOC Neuroheadset

### 2.1.1 Společnost Emotiv

Emotiv je společnost, která se zaměřuje především na vývoj a výrobu neurotechnologií založených na využití EEG. Společnost byla založena v roce 2003 v Austrálii. V současné době má své pobočky nebo laboratoře v USA, Vietnamu, Hongkongu nebo na Mauriciu.

V nabídce společnosti jsou dvě zařízení. Jedním je Emotiv EPOC resp. Emotiv EPOC+. Druhým zařízením je pak zařízení označované jako Emotiv Insight. Insight využívá pouze 5 senzorů a je určen pro monitorování mozkové činnosti během pohybových aktivit.

Obě výše zmíněná zařízení jsou bezdrátová a umožňují připojení jak k počítačům s operačními systémy Windows, Linux nebo Mac OS, tak i k zařízením s mobilními operačními systému iOS nebo Android. [24, 25]

### 2.1.2 Podobná zařízení

Vývojem zařízení založených na snímání mozkové aktivity se zabývá (zabývalo) mnoho společností. V současné době je ve stejné cenové hladině na trhu dostupné jediné zařízení. Tímto zařízením je MindWave (obrázek 2.1) od společnosti NeuroSky. Nutno ale podotknout, že společnost se primárně nezaměřuje na vývoj zařízení pro širokou veřejnost, ale cílí na vývoj technologií, které poté prodává svým partnerům.

Stejně jako Emotiv EPOC je i MindWave dostupné ve dvou variantách. První varianta je určena pro spolupráci s operačními systémy Windows, Linux a Mac OS. Druhá varianta, označovaná jako MindWave Mobile, je určena pro mobilní operační systémy iOS a Android.

Zařízení MindWave využívá jediný senzor umístěný na pozici FP1 dle systému 10/20. Z toho důvodu nedosahuje takových kvalit záznamu a není tedy možné jej využít pro stejné účely jako Emotiv EPOC. Toto zařízení je určeno spíše pro trénování mozku pomocí jednoduchých her (Mindflex, Star-wars Force trainer). [26]


Obrázek 2.1: Zařízení NeuroSky MindWave (zdroj: [26])

### 2.1.3 Charakteristika zařízení EPOC

V této kapitole se budu podrobněji věnovat popisu neuroheadsetu Emotiv EPOC, který byl využit v rámci této práce.

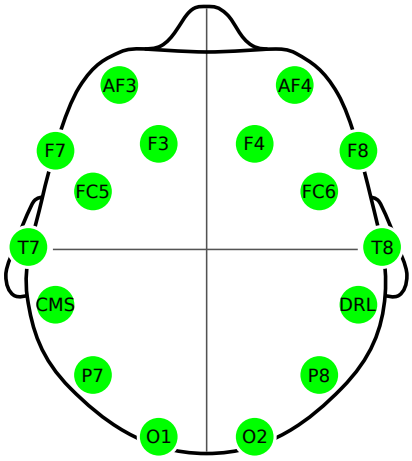
Emotiv EPOC (obrázek 2.2) je bezdrátová náhlavní souprava, která umožňuje zachytit neurosignály lidského mozku ve vysokém rozlišení a následně je zpracovat. Zařízení využívá celkem 14 senzorů + 2 referenční. Referenční senzory slouží pro odfiltrování externích artefaktů. Zařízení během zpracování signálu odečte ruch zachycený referenčním senzorem. [12] Rozmístění senzorů se řídí systémem 10/20 (viz obrázek 2.3).

Jedná se o levnou alternativu k profesionálním lékařským zařízením v ceně až několika tisíc dolarů. Emotiv EPOC přitom lze pořídit za cenu 399$, respektive 499$ za EPOC+. [25]

Primárně je zařízení určeno pro ovládání počítačových her, ale díky dodávanému API je možné využití i v jiných oblastech. Existují projekty, které využívají toto zařízení pro ovládání pohybu robotů, zařízení je také využíváno při různých výzkumech, dokonce se objevují pokusy o použití ve virtuální realitě.

Data jsou ze zařízení přenášena v zašifrovaném stavu do počítače pomocí bezdrátového spojení. Pro dešifrování a další zpracování dat slouží aplikace

Obrázek 2.2: Emotiv EPOC neuroheadset (zdroj: [25])




Obrázek 2.3: Rozložení elektrod zařízení Emotiv EPOC (zdroj: [25])

nazvaná EmoEngine. Tato aplikace je rozdělena do několika samostatných „modulů“, které zpracovávají různé oblasti lidských myšlenek a poskytují je ve formě předzpracovaných událostí. Mezi předzpracované události (signály) řadíme výrazy obličeje, emoce a kognitivní události (viz dále). [27] Následuje popis modulů aplikace EmoEngine.

**Expressiv Suite** – tento modul slouží k detekci základních výrazů obličeje. Dokáže rozpoznat 12 různých výrazů tváře: pohled doleva, pohled doprava, mrknutí, zavření levého oka, zavření pravého oka, zdvih obočí, zamračení, stisknutí zubů, úsměv, smích a úšklebek.

**Affectiv Suite** – zpracovává EEG a vrací míru (mezi 0 a 1) momentální hodnoty jedné z 5 základních emocí: angažovanost/znuděnost, frustrace, meditace, okamžité vzrušení a dlouhodobé vzrušení. Jedná se o emoce jejichž rozpoznání nemůže být zlepšeno tréninkem. K jejich rozpoznání

se používají univerzální charakteristiky lidských emocí vypozorované z EEG signálů.

**Cognitiv Suite** – slouží pro nadefinování a následnou detekci specifických myšlenek. Na rozdíl od Affectiv suite není možná okamžitá detekce bez nutnosti předchozího tréninku. Pro zprovoznění detekce vlastních myšlenek je potřeba nejprve zaznamenat referenční vzorek, který je pak použit pro rozpoznání konkrétní myšlenky. Během záznamu referenčního vzorku je nutné, aby byl uživatel v relativně uvolněném rozpoložení a nemyslel na nic jiného než na konkrétní myšlenku, které se trénink týká.

Kromě dat týkajících se mozkové aktivity poskytuje EmoEngine také informace o stavu baterie, kvalitě připojení jednotlivých elektrod nebo informace o hodnotách z vestavěného gyroskopu. V případě Research verze je možné získat i čistá EEG data v podobě hodnot naměřených jednotlivými elektrodami. [18, 24]

Emotiv EPOC obsahuje kromě senzorů pro záznam EEG signálu také gyroskop pro snímání pohybu hlavy[1] a poskytuje i EMG data. Jelikož Emotiv EPOC neobsahuje EMG senzory, jsou EMG data získávána díky tomu, že EEG senzory jsou dostatečně blízko obličejovým svalům a že EEG a EMG pracují na stejném principu. EMG data jsou v podstatě biologický artefakt, který ovlivňuje EEG signál. Oddělené biologické artefakty reprezentují samotná EMG data. [12]

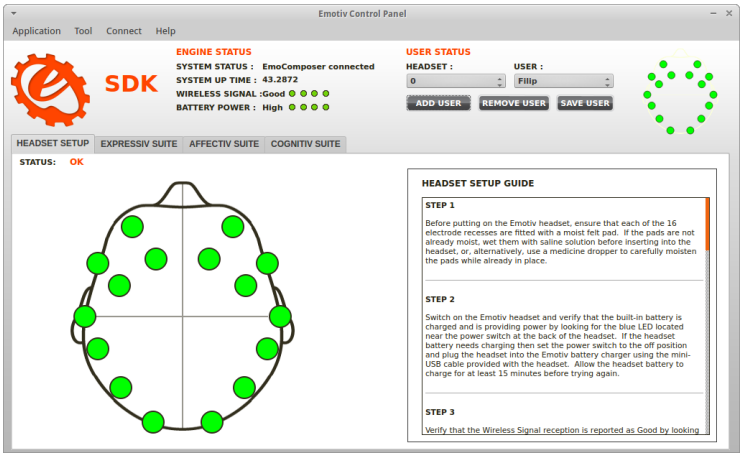
### 2.1.4 Softwarové vybavení

Spolu se zařízením dodává společnost Emotiv i základní softwarové vybavení pro práci s headsetem.

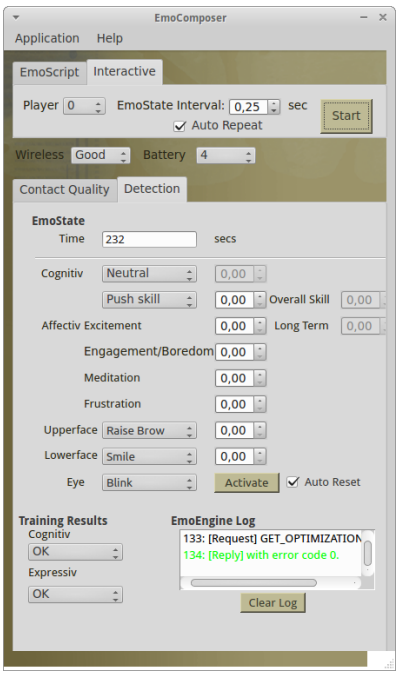
Základ tvoří aplikace Emotiv Control Panel (obrázek 2.4). Tato aplikace slouží k obsluze spojení, poskytuje informace o kvalitě spojení a o stavu baterie a přijímá a zpracovává mozkové signály. Aplikace umožňuje vizualizaci výrazů obličeje, zobrazení dešifrovaných emocionálních stavů a také je zde možnost trénovat své myšlenkové procesy a pomocí těchto myšlenek poté ovládat počítač.

Další aplikací je EmoComposer (obrázek 2.5). Jedná se o simulátor, který umožňuje pomocí grafického rozhraní nadefinovat hodnoty struktury EmoState a takto definovaný stav odeslat aplikaci, jako by se jednalo o data pocházející z fyzického zařízení. V případě, že bychom chtěli stejné nastavení používat opakovaně, je zde možnost využít tzv. Emo skript. Jedná se o XML dokument, kde jsou nadefinované hodnoty vlastností a čas, kdy se má tato hodnota využít pro vygenerování odpovídající události. Tuto aplikaci jsem hojně využíval pro testování.

[1] V zařízení není akcelerometr, takže není možné rozpoznat pohyby ve směrech nahoru/dolů, doleva/doprava apod., ale gyroskop rozpozná pouze otáčení hlavy




Obrázek 2.4: Emotiv Control Panel




Obrázek 2.5: Aplikace EmoComposer

K dispozici je také aplikace EmoKey, která umožňuje spojení s jinými aplikacemi jednoduchým převodem detekovaných událostí na kombinaci kláves. Uživatel si jednoduše nadefinuje požadovanou akci (klávesovou zkratku, kliknutí myši) a přiřadí jí určitý prvek, například úsměv. Pokud aplikace zaznamená, že se uživatel usmál, vykoná odpovídající akci.

Poslední aplikací je TestBench. Tato aplikace je ale dostupná pouze ve

variantě Research. Jedná se totiž o aplikaci, která slouží k vizualizaci čistého EEG signálu v podobě grafu, ve kterém jsou zobrazeny zaznamenané hodnoty z jednotlivých elektrod. Aplikace také umožňuje záznam čistého EEG signálu do formátu EDF[1].

Další aplikace je možné získat v oficiálním eshopu společnosti Emotiv na adrese https://emotiv.com/store/app. K dispozici jsou volně dostupné i placené aplikace.

### 2.1.5 Emotiv SDK

Společnost Emotiv nabízí jednoduché API, pomocí kterého je možné vytvářet aplikace kompatibilní se zařízeními Emotiv.

Emotiv SDK je poskytováno v podobě rozhraní pro jazyk C. SDK je rozděleno do 3 souborů - `edk.h`, `EmoStateDLL.h` a `edkErrorCode.h`. Hlavní část (tzv. EmoEngine) se nachází v souboru `edk.h`. EmoEngine je logická vrstva, která zajišťuje komunikaci mezi aplikací a připojeným zařízením. [29] Jsou zde definovány funkce pro komunikaci a získání dat. Příklad funkcí ze souboru `edk.h` je na ukázce 2.1. Přehled všech funkcí je k dispozici v dokumentaci na přiloženém CD ve složce `docs/EmotivSDK`.

Zdrojový kód 2.1: Příklad funkcí pro načtení dat z připojeného zařízení

```
1  /**
2   * Připojí zařízení s daným ID
3   * vrací chybové kódy definované v souboru edkErrorCode.h
4   **/
5  int EE_EngineConnect(String strDevID);
6
7  /**
8   * Funkce pro načtení další události z připojeného zařízení
9   * parametr hEvent je pointer na oblast paměti, kam je událost
        uložena
10  * vrací chybové kódy definované v souboru edkErrorCode.h
11  **/
12 int EE_EngineGetNextEvent(Pointer hEvent);
13
14 /**
15  * Funkce pro načtení struktury Emo state
16  * parametr hEvent je pointer na událost Emo Event
17  * parametr hEmoState  je pointer na oblast paměti, kam je
        uložena struktura Emo state
18  * vrací chybové kódy definované v souboru edkErrorCode.h
19  **/
20 int EE_EmoEngineEventGetEmoState(Pointer hEvent, Pointer
        hEmoState);
```

V souboru `edkErrorCode.h` jsou definovány chybové kódy. Přehled chybových kódů naleznete v příloze D.2. Tyto chybové kódy jsou návratovými

[1] EDF / EDF+ (European Data Format) je standardizovaný formát sloužící pro výměnu a ukládání biologických signálů. [28]

hodnotami některých funkcí a slouží pro určení, zda funkce proběhla v pořádku, a nebo byla zaregistrována nějaká chyba. Druh chyby je určen pomocí chybového kódu. Soubor `EmoStateDLL.h` je hlavičkový soubor, kde jsou definovány konstanty a rozhraní pro přístup ke struktuře EmoState.

EmoState je datová struktura, kterou generuje EmoEngine a která reprezentuje emoční stav uživatele v daném časovém okamžiku. EmoEngine komunikuje se zařízením, získává předzpracovaná EEG data a data z vestavěného gyroskopu, tato data dále zpracuje a výsledek přeloží do struktury EmoState. [29] Koncová aplikace nepřistupuje přímo ke struktuře EmoState, ale pro přístup k datům využívá funkce definované v souboru `EmoStateDLL.h`. Příklad funkcí definovaných v souboru `EmoStateDLL.h` je na ukázce 2.2. Přehled všech funkcí je k dispozici v dokumentaci na přiloženém CD ve složce `docs/EmotivSDK`.

Zdrojový kód 2.2: Příklad funkcí pro získání dat ze struktury EmoState

```
1  /**
2   * Funkce pro získání hodnoty emoce nadšení
3   * parametr state je pointer na strukturu EmoState
4   * funkce vrací hodnoty od 0.0 do 1.0
5   **/
6  float ES_AffectivGetExcitementShortTermScore(Pointer state);
7
8  /**
9   * Funkce pro získání expresivní akce
10  * parametr state je pointer na strukturu EmoState
11  * funkce vrací celočíselnou hodnotu, která odpovídá položce ve
        výčtovém typu EE_ExpressivAlgo_t
12  **/
13 int ES_ExpressivGetUpperFaceAction(Pointer state);
```

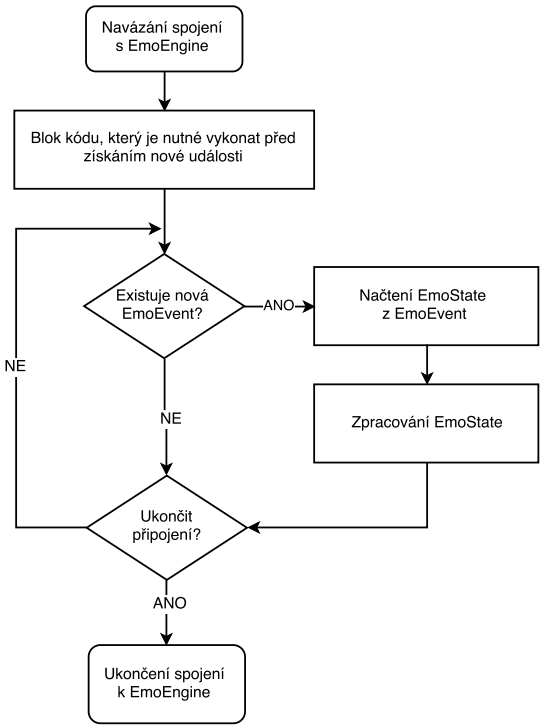
Komunikace mezi EmoEnginem a aplikací je zajištěna pomocí tzv. EmoEvents (přehled možných událostí je k dispozici v příloze D.1), které vytváří EmoEngine a aplikace je získává pomocí volání metody `EE_EngineGetNextEvent`. Z této struktury se poté pomocí volání metody `EE_EmoEngineEventGetEmoState` získá struktura EmoState, která již obsahuje samotná data.

Celá komunikace je naznačena na obrázku 2.6.

## 2.2 Model požadavků

### 2.2.1 Funkční požadavky

1. Komunikace se zařízením Emotiv EPOC nebo se simulátorem EmoComposer – vytvořená knihovna musí umožňovat komunikaci jak s fyzickým zařízením Emotiv EPOC, tak se simulátorem EmoComposer, který je součástí softwarového vybavení dodávaného s přístrojem.




Obrázek 2.6: Princip komunikace s EmoEngine (zdroj: [30])

2. Logování čistého EEG signálu – vytvořená sada aplikací musí umožňovat logování (záznam) čistého EEG signálu získaného ze zařízení Emotiv EPOC. Tento záznam bude ukládán do souboru.

3. Logování předzpracovaných událostí (výrazy obličeje, kognitivní události a emoce) pro následné využití v simulátoru EmoComposer - aplikace musí poskytovat funkce pro záznam předzpracovaných událostí, které Emotiv EPOC nabízí. Tento záznam bude ukládán do tzv. Emo skriptu. (viz kapitola 2.1.4)

4. Ovládání externí aplikace (prezentace, video přehrávač) pomocí událostí generovaných zařízením Emotiv EPOC, respektive EmoComposerem - na základě událostí získaných ze zařízení by mělo být možné základní ovládání nějaké externí aplikace. Ovládáním máme na mysli spuštění/zastavení prezentace (resp. videa), přepínání slidů apod.

5. Napojení na existující aplikaci InBeat (http://inbeat.eu) a umožnění tak jejího ovládání pomocí přístroje Emotiv EPOC.

6. Záznam emocí pro pozdější analýzu zájmu uživatele o prezentované informace

### 2.2.2 Nefunkční požadavky

1. Podpora operačních systémů Windows, Linux a Mac OS – vzhledem k tomu, že zařízení Emotiv EPOC je možné používat na počítačích s operačními systémy Windows, Linux i Mac OS, měly by i vytvořené aplikace pracovat bez potíží na těchto operačních systémech.

## 2.3 Návrh aplikací

Splnění výše popsaných požadavků nebude mít na starosti pouze jedna aplikace, ale bude vytvořena sada několika menších aplikací. Kromě těchto aplikací bude dále vytvořena sada funkcí (knihovna), která bude pracovat s připojeným zařízením a bude poskytovat další funkce, rozhraní a struktury potřebné pro fungování samotných aplikací. Tato knihovna by také měla usnadnit vývoj budoucích aplikací pro zařízení Emotiv EPOC.

### 2.3.1 Emotiv EPOC Library

Tato část se věnuje návrhu knihovny nazvané jako Emotiv EPOC Library. Jedná se v podstatě o sadu tříd, rozhraní a jiných struktur, jejichž účelem je vytvoření spojení se zařízením Emotiv EPOC, sběr dat, které zařízení produkuje, základní zpracování těchto dat a předání cílové aplikaci k dalšímu zpracování.

#### 2.3.1.1 Struktura knihovny

Zdrojové soubory jsou rozděleny do několika balíčků podle toho, které oblasti se dané soubory týkají. V knihovně se nachází celkem 7 kořenových balíčků (některé z nich mají další podbalíčky). Struktura balíčků, včetně stručného popisu je naznačena na následujících řádcích.

1. connection – jak již název napovídá, obsahuje tento balíček vše potřebné pro navázání spojení, ať už se jedná o fyzické zařízení nebo o simulátor

2. events – balíček obsahující abstraktní třídy a rozhraní pro vytvoření vlastní typů událostí a jejich zpracování

3. exceptions – balíček vlastních výjimek, které jsou použity v případě neočekávaných událostí (zejména v případě chyby během navazování spojení)

4. io – balíček obsahující třídy pro obsluhu vstupu, respektive výstupu

a) loggers – sada tříd zajišťující zalogování událostí přijatých z připojeného zařízení

5. sdk – API od společnosti Emotiv. Jsou zde obsaženy definice funkcí pro vlastní komunikaci se zařízením Emotiv EPOC

6. structures – různé pomocné struktury, které slouží například pro získání jména události z číselného kódu. Dále jsou zde třídy, které umožňují jednodušší přístup ke strukturám, které poskytuje EmoEngine.

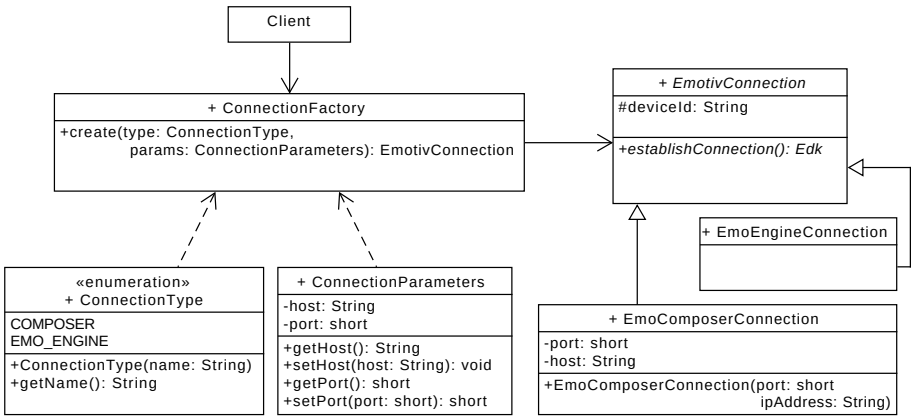
7. test – balíček obsahující unit testy pro ověření správné funkčnosti knihovny. Více o těchto testech naleznete v kapitole 4.

#### 2.3.1.2 Navázání spojení

Z požadavků vyplývá, že bude potřeba zajistit připojení k zařízení Emotiv EPOC, ale také k simulátoru EmoComposer. Pro tyto účely budou vytvořeny dvě třídy - jedna pro připojení k Emotiv EPOC a druhá pro připojení k EmoComposeru. Pro získání konkrétní instance bude vytvořena jednoduchá třída `ConnetionFactory`, která v závislosti na předaném typu vrátí instanci odpovídající třídy. Přípustné typy spojení jsou definovány ve výčtovém typu `ConnectionType` v balíčku `structures`.

Ve chvíli, kdy bude chtít klient navázat spojení, požádá `ConnectionFactory`, která klientovi na základě hodnoty proměnné `type` vrátí instanci odpovídající třídy. Tuto třídu poté klient využije pro samotné připojení.

Diagram zachycující způsob vytváření spojení je na obrázku 2.7.



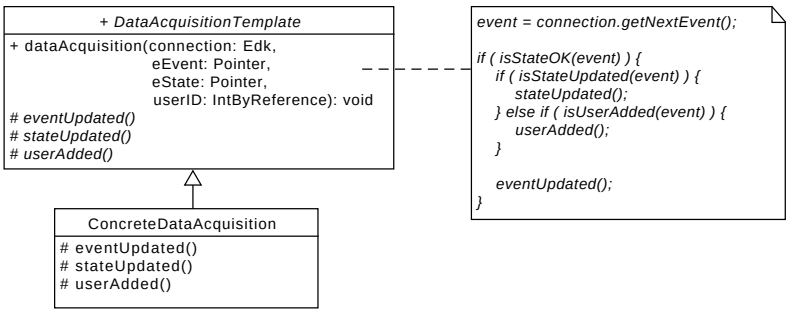

Obrázek 2.7: Diagram tříd zajišťující vytvořeení požadovaného typu spojení

#### 2.3.1.3 Získávání dat z připojeného zařízení

Dalším úkolem, který bude mít knihovna na starosti, je získávání dat z připojeného zařízení. Během samotné implementace jsem zjistil, že se kostra kódu pro načítání dat dokola opakuje, proto jsem se rozhodl vytvořit šablonu, která samotné načítání dat v koncových aplikacích co nejvíce zjednoduší.

K tomuto účelu byl použit návrhový vzor Template method, který slouží pro řešení podobných situací. Návrhový vzor Template method umožňuje v rodičovské třídě definovat kostru algoritmu a potomky této třídy nechává upřesnit některé dílčí kroky algoritmu. V rodičovské (abstraktní) třídě je definovaná tzv. template metoda, která definuje kostru algoritmu a volá primitivní operace, které představují kroky algoritmu, jež je možné měnit. Template metodu nesmí být možné přetížit! Konkrétní třída pak implementuje abstraktní operace definované v rodičovské třídě. [31]

Způsob realizace je naznačen na obrázku 2.8. Na obrázku je také vidět struktura kódu template metody.




Obrázek 2.8: Diagram template metody pro získání dat ze zařízení

#### 2.3.1.4 Záznam přijatých dat

Knihovna musí dále poskytovat funkce, které umožní záznam dat. Primárně se bude jednat o záznam čistého EEG signálu (tento záznam bude uložen v CSV souboru) a o záznam předzpracovaných událostí, které budou ukládány do Emo skriptu pro pozdější použití v simulátoru.

Každý typ záznamu bude mít na starost samostatná třída - dále budu tuto třídu označovat jako logger. Budou tedy vytvořeny dva loggery - `EEGLogger` pro záznam EEG signálů a `EmoLogger` pro předzpracované události. Rodičem je abstraktní třída `Logger`, která definuje společné vlastnosti a funkce.

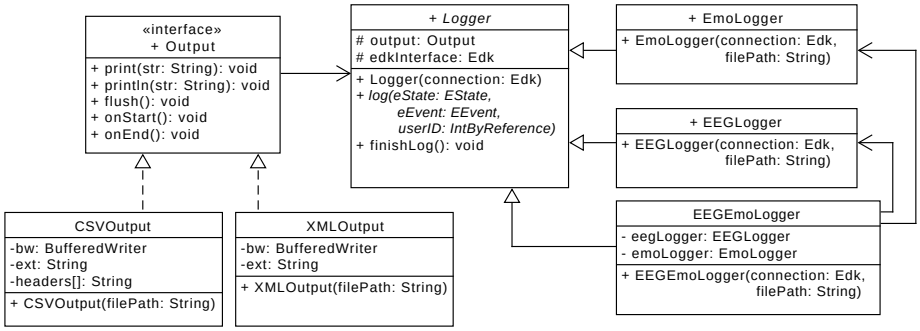
Po konzultaci s vedoucím práce byl přidán další logger (`EEGEmoLogger`), který umožní současný záznam EEG signálu a předzpracovaných událostí. Tento logger v sobě kombinuje `EEGLogger` a `EmoLogger`.

V případě, že by se v budoucnosti naskytla situace, kdy bude potřeba vytvořit jiný typ loggeru, který bude záznam ukládat například do EDF formátu, je možné vytvořit nový logger jednoduchým rozšířením abstraktní třídy `Logger`.

Výše zmíněné loggery mají na starost zpracování struktur (zejména struktury EmoState) získaných pomocí Emotiv API z připojeného zařízení. Samotný zápis ale mají na starosti třídy `CSVOutput` v případě EEGLoggeru a třída `XMLOutput` v případě EmoLoggeru. Instance těchto tříd jsou uloženy v instanční proměnné daného loggeru. `CSVOutput` a `XMLOutput` implementují rozhraní `Output`, které definuje dostupné operace.

Podobně jako v případě Loggeru je možné vytvořit nový typ výstupu implementování rozhraní `Output`.

Diagram tříd určených pro záznam je na obrázku 2.9.




Obrázek 2.9: Diagram tříd starající se o záznam dat

#### 2.3.1.5 Pomocné struktury

Pro lepší manipulaci s daty nebo pro usnadnění práce jsem navrhl sadu pomocných struktur a tříd.

**Channels** Někdy může být užitečné získat jméno elektrody (kanálu), ze které získáváme data. Totéž je užitečné i v případě, že chceme k jednotlivým kanálům zobrazit kvalitu spojení. Seznam kanálů je definován v Emotiv API ve výčtovém typu `EE_InputChannels_t`. Třída `Channels` obsahuje funkci, která hodnotu z tohoto výčtového typu převede na odpovídající název kanálu.

**Cognitiv** Tato třída má na starost převod číselného označení vyvolané kognitivní události na její textové označení. Toto označení je pak používáno například při záznamu do Emo skriptu.

**ConnectionParameters** Třída, která slouží jako kontejner pro předání parametrů potřebných pro navázání spojení do `ConnectionFactory` (viz obrázek 2.7)

**ConnectionType** Jedná se o výčtový typ, kde jsou nadefinovány typy připojení, které je možné vytvořit pomocí `ConnectionFactory` (viz obrázek 2.7)

**EEvent** Třída obalující datovou strukturu eEvent, kterou vytváří Emotiv API. Poskytuje funkce pro snadnější získání dat z této struktury.

**EState** Třída obalující datovou strukturu EmoState, kterou vytváří Emotiv API. Poskytuje funkce pro snadnější získání dat z této struktury a další pomocné funkce (porovnání dvou stavů apod.).

**Expressiv** Podobně jako třída Cognitiv má i tato třída na starost převod číselného označení vyvolané události na její textové označení. V tomto případě se ale jedná o expresivní události.

### 2.3.2 Emotiv EPOC Logger

Aplikace EPOC Logger je nástroj s grafickým rozhraním, který bude umožňovat jednoduché ovládání záznamu dat a bude poskytovat základní informace o stavu a kvalitě spojení. Tato aplikace bude mít na starost splnění funkčních požadavků číslo 2 a 3.

#### 2.3.2.1 Návrh GUI

Aplikace bude rozdělena do dvou panelů. Jeden panel (`StatusPanel`) bude čistě informativní a budou se v něm zobrazovat informace o současném připojení (kvalita spojení, stav baterie, čas připojení) a také se zde bude zobrazovat kvalita signálu jednotlivých senzorů.

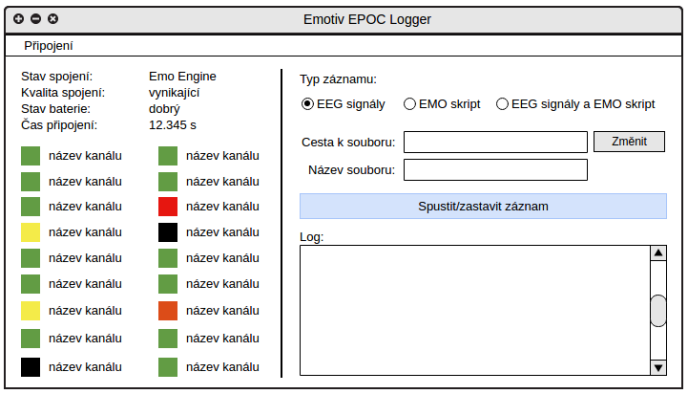
Druhý panel (`LoggerPanel`) bude sloužit pro nastavení samotného záznamu. Bude zde volba, která data chce uživatel zaznamenat a určí název a umístění souboru, kam se má záznam uložit. O průběhu záznamu bude uživatel informován pomocí jednoduchého výpisu.

K výběru typu připojení bude sloužit sekce „Připojení“ v hlavní nabídce aplikace. V této nabídce bude možnost zvolit připojení k simulátoru nebo k zařízení Emotiv EPOC. Dále zde bude možnost ukončit navázané spojení.

Pro lepší představu o výsledné podobě aplikace byl vytvořen wireframe, který je zobrazen na obrázku 2.10.

#### 2.3.2.2 Komunikace v rámci aplikace

Tím, že je aplikace rozdělena do dvou oddělených panelů, bylo potřeba zajistit komunikaci mezi nimi a ostatními částmi aplikace. K tomuto účelu se skvěle




Obrázek 2.10: Wireframe aplikace Emotiv EPOC Logger

hodí návrhový vzor Observer. Návrhový vzor Observer se používá v situacích, kdy na objektu závisí několik dalších objektů a tyto závislé objekty je třeba informovat o změnách stavu. [32]

Pro účely této komunikace bylo vytvořeno rozhraní `DeviceStateChangedListener`, které slouží pro informování o nově přijatých datech z připojeného zařízení. Dále bylo vytvořeno rozhraní `ConnectionListener`, které slouží pro informování o nově navázaném spojení.

Třídy `StatusPanel` a `LoggerPanel` implementují rozhraní `DeviceStateChangedListener`. Instance těchto tříd jsou zaregistrovány v instanci třídy `MainWindow`. `MainWindow` se poté postará o informování panelů ve chvíli, kdy dojde ke vzniku nové události. Podobně je tomu i v případě `ConnectionListeneru`. Diagram komunikace je k nahlédnutí v příloze E, obrázek E.1.

Podrobněji se tomuto tématu věnuje kapitola 3.2.

### 2.3.3 Formáty pro záznam dat

Zaznamenávaná data jsou ukládána do souborů. Pro uložení EEG signálů je použit formát CSV a záznam Emo skriptu využívá formát EML (EmoComposer Markup Language).

CSV je jednoduchý textový formát pro uložení tabulkových dat, kde jsou jednotlivé hodnoty (sloupce) odděleny čárkami, řádky tabulky jsou odděleny znakem pro odřádkování. Vzhledem k charakteru EEG dat, které jsou poskytovány jako pole hodnot zachycených jednotlivými senzory, je tento formát pro jejich uložení dostačující a není třeba používat nějaký složitější způsob uložení.

V případě Emo skriptu byl formát pevně daný. Aby bylo možné použití v simulátoru EmoComposer, pro který je Emo skript primárně určen, bylo třeba použít EML formát (ukázka 2.3). Jedná se o XML dokument, který ale umí EmoComposer interpretovat. [30]

Zdrojový kód 2.3: Příklad dokumentu ve formátu EML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE EML>
<EML version="1.0" language="en_US">
  <sequence>
    <time value="0s15t">
      <expressiv_eye event="blink" value="1" />
      <affectiv event="engagement_boredom" value="0.3" />
      <expressiv_upperface event="eyebrow_raised" value ="0.85"
          />
      <expressiv_lowerface event="clench" value ="0.85" />
      <affectiv event="engagement_boredom" value="0.2" />
      <signal_quality
          value="1,1,1,0,0.5,1,0.75,0,0,0.75,0.5,0.25,
          0.5,0,0.25,0,0.5,0.5" />
    </time>
  </sequence>
</EML>
```

Struktura dokumentu je jednoduchá. Veškeré události, které chceme generovat, musí být uvedeny v sekci `<sequence>`. Jednotlivé události jsou pak reprezentovány sekcí `<time>`. U této sekce lze pomocí parametru `value`[2] nadefinovat, kdy se má daná událost vygenerovat. Sekce `<time>` poté obsahuje definici signálů, které bude EmoComposer v rámci události generovat. Dostupné možnosti jsou v tabulce 2.1. [30]

Tabulka 2.1: Možnosti definování signálů v rámci formátu EML

| **Skupina událostí** | **Povolené hodnoty** |
|---|---|
| cognitiv | push, pull, lift, drop, left, right, rotate_left, rotate_right, rotate_clockwise, rotate_counter_clockwise, rotate_forwards, rotate_reverse, disappear |
| expressiv_eye | blink, wink_left, wink_right, look_left, look_right |
| expressiv_upperface | eyebrow_raised, furrow |
| expressiv_lowerface | smile, clench, laugh, smirk_left, smirk_right |
| affectiv | excitement_long_term, engagement_boredom, excitement_short_term |
| signal_quality | Hodnoty oddělené čárkami vyjadřující kvalitu spojení jednotlivých senzorů. Hodnoty jsou v rozsahu 0-1. |

[2]EmoComposer dělí každou sekundu na 32 rámců, hodnota parametru `value` musí být ve tvaru 1s16t. Tato hodnota znamená, že daná událost se vygeneruje po uplynutí 1 sekundy a 16 rámců od spuštění skriptu, tedy asi po 1,5 sekundě.

Sekce `<time>` je v podstatě vyjádření struktury EmoState v jazyce EML. Proto bude vytvořena funkce, která na základě hodnot obsažených ve struktuře EmoState, vytvoří odpovídající záznam, který bude poté uložen do Emo skriptu.

### 2.3.4 EPOC Events Generator

Tento nástroj bude určen pro zajištění komunikace s externími aplikacemi pomocí klávesových zkratek, umožní napojení na aplikaci InBeat a zajistí záznam emocí uživatele. Bude tedy zajišťovat splnění funkčních požadavku číslo 4, 5 a 6.

#### 2.3.4.1 Předzpracování přijatých stavů

Zařízení, respektive nástroj EmoEngine, generuje obrovské množství stavů (struktury EmoState). Nové stavy jsou v zařízení generovány cca 128krát za sekundu. Každý nově vygenerovaný stav je poté pomocí funkcí SDK přijat aplikací EPOC Events Generator. Každý přijatý stav by měl EPOC Events Generator zpracovat a na základě dat, která stav obsahuje, by měl vygenerovat odpovídající událost (stisknutí klávesy, odeslání zprávy aplikaci InBeat apod.). Aby došlo k mírnému redukování počtu stavů, na které musí aplikace reagovat, bude uložen pouze takový stav, který se v alespoň jedné sledované hodnotě liší od posledního uloženého stavu. Sledované hodnoty jsou například hodnoty emocí, výrazy obličeje nebo hodnoty kognitivních událostí.

#### 2.3.4.2 Producent - Konzument

Vzhledem k faktu, že nové stavy jsou v zařízení generovány cca 128krát za sekundu, bylo potřeba zajistit, aby tyto stavy byly i v pravidelných intervalech odebírány. Z tohoto důvodu jsem se rozhodl rozdělit logiku aplikace do dvou vláken. Jedno vlákno (tzv. producent) se stará o získání stavu, provede rozhodnutí, zda se jedná o změněný stav, uloží ho do bufferu a připojené generátory na základě obsahu bufferu vygenerují události, které jsou uloženy do sdílené fronty. Druhé vlákno (tzv. konzument) se pak stará o zpracování událostí ze sdílené fronty. Díky tomu bude v případě déle trvajícího zpracování události zajištěno pravidelné odebírání nových stavů ze zařízení.

#### 2.3.4.3 Generování událostí

Aplikace generuje z přijatých struktur EmoState vlastní události. Ke generování těchto událostí slouží třídy, které budu označovat jako generátory. Tyto generátory dostanou na vstupu seznam doposud zachycených stavů a podle pravidel, která má generátor nadefinovaná, vytvoří novou událost. Událost je tvořena třídou `AbstractEvent`, respektive jejími potomky. Jednou z těchto

událostí může být například událost označující pokles hodnoty dlouhodobého nadšení. K vygenerování této události slouží generátor `LongExcitementRaisedEventGenerator`, který událost vygeneruje pouze v případě, že poslední uložený stav má hodnotu emoce „dlouhodobé nadšení“ nižší než stav předposlední. Obdobně fungují i ostatní generátory, pouze se liší výstupní události a pravidla pro jejich generování.

Kromě připravených generátorů je možné vytvářet další generátory. Pro tyto účely je v knihovně Emotiv EPOC Library připraveno rozhraní `EventGeneratorInterface`. Toto rozhraní definuje metodu `generateEvent`, která má jako parametr seznam stavů zachycených ze zařízení. V této metodě je obsažena logika pro generování vlastních událostí.

Generování událostí má na starosti vlákno „Producent“. Producent si udržuje seznam generátorů (instancí implementující rozhraní `EventGeneratorInterface`). Každý nový generátor je třeba do tohoto seznamu přidat, jinak nedojde k jeho spuštění. Všechny generátory uložené v seznamu jsou postupně spouštěny a na základě svých definovaných pravidel a zachycených emo stavů generují nové události. Tyto události (instance třídy `AbstractEvent`), jsou poté vloženy do sdílené fronty, kde čekají na zpracování vláknem „Konzument“.

#### 2.3.4.4 Zpracování událostí

Aby nedošlo k zaplnění sdílené fronty událostí, je nutné události z fronty postupně vybírat a zpracovávat je. K tomu slouží vlákno „Konzument“. Toto vlákno si ze sdílené fronty (v případě, že není prázdná), vyzvedne instanci vygenerované události a tuto událost zpracuje.

Ke zpracování slouží dílčí konzumenti, kteří jsou k vláknu přiřazeni. Zpracováním události je myšleno vygenerování klávesové zkratky, odeslání zprávy aplikaci Inbeat nebo nějaká jiná akce. Způsob jakým bude daná událost zpracována je definován logikou daného konzumenta. Například konzument určený ke generování klávesových zkratek nalezne v definovaných zkratkách takovou, která odpovídá jménu a hodnotě zpracovávané události.

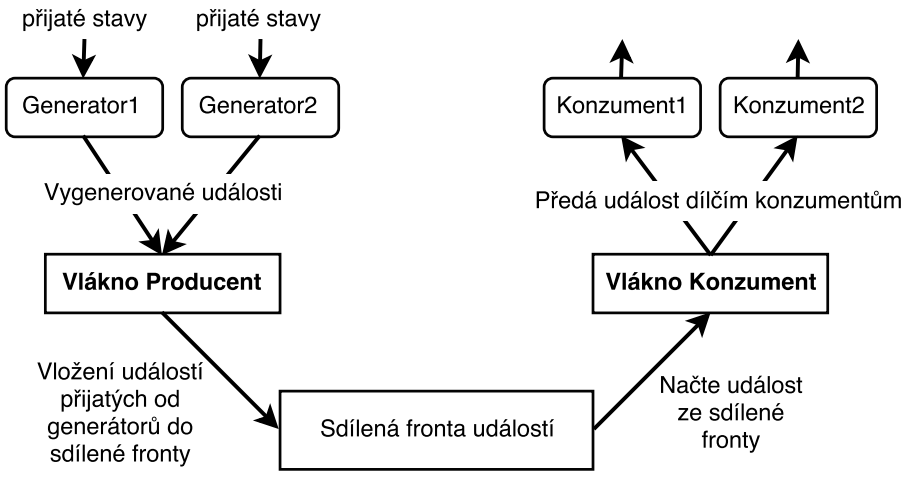
Dílčí konzumenti implementují rozhraní `ConsumerInterface`. Toto rozhraní definuje metodu `consumeEvent`, která jako parametr očekává instanci `AbstractEvent`.

Celý proces generování a zpracování událostí je naznačen na obrázku 2.11.

#### 2.3.4.5 Ovládání externích aplikací pomocí klávesových zkratek

Jedním z požadavků je možnost ovládat externí aplikace pomocí klávesových zkratek generovaných na základě událostí z neurohedsetu Emotiv EPOC.

Tento úkol bude mít na starosti konzument `ShortcutConsumer`, který bude na základě vygenerovaných událostí a předem nadefinovaných pravidel simulovat vykonání klávesových zkratek.




Obrázek 2.11: Generování a zpracování událostí nástrojem EPOC Events Generator

### Definice pravidel pro generování klávesových zkratek

Aby bylo umožněno co nejuniverzálnější řešení, byl vytvořen jednoduchý XML soubor, který bude sloužit pro definování pravidel pro klávesové zkratky.

Sktruktura dokumentu je definována pomocí XSD schématu[33]. Jedná se o způsob jak formálně popsat strukturu XML dokumentu, povolené hodnoty a dostupné elementy. Vytvořené XSD schéma je k nahlédnutí v příloze F nebo na přiloženém CD. Příklad XML dokumentu definující klávesové zkratky je na ukázce 2.4.

Zdrojový kód 2.4: XML souboru pro definování klávesových zkratek

```
<?xml version="1.0" ?>
<actions>
    <action>
        <shortcut>
            <value>CTRL+N</value>
        </shortcut>
        <name>blink</name>
        <value>0.0</value>
    </action>
    <action>
        <shortcut>
            <value>CTRL+SHIFT+A</value>
        </shortcut>
        <name>push</name>
        <value>0.3</value>
    </action>
</actions>
```

V případě, že se uživatel rozhodne, že chce ovládat jinou aplikaci, která využívá jiné klávesové zkratky, stačí pouze vytvořit nový XML soubor a tento soubor předat aplikaci EPOC Events Generator. Není tedy nutný žádný zásah do kódu aplikace.

#### 2.3.4.6 Napojení na aplikaci InBeat

Další úkolem bylo napojit EPOC Events Generator na webovou aplikaci InBeat (http://inbeat.eu). InBeat je experimentální akademická práce. Jedná se o službu pro doporučování obsahu. Aplikace obsahuje video přehrávač a snaží se, na základě snímání uživatele pomocí webové kamery, odhadnout zájem o sledované video a podle toho doporučit jiná videa, která by se mohla uživateli líbit. Doporučování pouze na základě snímání pomocí kamery není ideální, lepších výsledků by mohlo být dosaženo v případě využití signálů, které poskytuje Emotiv EPOC.

K aplikaci je možné se připojit pomocí Web Sockets. Díky technologie Web Sockets je možné vytvářet webové aplikace, kde klient navazuje obousměrné spojení se serverem. Po navázání spojení je možná výměna dat v reálném čase. Není tedy nutné, aby se klient v pravidelných intervalech dotazoval serveru, zda pro něj má nějaké nové informace. [34]

Komunikace s aplikací je zajištěna pomocí zasílání JSON zpráv. Aplikace definuje zprávu pro připojení k přehrávači a zprávu definující akci, která se má s přehrávačem vykonat. Formát těchto zpráv je uveden na ukázce 2.5.

Zdrojový kód 2.5: JSON zprávy pro komunikaci s aplikací InBeat

```
// zpráva pro připojení k přehrávači
{
  "interaction": {
    "type":"auth"
  },
  "attributes":{
    "action":"join",
    "value":"InBeatPlayer"
  }
}

// zpráva definující akci pro ovládání přehrávače
{
  "interaction":{
    "type":"event"
  },
  "attributes": {
    "action":"Play" // další povolené akce: pause, stop, seek+,
        seek-, volume+, volume-, previous, next, bookmark
  }
}
```

Jedním z dílčích konzumentů tedy bude konzument, který bude umožňovat komunikaci s aplikací InBeat pomocí protokolu WebSocket.

### 2.3.5 Volba programovacího jazyka

Před samotnou implementací bylo ještě nutné vybrat vhodný programovací jazyk. Výběr byl omezen na C++, Javu a C#, jelikož pro tyto jazyky je dostupné API pro komunikaci se zařízením Emotiv EPOC.

Volba nakonec padla na Javu. Tento jazyk jsem zvolil z důvodu snadné přenositelnosti „javovských“ aplikací mezi různými operačními systémy.

Kapitola 3

# Implementace

Tato kapitola popisuje implementaci knihovy Emotiv EPOC Library a aplikací Emotiv EPOC Logger a EPOC Events Generator.

## 3.1 Emotiv EPOC Library

### 3.1.1 Obsluha připojení

Jedním z hlavních úkolů knihovny je navázání připojení a jeho obsluha. Knihovna umožňuje volbu mezi dvěma typy připojení - připojení k EmoEngine nebo k simulátoru EmoComposer. O samotné vytvoření spojení se stará třída `EmoEngineConnection`, respektive `EmoComposerConnection`. Obě třídy jsou potomkem třídy `EmotivConnection`. V této třídě jsou definovány společné parametry pro oba typy spojení. Zároveň je zde definována abstraktní metoda `establishConnection`.

O vytvoření třídy, která odpovídá požadovanému typu spojení, se stará třída `ConnectionFactory`, konkrétně její metoda `create` Metodě je předán požadovaný typ spojení (položka výčtového typu `ConnectionType`) a instance třídy `ConnectionParameters`, která obsahuje parametry vyžadované pro navázání spojení. Obsah metody `create()` je vidět na ukázce 3.1.

Zdrojový kód 3.1: Metoda create třídy ConnectionFactory

```
1 public EmotivConnection create(ConnectionType type,
      ConnectionParameters params) throws UnknownConnectionType {
2   switch ( type ) {
3     case EMO_ENGINE:
4       return new EmoEngineConnection();
5     case COMPOSER:
6       return new EmoComposerConnection(params.getPort(),
            params.getHost());
7   }
8   throw new UnknownConnectionType();
9 }
```

Metoda na základě předaného typu rozhodne o tom, jaká instance má být vytvořena. Pokud je použit typ, pro který není známá implementace, metoda skončí vyjímkou.

Použití továrny `ConnectionFactory` je naznačeno na ukázce 3.2.

Zdrojový kód 3.2: Použití továrny ConnectionFactory

```
1  // Vytvoření továrny
2  ConnectionFactory factory = new ConnectionFactory();
3  ...
4  // Vytvoření parametrů pro připojení
5  ConnectionParameters params = new ConnectionParameters();
6  ...
7  // Vytvoření instance pro připojení k EmoComposer
8  EmotivConnection composerConnection =
       factory.create(ConnectionType.COMPOSER, params);
9
10 // Vytvoření instance pro připojení k EmoEngine
11 EmotivConnection engineConnection =
       factory.create(ConnectionType.EMO_ENGINE, params);
12
13 // Navázání spojení
14 Edk edk = engineConnection.establishConnection();
```

Z ukázky je vidět, že návratová hodnota metody `create` je uložena do proměnné typu `EmotivConnection`, na této proměnné je poté volána metoda `establishConnection`. Díky dodržení principu LSP (Liskov substitution principle), který říká, že všechny podtřídy musí být v souladu s chováním, které klienti očekávají od základních tříd a díky dědičnosti, máme zaručeno, že se vykoná správná implementace metody `establishConnection`.

### 3.1.2 Načítání dat ze zařízení

Nyní máme navázané spojení se zařízením a můžeme začít načítat data, která zařízení poskytuje. Jelikož kostra algoritmu pro načítání dat je pořád stejná a s největší pravděpodobností bude získávání dat implementováno ve většině aplikací, které budou knihovnu využívat, vytvořil jsem třídu `DataAcquisitionTemplate` a její metodu `dataAcquisition`. V těle metody `dataAcquisition` je definována kostra algoritmu pro získávání dat (ukázka 3.3). Tato kostra bude ve všech případech stejná, měnit se mohou jen její části. Tyto části jsou vymezeny abstraktními metodami `eventUpdated`, `stateUpdate` a `userAdded`. Potomci třídy `DataAcquisitionTemplate` implementují tyto metody a tím si přizpůsobí algoritmus pro své potřeby.

Zdrojový kód 3.3: Metoda definující kostru algoritmu pro získávání dat

```
1  public final void dataAcquisition(Edk connection, Pointer
       eEvent, Pointer eState, IntByReference userID) throws
       InternalDeviceErrorException {
2    // načtení struktury eEvent
3    int state = connection.EE_EngineGetNextEvent(eEvent);
4
5    // načtení proběhlo v pořádku
6    if (state == EdkErrorCode.EDK_OK.ToInt()) {
7      // získání typu události
8      int eventType = connection.EE_EmoEngineEventGetType(eEvent);
9
10     // načtení ID uživatele
11     connection.EE_EmoEngineEventGetUserId(eEvent, userID);
12
13     // na základě typu události se rozhodne, co se má stát
14     if (eventType == Edk.EE_Event_t.EE_EmoStateUpdated.ToInt()) {
15       connection.EE_EmoEngineEventGetEmoState(eEvent, eState);
16
17       // metoda, která může být změněna v potomkovi
18       stateUpdated();
19     } else if (eventType == Edk.EE_Event_t.EE_UserAdded.ToInt()
           && userID != null) {
20       connection.EE_DataAcquisitionEnable(userID.getValue(),
             true);
21
22       // metoda, která může být změněna v potomkovi
23       userAdded();
24     }
25
26     // metoda, která může být změněna v potomkovi
27     eventUpdated();
28   } else if (state != EdkErrorCode.EDK_NO_EVENT.ToInt()) {
29     throw new InternalDeviceErrorException();
30   }
31 }
```

Konkrétní použití třídy `DataAcquisitionTemplate` je ukázáno v kapitole 3.2.2.

### 3.1.3 Záznam přijatých dat

Knihovna poskytuje třídy pro záznam EEG signálů a předzpracovaných událostí(smích, mrknutí, kognitivní akce apod.). Vytvořil jsem tedy dvě třídy, jedná má na starost záznam EEG signálů a druhá záznam předzpracovaných událostí. Konkrétně se jedná o třídy `EEGLogger` a `EmoLogger`. Tyto třídy jsou potomkem abstraktní třídy `Logger`, která definuje abstraktní metodu `log`. Tato metoda je implementována v potomcích. Implementace metody `log` ve třídě `EEGLogger` je na ukázce 3.4. Na této ukázce je také vidět způsob načítání EEG dat.

Tělo metody `log` třídy `EmoLogger` je velice jednoduché. Spočívá pouze v tom získat z instance třídy `EState`, která je metodě `log` předána jako vstupní parametr, XML řetězec reprezentující daný stav v jazyce EML a předat ho k zápisu do souboru.

Zdrojový kód 3.4: Načítání EEG dat

```
IntByReference pEnableOut = new IntByReference(0);
IntByReference nSamplesTaken = new IntByReference(0);
Pointer hData = edkInterface.EE_DataCreate();

edkInterface.EE_DataUpdateHandle(0, hData);

// získá počet záznamů, které byly načteny
edkInterface.EE_DataGetNumberOfSample(hData, nSamplesTaken);

if (nSamplesTaken != null) {
  if (nSamplesTaken.getValue() != 0) {
    double[] data = new double[nSamplesTaken.getValue()];
    for (int sampleIdx = 0; sampleIdx <
        nSamplesTaken.getValue(); ++sampleIdx) {
      for (int i = 0; i < CSVOutput.headers.length; i++) {
        // načti data ze senzoru na pozici i
        edkInterface.EE_DataGet(hData, i, data,
            nSamplesTaken.getValue());

        // zapiš hodnotu na výstup
        output.print(data[sampleIdx] + ",");
      }
      output.println("");
    }
  }
}
```

O samotný zápis dat do souborů se starají potomci třídy `Output`. V případě `EEGLogger` se jedná o třídu `CSVOutput` a v případě `EmoLogger` je to třída `XMLOutput`. Tyto třídy kromě zápisu získaných dat zajišťují také zápis hlavičky a korektní ukončení souboru.

### 3.1.4 Pomocné struktury

Z vytvořených struktur stojí za zmínku především třídy `Cognitiv`, `Expressiv` a `EState`.

#### 3.1.4.1 Třídy Cognitiv a Expressiv

Funkce v Emotiv API určené pro získání informací o tom, která akce byla právě vykonána, vrací celočíselný identifikátor, který odpovídá číselné hodnotě položky ve výčtovém typu `EE_CognitivAction_t`, respektive `EE_ExpressivAlgo_t`. Tento číselný kód je většinou nicneříkající, lepší by proto bylo získat přímo název dané akce.

Pro získání názvu ale Emotiv API žádnou funkci neposkytuje a proto jsem vytvořil třídy `Cognitiv` a `Expressiv`. Tyto třídy mají za úkol převod získané číselné hodnoty na název akce. K tomu slouží metoda `getName`, která jako parametr očekává identifkátor akce a vrací řetězec obsahující název akce.

Chování obou tříd je totožné, liší se pouze v tom, kterou podmnožinu akcí (expresivní vs. kognitivní) mají na starosti.

#### 3.1.4.2 Třída EState

Třída `EState` obsahuje metody pro práci se strukturou EmoState, kterou generuje EmoEngine. Instance třídy `EState` reprezentuje jeden konkrétní stav načtený pomocí EmoEngine. Tento stav je předáván jako `Pointer`[3] v konstruktoru třídy `EState`. Aby nedošlo k náhodnému přepsání dat z jiného místa v aplikaci (což by díky pointeru bylo možné), je před samotným uložení vytvořena kopie dat (nová instance třídy `Pointer`) a teprve tato kopie je uložena do instanční proměnné třídy `EState` (ukázka 3.5).

Zdrojový kód 3.5: Konstruktor třídy EState

```
1 public EState(Pointer eState) {
2   ...
3   // Vytvoření nového pointeru na strukturu EmoState
4   this.eState = Edk.INSTANCE.EE_EmoStateCreate();
5
6   // Nakopírování dat do vytvořené struktury, první parametr je
        cíl, druhý parametr představuje zdroj dat
7   Edk.INSTANCE.ES_Copy(this.eState, eState);
8   ...
9 }
```

Hlavním důvodem pro vytvoření této třídy bylo usnadnění práce se strukturou EmoState. Rozhraní EmoState sice poskytuje funkce pro získání dat z této struktury, ale jejich názvy jsou většinou velmi dlouhé a při použití se kód stával místy dost nepřehledný.

Další důvod byl ten, že pro načtení některých informací nestačí zavolat pouze jedinou funkci, ale je potřeba inicializovat další pomocné struktury nebo volat několik funkcí v předepsaném pořadí. Tento případ je vidět na ukázce 3.6, kde je zobrazena funkce pro načtení kvality spojení jednotlivých senzorů.

Struktura dále obsahuje funkci pro porovnání dvou stavů a určení, zda jsou stavy totožné nebo funkci umožňující vytvoření záznamu pro Emo skript.

[3] Třída z knihovny JNA představující datový typ pointer. Knihovna JNA zapouzdřuje rozhraní JNI, které slouží pro propojení Javy a C/C++

Zdrojový kód 3.6: Načtení kvality spojení jednotlivých senzorů

```
1  public int[] getContactQuality() {
2    // emoStateInterface je implementace rozhraní EmoState
         umožnující přístup ke struktuře EmoState
3    int nquality =
         emoStateInterface.ES_GetNumContactQualityChannels(
         this.eState);
4    Pointer contactQualityP = new Memory(4 * nquality);
5    IntByReference contactQuality = new IntByReference();
6
7    contactQuality.setPointer(contactQualityP);
8
9    emoStateInterface.ES_GetContactQualityFromAllChannels(
         this.eState, contactQuality, nquality);
10
11   return contactQualityP.getIntArray(0, nquality);
12 }
```

## 3.2 Emotiv EPOC Logger

Tato kapitola se věnuje popisu implementace aplikace Emotiv EPOC Logger.

### 3.2.1 Grafické rozhraní

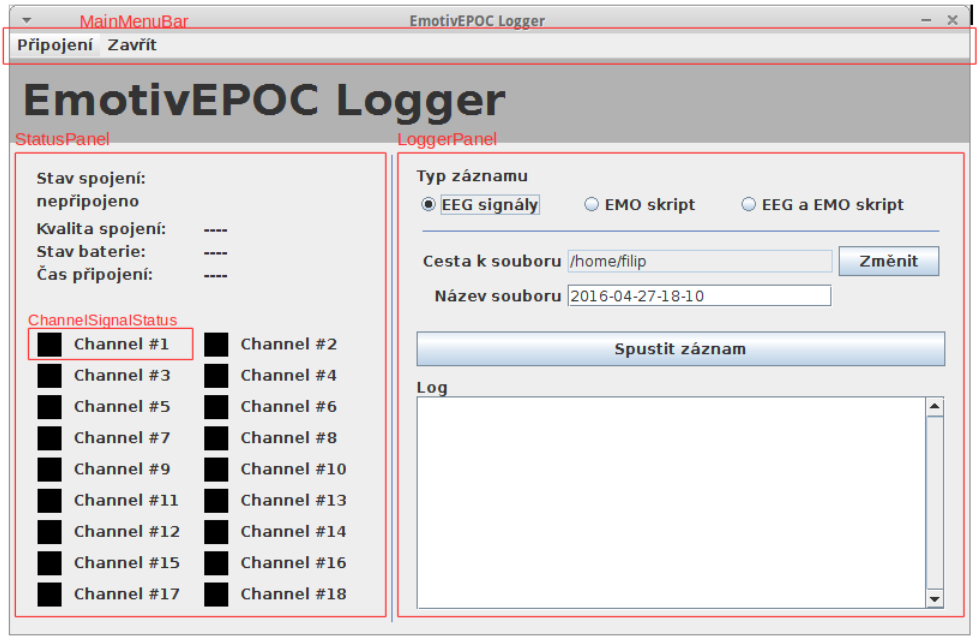
Aplikace je tvořena jedním hlavním oknem (třída `MainWindow`), které je rozděleno do dvou samostatných panelů. Tyto panely jsou reprezentovány třídami `StatusPanel` a `LoggerPanel`. Některé volby pro ovládání aplikace jsou obsaženy v hlavním menu, které je reprezentováno třídou `MainMenuBar`. Výsledná podoba grafického rozhraní je na obrázku 3.1.

Pro vytvoření základních grafický prvků byla použita knihovna Swing, která je standardní součástí Javy.

#### 3.2.1.1 Informace o spojení a stavu přístroje

Zobrazení informací, které se týkají stavu spojení nebo přístroje, má na starosti třída `StatusPanel`. Je zde informace o tom, zda je navázáno spojení. V případě, že je spojení navázáno, zobrazí se informace o tom, se kterou aplikací (EmoEngine nebo EmoComposer) je Emotiv EPOC Logger spojen. Dále je zde vidět kvalita spojení, stav baterie a čas, jak dlouho je spojení navázáno. V poslední řadě panel obsahuje grafické zobrazení stavu připojení jednotlivých senzorů.

Každý senzor je reprezentováno třídou `ChannelSignalStatus`. Tato třída se stará o překreslení stavu v případě, že dojde ke změně. Způsob aktualizace je zobrazen na ukázce 3.7.




Obrázek 3.1: Grafické rozhraní aplikace Emotiv EPOC Logger

Zdrojový kód 3.7: Aktualizace informací o kvalitě připojení senzorů

```
1  this.channelSignalStatusMap = new HashMap<>();
2
3  // Vložení čidel do mapy, uvádím pouze jedno čidlo, obdobně je
       to i pro zbývajících 15 čidel
4  channelSignalStatus1 = new ChannelSignalStatus();
5  channelSignalStatusMap.put(0, channelSignalStatus1);
6  ...
7
8  // aktualizace pak probíhá následovně
9  int [] f = eState.getContactQuality();
10
11 for(int i = 0; i < f.length; i++) {
12   if ( channelSignalStatusMap.containsKey(i) ) {
13     channelSignalStatusMap.get(i).updateStatus(f[i]);
14     channelSignalStatusMap.get(i).setName((new
           Channels()).getChannelName(i));
15   }
16 }
```

Každá instance třídy `ChannelSignalStatus` reprezentující senzor je uložena do mapy pod klíčem, který odpovídá indexu senzoru v poli, které se získá voláním metody `getContactQuality()` třídy `EState`. Pro každou hodnotu v tomto poli se na základě jejího indexu vybere z mapy odpovídající instance třídy `ChannelSignalStatus` a pomocí volání metody `updateStatus` se zajistí aktualizace stavu daného senzoru.

#### 3.2.1.2 Nastavení záznamu

Nastavení záznamu obstarává panel `LoggerPanel`. Je zde možné zvolit, jaký typ záznamu se má uskutečnit, uživatel může určit jméno souboru, do kterého se záznam zapíše a složku, kde bude tento soubor vytvořen. Také je zde tlačítko pro spuštění, resp. ukončení záznamu. O průběhu záznamu je uživatel informován pomocí textové oblasti, do které se propisují přijaté instance třídy `EState`.

O spuštění/zastavení záznamu se stará jediné tlačítko. Ve výchozím stavu, je tlačítko určeno pro spuštění záznamu. Ve chvíli, kde je záznam spuštěn, se funkce tlačítka změní a slouží pro zastavení záznamu.

Aby nedošlo během záznamu k nechtěné změně nastavených parametrů (typ záznamu, umístění a název souboru), dojde pro spuštění záznamu k deaktivaci všech komponent (kromě tlačítka pro ovládání záznamu) v `LoggerPanelu`. Po zastavení záznamu jsou všechny komponenty opět aktivovány.

#### 3.2.1.3 Vytvoření spojení

Připojení k zařízení je umožněno z hlavní nabídky aplikace v sekci „Připojení“. Zde jsou akce pro vytvoření a ukončení připojení. Jedná se o jednoduché akce, které zajistí vytvoření požadovaného druhu spojení. K dispozici jsou akce „Připojit k EmoEngine“ nebo „Připojit k EmoComposer“.

Při výběru možnosti „Připojit k EmoEngine“ dojde ihned k připojení (není třeba nic nastavovat). Při výběru možnosti „Připojit k EmoComposer“ se ale zobrazí dialogové okno, které slouží pro upřesnění adresy a portu, na kterých EmoComposer naslouchá.

Po úspěšném navázání spojení je spuštěno nové vlákno, které se stará o načítání dat z připojené aplikace. Toto nové vlákno je vytvořeno z důvodu, aby během načítání dat nedošlo k zablokování grafického prostředí.

### 3.2.2 Načítání dat ze zařízení

Jak bylo zmíněno v předchozí části, je načítání dat odděleno do samostatného vlákna. Toto vlákno je reprezentováno třídou `MainRunnable`.

Jelikož se jedná o třídu, která bude obstarávat načítání dat, je tato třída potomkem třídy `DataAcquisitionTemplate`, ve které je definována kostra algoritmu pro načítání dat (viz kapitoly 2.3.1.3). Ve třídě `MainRunnable` jsou implementovány metody `eventUpdate`, `stateUpdate` a `userAdded` rodičovské třídy `DataAcquisitionTemplate`. Samotné načítání je pak obsluhováno kódem uvedeným na ukázce 3.8.

Zdrojový kód 3.8: Způsob načítání dat v aplikaci Emotiv EPOC Logger

```
1  public void run() {
2     while (this.running) {
3          eEvent = new EEvent(connection.EE_EmoEngineEventCreate());
4          eState = new EState(connection.EE_EmoStateCreate());
5          try {
6              this.dataAcquisition(this.connection,
                   eEvent.geteEvent(), eState.geteState(),
                   user.getUserId());
7          } catch (InternalDeviceErrorException e) {
8              e.printStackTrace();
9          }
10    }
11 }
12
13 @Override
14 protected void eventUpdated() { this.fireStateChangedEvent(); }
15
16 @Override
17 protected void stateUpdated() {}
18
19 @Override
20 protected void userAdded() { }
```

Metoda `eventUpdate` se volá po načtení dat ze zařízení a v tomto případě je jejím úkolem informování zaregistrovaných posluchačů o načtení nových dat, které je třeba zpracovat. Více o informování zaregistrovaných posluchačů naleznete v kapitole 3.2.3.

Metody `stateUpdate` a `userAdded` jsou prázdné, jelikož v této části algoritmu není třeba vykonávat žádnou akci.

### 3.2.3 Komunikace v rámci aplikace

Jelikož je aplikace rozdělena do několika tříd, které ale potřebují vzájemně komunikovat, bylo třeba vytvořit nějaké rozhraní, které tuto komunikaci umožní. Návrhu a popisu této komunikace jsem se věnoval již v kapitole 2.3.2.2.

Celá komunikace začíná ve chvíli, kdy uživatel vytvoří nové spojení kliknutím na jednu z možností v hlavním menu. O vytvoření nového spojení a tedy i o skutečnosti, že je možné přijímat data, je nutné informovat panely `StatusPanel` a `LoggerPanel`.

Akce pro vytvoření spojení se volají ze třídy `MainMenuBar`, která má odkaz na svého rodiče, kterým je instance třídy `MainWindow`. Třída `MainWindow` implementuje rozhraní `ConnectionListener` s metodami `connectionEstablished()` a `connetionClosed()`.

Po kliknutí na tlačítko v menu dojde k zavolání metody `connectionEstablished()`. V rámci této metody je pak vytvořeno nové vlákno (viz kapitola 3.2.2) pro sběr dat a toto vlákno si další komunikaci řídí samo.

Vytvořené vlákno má na starosti informování panelů `StatusPanel` a `LoggerPanel`. Oba panely implementují rozhraní `DeviceStateChangedListener` s metodou `stateChanged`. Třída `MainRunnable` obsahuje seznam `EventListenerList`, do kterého je možné uložit instance implementující rozhraní `EventListener` (potomkem tohoto rozhraní je i `DeviceStateChangedListener`). Pomocí metody `addDeviceStateChangedListener` (viz řádky 15 a 16 v ukázce G.1) jsou tedy k vláknu zaregistrovány instance tříd `StatusPanel` a `LoggerPanel`. Třída `MainRunnable` již zajistí, pomocí metody `fireStateChangedEvent`, informování panelů o nové události (viz řádky 32 až 39 v ukázce G.1).

Třídy `StatusPanel` a `LoggerPanel` implementují metodu `stateChanged`, která zajišťuje obsluhu nově vzniklých událostí, o kterých byly panely informovány. Volání metody `stateChanged` je zajištěno z metody `addDeviceStateChangedListener`. V případě StatusPanelu je to překreslení zobrazovaných informací a v případě LoggerPanelu pak zajištění zapsání dat do souboru. Zdrojový kód metody `stateChanged` je dostupný v příloze G, ukázka G.2.

## 3.3 EPOC Events Generator

EPOC Events Generator je jednoduchá aplikace pro ovládání externích aplikací pomocí zařízení Emotiv EPOC. Ke svému běhu nevyžaduje grafické prostředí, její konfigurování je zajištěno pomocí argumentů načítaných z příkazového řádku.

### 3.3.1 Načítání argumentů z příkazového řádku

Prvním úkolem, který bylo třeba v rámci této aplikace vyřešit, bylo načítání argumentů z příkazového řádku. K tomuto účelu jsem na internetu objevil knihovnu s název JCommander[35]. Jedná se o knihovnu s dobrou funkcionalitou a velice snadnou konfigurovatelností.

V první řadě je třeba vytvořit třídu, která bude sloužit pro uložení hodnot načtených z příkazového řádku. U instačních proměnných, jejiž hodnoty chceme načítat z příkazového řádku, pak stačí uvést anotaci `@Parameter` s příslušnými konfiguračními parametry.

Jelikož v aplikaci bude několik samostatných „modulů“, kteří budou vyžadovat zadání konfiguračních parametrů a v budoucnu mohou přibývat další, rozhodl jsem se, že každý „modul“ bude mít vlastní třídu, která bude sloužit pro uchování jejich parametrů. Obsah jedné z těchto tříd je na ukázce 3.9.

Zdrojový kód 3.9: Ukázka třídy pro načítání argumentů z příkazového řádku

```
1  public class ExtendedConnectionParameters extends
       ConnectionParameters implements ValidateableParameters {
2    @Parameter(names = "--connectionType", description = "Typ
       připojení", required = true, validateWith =
       ConnectionTypeValidator.class)
3    private String connectionType;
4
5    @Parameter(names = "--connectionComposerHost", description =
       "Adresa, na které běží EmoComposer")
6    private String connectionComposerHost;
7
8    @Parameter(names = "--connectionComposerPort", description =
       "Port, na kterém běží EmoComposer")
9    private String connectionComposerPort;
10
11   @Override
12   public void validate() {
13     if ( this.getConnectionType() == ConnectionType.COMPOSER ) {
14       if ( this.connectionComposerHost == null ||
             this.connectionComposerHost.isEmpty() ) {
15         throw new ParameterException("Hodnota parametru
             '--connectionComposerHost' musí být vyplněna.");
16       }
17       if ( this.connectionComposerPort == null ||
             this.connectionComposerPort.isEmpty() ) {
18         throw new ParameterException("Hodnota parametru
             '--connectionComposerPort' musí být vyplněna.");
19       }
20     }
21   }
22 }
```

Na ukázce 3.9 je vidět, že třída implementuje rozhraní `ValidateableParameters`. Toto rozhraní jsem vytvořil z toho důvodu, aby bylo možné validovat parametry nad rámec validací, které poskytuje JCommander. Například u parametrů zobrazených na ukázce je třeba zkontrolovat, zda jsou v případě volby připojení k EmoComposeru vyplněny i parametry `connectionComposerHost` a `connectionComposerPort`.

Zpracování argumentů zadaných na příkazové řádce pak zajišťuje kód zobrazený na ukázce 3.10.

Zdrojový kód 3.10: Ukázka zpracování argumentů z příkazového řádku

```
// Načtení parametrů z příkazového řádku
java.util.List<ValidateableParameters> validateableParameters =
    new ArrayList<>();
JCommander commander = new JCommander();

ExtendedConnectionParameters connectionParameters = new
    ExtendedConnectionParameters();
InBeatParameters inBeatParameters = new InBeatParameters();
ShortcutParameters shortcutParameters = new ShortcutParameters();

commander.addObject(connectionParameters);
commander.addObject(inBeatParameters);
commander.addObject(shortcutParameters);

validateableParameters.add(connectionParameters);
validateableParameters.add(inBeatParameters);
validateableParameters.add(shortcutParameters);

try {
  // parsuj zadané argumenty z příkazového řádku
  commander.parse(args);

  // validuj parametry, které jsou validovatelné
  for (ValidateableParameters par : validateableParameters) {
    par.validate();
  }
} catch (ParameterException e) {
  // pokud došlo k chybě (špatně zadaný parametr, nevyplněný
      povinný parametr apod.), tak zobraz nápovědu
  System.err.println("Během parsování parametrů došlo k chybě: "
      + e.getMessage());
  commander.usage();
  return;
}
```

### 3.3.2 Generování a zpracování událostí

Jak bylo zmíněno v návrhu, bude generování a zpracování událostí fungovat na principu Producent–Konzument. Vytvořil jsem tedy dvě vlákna a sdílenou frontu, pomocí které producentpředává události konzumentovi.

Generování události je zajištěno pomocí několika generátorů. Tyto generátory jsou implementací rozhraní `EventGeneratorInterface`. K vláknu reprezentující producenta jsou připojeny pomocí metody `addGenerator`. Kdykoliv je producentské vlákno požádáno o vygenerování nových událostí, dojde postupně ke spuštění všech přiřazených „malých“ generátorů.

Podobně je tomu i v případě konzumentů událostí. V tomto případě ale třídy představující jednotlivé konzumenty implementují rozhraní `Consumer-Interface`.

Způsob vytváření generátorů a konzumentů je na ukázce 3.11.

Zdrojový kód 3.11: Vytvoření generátorů a konzumentů událostí

```
1  // Vytvoření konzumentů
2  Consumer consumer = new Consumer(events);
3
4  if (shortcutParameters.isActive()) {
5      consumer.addConsumer(new
           ShortcutConsumer(shortcutParameters));
6  }
7  if (inBeatParameters.isActive()) {
8      consumer.addConsumer(new InBeatConsumer(inBeatParameters));
9  }
10 consumer.start();
11
12 // Vytvoření producentů
13 ConnectionFactory f = new ConnectionFactory();
14 Producer producer = new
       Producer(f.create(connectionParameters.getConnectionType(),
       connectionParameters), events);
15 producer.addGenerator(new ExcitementRaisedEventGenerator());
16 producer.addGenerator(new LongExcitementRaisedEventGenerator());
17 producer.addGenerator(new EyeEventGenerator());
18 producer.addGenerator(new CognitivEventGenerator());
19 Thread producerThread = new Thread(producer);
20 producerThread.start();
```

### 3.3.3 Generování klávesových zkratek

Jedním z úkolů, které aplikace EPOC Events Generator má, je generování klávesových zkratek. O tento úkol se stará třída `ShortcutConsumer`. V případě, že je tento konzument aktivní (jestli se konzument použije nebo ne, lze nastavit argumentem na příkazové řádce), generuje na základě pravidel v zadaném XML dokumentu (viz kapitola C.4.2) definované klávesové zkratky. Cesta ke XML dokumentu s pravidly se zadává argumentem z příkazového řádku.

Zvolený XML dokument je nutné převést do programové reprezentace. XML dokument je v aplikace EPOC Events Generator reprezentován třídou `KeyboardActionList`. Jednotlivé akce definované tagem `<action>` jsou reprezentovány třídou `KeyboardAction`.

O převod XML dokumentu na instance zmíněných tříd se stará třída `XMLActionsConvertor`, která s využitím knihovny X–Stream [36] převede formát XML na instanci třídy `KeyboardActionList`.

Jakmile `ShortcutConsumer` obdrží požadavek na zpracování nové události, zjistí, o jakou událost se jedná (načte její jméno, jménem se myslí například úsměv, mrknutí apod.) a získá její hodnotu (pokud nějakou má). Na základě jména a hodnoty se poté v seznamu akcí vyhledá odpovídající akce. Pokud se taková akce nalezne, dojde k vykonání přiřazené klávesové zkratky.

O „stisknutí“ kláves se stará třída `java.awt.Robot`, která umí stisk kláves simulovat.

Zjednodušená podoba třídy `ShortcutConsumer` je na ukázce 3.12.

Zdrojový kód 3.12: Třída generující klávesové zkratky

```
1  public class ShortcutConsumer implements ConsumerInterface {
2    public ShortcutConsumer(ShortcutParameters parameters) throws
         AWTException, FileNotFoundException {
3      this.convertor = new XMLActionsConvertor(); // converter
           zajišťující převod XML do vnitřní reprezentace
4      this.actionList =
           convertor.convert(parameters.getFilePath()); // převod
           XML na instanci třídy KeyboardActionList
5      this.keyboard = new Keyboard(); // třída simulující práci s
           klávesnicí
6    }
7
8    // metoda volaná vždy, když dojde k vygenerování nové
         události, kterou je třeba zpracovat
9    @Override
10   public void consumeEvent(AbstractEvent abstractEvent) {
11     // v seznamu akcí vyhledá akci podle zadaných parametrů, v
           případě nalezení ji vykoná
12     actionList.performAction(abstractEvent.getName(),
           abstractEvent.getValue(), this.keyboard);
13   }
14 }
15
16 public class KeyboardActionList {
17   public void performAction(String name, double value, Keyboard
         keyboard) {
18     for (KeyboardAction a : keyboardActions) {
19       if ( a.isEqualAction(name, value) ) {
20         // zajistí vykonání nalezené klávesové zkratky
21         keyboard.doShortcut(a.getShortcut());
22       }
23     }
24   }
25 }
```

### 3.3.4 Napojení na aplikaci InBeat

Aplikace je také schopna komunikovat s webovou aplikací InBeat. Tato komunikace je realizována zasíláním zpráv ve formátu JSON na adresu, na které běží websocketový server. Tento server zajistí předání poslaných zpráv aplikaci InBeat.

Komunikaci se serverem obstarává třída `InBeatConsumer`, což je v podstatě websocketový klient. Kód obstarávající komunikaci je na ukázce 3.13.

Zdrojový kód 3.13: Zasílání zpráv aplikaci InBeat

```
1  public class InBeatConsumer extends WebSocketClient implements
       ConsumerInterface {
2
3    private InBeatJson inBeatJson;
4
5    private boolean openedConnection = false;
6
7    public InBeatConsumer(InBeatParameters parameters) {
8      super(parameters.getServerURL()); // nastaví adresu
           websocketového serveru, ke kterému se budeme připojovat
9      this.connect(); // připojení k serveru
10
11     this.inBeatJson =  new InBeatJson();
12   }
13
14   @Override
15   public void consumeEvent(AbstractEvent abstractEvent) {
16     if ( !openedConnection ) { return; }
17
18     // Odeslání události
19     this.send(this.inBeatJson.eventJson(abstractEvent));
20   }
21
22   @Override
23   public void onOpen(ServerHandshake serverHandshake) {
24     System.out.println("connection open");
25     this.send(this.inBeatJson.joinRoomJson("InBeatPlayer"));
26     this.openedConnection = true;
27   }
28
29   @Override
30   public void onMessage(String s) { }
31
32   @Override
33   public void onClose(int i, String s, boolean b) {
34     System.out.println("connection closed");
35   }
36
37   @Override
38   public void onError(Exception e) {
39     System.out.println(e.getMessage());
40   }
41 }
```

Předávání informací aplikaci InBeat je zajištěno pomocí JSON zpráv. O generování těchto zpráv se stará třída `InBeatJson` a její metody. Kromě zpráv pro připojení a pro ovládání přehrávače (viz kapitola 2.3.4.6.) jsou aplikaci InBeat předávána data nasbíraná zařízením Emotiv EPOC. Těmito daty jsou především získané hodnoty emocí. Aplikace InBeat si poté sama rozhodne, co se získanými daty dále udělá. Formát těchto zpráv je na ukázce 3.14.

Zdrojový kód 3.14: Formát zpráv pro zasílání dat z Emotiv EPOC aplikaci InBeat

```
{
  "interaction":{
    "type":"epocEvent"
  },
  "attributes": {
    "name":"nazev_udalosti",
    "value":"hodnota_udalosti"
  }
}
```

### 3.3.5 Záznam emocí uživatele

Aplikace také umožňuje zaznamenávat emoce uživatele. Jedná se o velice jednoduchý proces, kdy jsou z přijatých událostí extrahovány data týkající se uživatelových emocí a tyto informace jsou zapsány do souboru včetně času (čas je počítán od spuštění nahrávání), kdy došlo k výskytu té konkrétní emoce. Takto získaná data je poté možné využít k analýze zájmu uživatele o sledovanou činnost.

Získaná data jsou ukládána do CSV souboru. Každá z 5 emocí je ukládána do vlastního souboru. Jeden řádek je tvořen časovým údajem v milisekundách a získanou hodnotou dané emoce. Tyto hodnoty jsou odděleny středníkem.

CSV formát jsem zvolil hlavně proto, že zápis do tohoto formátu je velice snadný (není třeba dodržovat nějaká složitá pravidla) a zároveň lze tento formát bez problémů otevřít v tabulkových editorech a pracovat s daty, jako by se jednalo o klasickou tabulku.

## 3.4 Shrnutí

V žádném případě nejsou v této kapitole popsány veškeré situace, které jsem během implementace musel řešit. Snažil jsem se zde popsat nejdůležitější a nejzajímavější části vytvořených aplikací. Kompletní zdrojový kód včetně doplňujících komentářů je k dispozici na přiloženém CD ve složce `src`, dokumentace je pak ve složce `docs`.

Kapitola 4

# Testování

V této kapitole je popsáno testování vytvořených aplikací. Jsou zde popsány způsoby testování a čtenář je seznámen s jejich výsledky.

## 4.1 Emotiv EPOC Library

O testování správné funkčnosti knihovny Emotiv EPOC Library se stará sada automatizovaných jednotkových testů. Pro tvorbu těchto testů byl použit framework JUnit verze 4, pro vytváření mock objektů pak framework Mockito. Soubory testů jsou dostupné ve zdrojových kódech projektu v balíčku `test`.

Pomocí vytvořených testů jsou otestovány následující oblasti funkčnosti knihovny:

- vytvoření správného typu připojení pomocí třídy `ConnectionFactory`
- vytvoření spojení s EmoEngine nebo s aplikací EmoComposer
- záznam dat pomocí tříd `EEGEmoLogger`, `EEGLogger` a `EmoLogger`
- výstup do souborů CSV a XML pomocí tříd `XMLOutput` a `CSVOutput`
- třída EState a její metody

Knihovna využívá funkce, které pracují se zařízením Emotiv EPOC a pro správnou funkčnost vyžadují jeho připojení. V případě spouštění jednotkových testů se ale nedá předpokládat, že budeme připojovat zařízení. Navíc bychom neměli zaručeno, že zařízení vrátí hodnoty, které jsou pro splnění testu očekávány.

Naštěstí ale existují tzv. mock objekty. V podstatě se jedná o nahrazení reálného objektu testovacím objektem (mock objektem), který neprovádí žádnou funkcionalitu nahrazovaného objektu, jen se jako tento objekt tváří. Místo původní logiky objektu je vloženo chování, které je v daném testu očekáváno.

Knihovna pracuje s instancemi tříd `Edk` a `EmoState`, které přímo komunikují s připojeným zařízením. Aby byla zajištěna očekávaná funkčnost, bylo třeba vytvořit k těmto třídám mock objekty. O inicializaci a nastavení požadované funkčnosti těchto objektů se starají třídy `EdkPrepare` a `EmoStatePrepare`. Ostatní třídy v balíčku `test` zajišťují otestování správné funkčnosti knihovnou poskytovaných funkcí.

### 4.1.1 Výsledek testů

Testy je možné spustit ve složce projektu (`src/EmotivEPOCLibrary`) pomocí příkazu `mvn surefire:test`.

Celkem bylo vytvořeno 7 testovacích tříd s celkem 22 testovacími metodami. Všechny tyto testovací metody skončili úspěšně.

## 4.2 Emotiv EPOC Logger

Aplikace Emotiv EPOC Logger je určena pro záznam dat získaných z připojeného zařízení Emotiv EPOC nebo ze simulátoru EmoComposer. Kromě záznamu aplikace také zobrazuje informace o navázaném spojení. Hlavní část testování tedy spočívala v ověření, že aplikace zaznamenává data, která přijme. V rámci testování byla také kontrolována správnost zobrazených informací o navázaném spojení.

### 4.2.1 Připojení a informace o spojení

V první části jsem testoval připojení k zařízení Emotiv EPOC a k EmoComposeru a kontrolovat správnost informací o navázaném spojení.

Nejprve jsem testoval komunikaci s EmoComposerem. Zaroveň s tím jsem i otestoval, že Emotiv EPOC Logger správně zobrazuje informace o spojení. Pomocí menu `Připojení > Připojit k EmoComposer` jsem vytvořil spojení k simulátoru, který jsem již před tím spustil a nastavil tak, aby všechny senzory byly zelené (nejlepší kvalita signálu). Poté jsem postupně měnil sílu signálu na jednotllivých senzorech a sledoval, zda dojde k aktualizaci správného senzoru v aplikaci Emotiv EPOC Logger. Vše probíhalo podle očekávání.

Podobně jsem postupoval i v případě připojení fyzického zařízení. V tomto případě mi jako zdroj pro kontrolu zobrazovaných informací posloužila aplikace Emotiv Control Panel. Jako v předchozím případě proběhlo vše podle očekávání.

### 4.2.2 Záznam dat

Dalším úkolem bylo otestovat správnost záznamu, který aplikace Emotiv EPOC Logger produkuje.

#### 4.2.2.1 Emo skript

Pro otestování korektního zaznamenání Emo skriptu jsem vytvořil referenční skript, který jsem poté spustil pomocí EmoComposeru a nechal zaznamenat pomocí aplikace Emotiv EPOC Logger. Výsledný soubor jsem poté porovnal s referenčním skriptem. Zaznamenaný skript sice není dokonale shodný se zdrojovým skriptem. Důležité ale je, že žádná z událostí nechybí a skript ani neobsahuje žádnou novou událost, která by nebyla ve zdrojovém souboru. Na základě tohoto testu můžeme prohlásit záznam Emo skriptu za korektní. Porovnání části těchto skriptů je zobrazeno na ukázce H.1. Kompletní zdrojový i zaznamenaný soubor je k nahlédnutí na přiloženém CD ve složce `test/EmotivEPOCLogger/emo_skript`. Jedná se o soubory s názvem `test_template.emo` a `test_template_zaznam.emo`.

Nyní bylo třeba otestovat záznam Emo skriptu z dat, která poskytuje fyzické zařízení. Připojil jsem tedy zařízení Emotiv EPOC a spustil záznam Emo skriptu. I zde probíhalo vše podle očekávání. Malá část tohoto skriptu je k dispozici na ukázce H.2. Kompletní zaznamenaný skript je uložen ve složce `test/EmotivEPOCLogger/emo_skript`. Jedná se o soubor s názvem `test_emo_skript.emo`.

#### 4.2.2.2 EEG signály

V případě EEG signálů není k dispozici žádný vzorový soubor, podle kterého bychom mohli zkontrolovat správnost zaznamenaných dat. Samotná data jsem tedy považoval za správná a pouze jsem kontroloval, že jsou načteny všechny hodnoty, které by načteny být měly. V každém řádku výsledného CSV souboru by mělo být celkem 22 hodnot (sloupců) oddělených čárkou. První řádek pak obsahuje hlavičku s označením sloupce. Po kontrole zaznamenaných dat si myslím, že záznam funguje jak má. Vytvořený soubor je uložen ve složce `test/EmotivEPOCLogger/eeg_signaly`.

#### 4.2.2.3 Současný záznam Emo skriptu a EEG signálů

Poslední možnost, kterou Emotiv EPOC Logger poskytuje, je záznam Emo skriptu a EEG signálů zároveň. Tento způsob byl v podstatě otestován předešlými dvěma testy, protože se jedná pouze o kombinaci předchozím dvou typů zaznamu. Vygenerované soubory `test_eeg_emo.csv` a `test_eeg_emo.emo` jsou opět na přiloženém CD ve složce `test/EmotivEPOCLogger/eeg_emo`.

### 4.2.3 Shrnutí

Na základě provedených testů můžeme prohlásit, že aplikace funguje spolehlivě. Je ale možné, že během dalšího používání aplikace dojde k odhalení některých skrytých chyb, které toto testování neodhalilo.

Průběh testování je zaznamenán ve dvou video souborech. První část týkající se testování zobrazených informací je v souboru `logger_test_info.mp4`, záznam dat je pak k dispozici ve videu `logger_test_log.mp4`. Videa jsou k dispozici na přiloženém CD ve složce `test/EmotivEPOCLogger`. Ukázky z těchto video záznamů jsou v příloze I, obrázky I.1, I.2, I.3 a I.4.

## 4.3 EPOC Events Generator

Nástroj EPOC Events Generator je složen ze 3 částí. Jedna část umožňuje generování klávesových zkratek, druhá pak slouží k napojení na aplikaci InBeat a poslední je určena pro záznam emocí uživatele.

### 4.3.1 Generování klávesových zkratek

První částí, kerou budeme testovat je generování klávesových zkratek. Tato část aplikace umožňuje na základě událostí získaných ze zařízení generovat klávesové zkratky, které jsou dané události přiřazeny.

Pro otestování správné funkčnosti jsem vytvořil testovací Emo skript pro spuštění v aplikaci EmoComposer. Tento skript je uložen v souboru `shortcut_test_script.emo` (část skriptu je na ukázce H.3) ve složce `test/EPOCEventsGenerator/shortcut`. Ve stejné složce je uložena i zkušební sada pravidel `shortcut_rules_test.xml`. Obsah tohoto souboru je uveden také na ukázce H.4. Jedná se o velice jednoduchou sadu, která obsahuje pouze akce pro vygenerování směrových šipek VLEVO a VPRAVO. Těmito klávesovými zkratkami bude ovládána jednoduchá javascriptová prezentace, která umožňuje přepíná slidů pomocí směrových šipek. Nástroj EPOC Events Generator zajistí spojení mezi aplikací EmoComposer simulující fyzické zařízení a webovou prezentací.

Průběh testování je zachycen na videu `shortcut_test.mp4`, které naleznete ve složce `test/EPOCEventsGenerator`. Ukázka z tohoto video záznamu je v příloze I, obrázek I.5.

### 4.3.2 Napojení na aplikaci InBeat

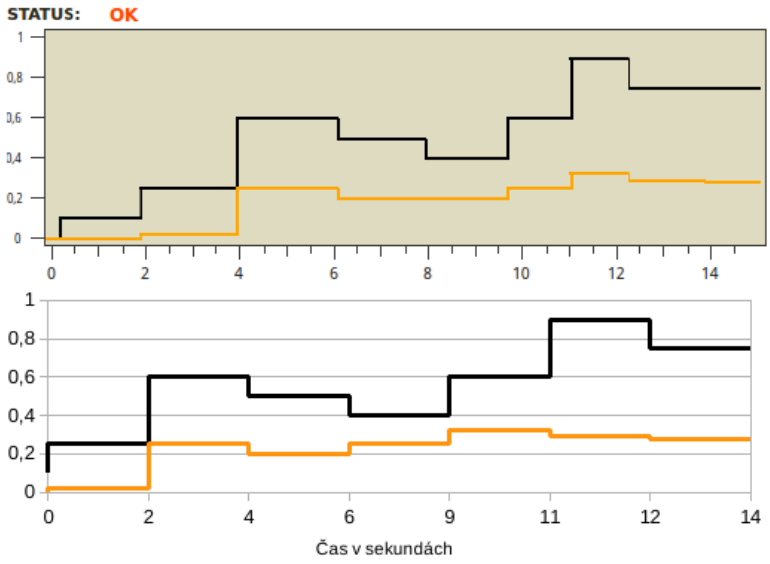
Pro otestování správného napojení na aplikaci InBeat jsem vytvořil Emo skript `inbeat_test_script.emo` (ukázka H.5) ve složce `test/EPOCEventsGenerator/inbeat`.
Tento skript obsahuje akce (blink, clench, wink_left, wink_right), které jsou určeny pro ovládání přehrávače v aplikaci InBeat. Na základě výskytu těchto akcí generuje nástroj EPOC Events Generator odpovídající JSON zprávy a odesílá je aplikaci InBeat. Kromě zpráv určených k ovládání přehrávače, jsou do aplikace InBeat zasílány i všechny přijaté události.

Průběh testu byl nahráván a výsledné video je k dispozici v souboru `inbeat_test.mp4` ve složce `test/EPOCEventsGenerator`. Na videu je vidět, že

aplikace InBeat správně reaguje na zaslané zprávy (spuštění videa, pozastavení videa, přepnutí na následující video). Ukázka z tohoto video záznamu je v příloze I, obrázek I.6.

### 4.3.3 Záznam emocí uživatele

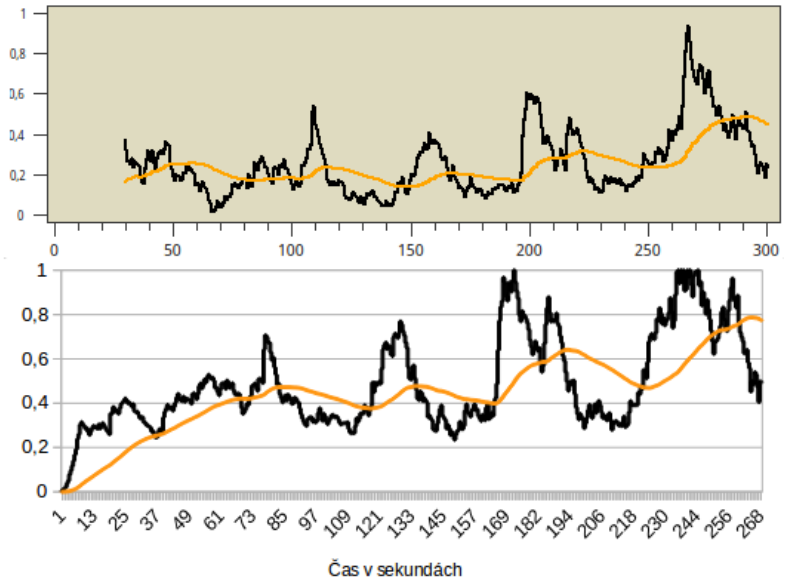
Poslední testovanou částí je část týkající se záznamu emocí uživatele. V této části bylo nutné otestovat, že nástroj skutečně zaznamenává správné hodnoty emocí. Pro tyto účely jsem vytvořil Emo skript (`emotion_test_script.emo`), který poskytne referenční data. Část skriptu je uvedena na ukázce H.6. Pomocí tohoto skriptu a aplikace EmoComposer byla nejprve vygenerovaná data odeslána do aplikace Emotiv Control Panel, díky které byl získán graf zobrazující vygenerované hodnoty emocí v závislosti na čase. Stejná data byla poté odeslána i do aplikace EPOC Events Generator, která provedla jejich záznam. Ze zaznamenaných hodnot byl vytvořen graf. Porovnáním obou grafů jsem dospěl k závěru, že EPOC Events Generator zaznamenává správné hodnoty emocí. Porovnání obou grafů je k nahlédnutí na obrázku 4.1 nebo v souboru `porovnani_emoci.pdf` ve složce `test/EPOCEventsGenerator/emoce`.




Obrázek 4.1: Porovnání zachycených emocí aplikací Emotiv Control Panel a nástrojem EPOC Events Generator - simulovaná data

Nyní bylo možné provést experiment s využitím fyzického zařízení. Připojil jsem tedy zařízení Emotiv EPOC a spustil aplikace EPOC Events Generator a Emotiv Control Panel. EPOC Events Generator zajistí záznam emocí a Emotiv Control Panel vytvoří referenční graf pro porovnání. Na serveru youtube.com jsem poté sledoval zhruba 5 minutové video. Po jeho skončení

jsem ze zaznamenaných dat vytvořil graf, který jsem porovnal s referenčním grafem získaným díky aplikaci Emotiv Control Panel. Porovnání grafů je na obrázku 4.2. Černá křivka představuje emoci „nadšení“ a oranžová křivka emoci „dlouhodobé nadšení“. Můžeme vidět, že grafy jsou si velice podobné a tedy, že záznam pomocí nástroje EPOC Events Generator funguje.




Obrázek 4.2: Porovnání zachycených emocí aplikací Emotiv Control Panel a nástrojem EPOC Events Generator - reálná data

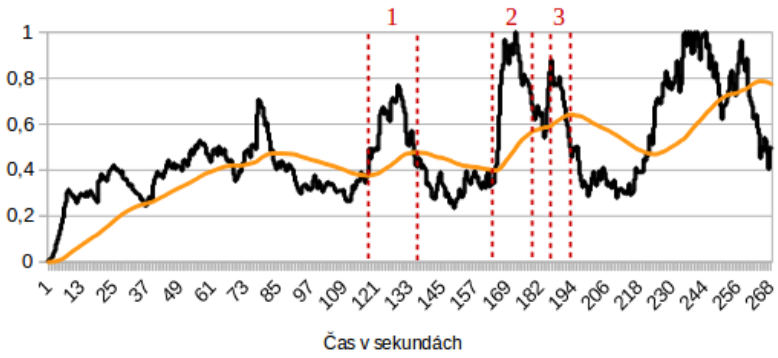
#### 4.3.3.1 Analýza zaznamenaných dat

Z vytvořeného grafu je vidět, že pozornost uživatele v průběhu sledování kolísala. Na 4 místech ale video uživatele zaujalo více než v jiných částech. Tento zájem je vyjádřen rostoucími hodnotami emoce „nadšení“ (černá křivka) zachycenými přístrojem. Čím vyšší tyto hodnoty jsou, tím více uživatele daná část videa zaujala. Naopak klesající hodnoty emoce „nadšení“ jsou známkou toho, že uživatelův zájem klesá.

Zatímco hodnoty emoce „nadšení“ jsou vyjádřením momentálního zájmu v daném okamžiku, přístroj z těchto hodnot navíc počítá další hodnoty, které vyjadřují nadšení v delším časovém horizontu (oranžová křivka). Pokud tedy tato křivka v nějakém úseku pouze roste, znamená to, že okamžiky, kdy se uživateli video líbilo, převládali na těmi kdy ne.

Z těchto dvou hodnot je tedy možné určit části, které uživatele zaujaly a které ne. Na grafu 4.3 jsou označeny části, které uživatele zaujaly. V rámci vyznačené oblasti je vidět, že oranžová křivka, vyjadřující dlouhodobé nadšení,

po celou dobu stoupá. Tento závěr je navíc podpořen i velikým výkyvem černé křivky zachycující hodnoty okamžitého nadšení.




Obrázek 4.3: Graf zachycující uživatelův zájem s vyznačenými úseky, které uživatele zaujaly

## 4.4 Shrnutí

Provedl jsem otestování implementované knihovny a aplikací Emotiv EPOC Logger a EPOC Events Generator. Testování proběhlo bez větších obtíží a bez chyb. To ale neznamená, že jsou aplikace bezchybné.

O otestování knihovny Emotiv EPOC Library se stará sada jednotkových testů. Aplikace Emotiv EPOC Logger a EPOC Events Generator byly testovány manuálně. U aplikace Emotiv EPOC Logger bylo testováno připojení, správnost zobrazovaných informací o stavu připojení a hlavní část tvořilo testování záznamu dat. U nástroje EPOC Events Generator bylo testováno ovládání externí aplikace pomocí klávesových zkratek generovaných na základě událostí ze zařízení Emotiv EPOC, napojení na aplikaci InBeat a záznam emocí uživatele.

Jediný problém se objevil při ovládání externí aplikace pomocí nástroje EPOC Events Generator s připojeným zařízením Emotiv EPOC. Pro toto ovládání jsou použity různém výrazy obličeje, které umí neuroheadset rozpoznat. Jelikož ale není zařízení primárně určeno ke snímaní pohybů v obličeji, dochází často k falešným událostem (především kvůli nižší kvalitě spojení). Zařízení Emotiv EPOC ale umožňuje v rámci modulu Cognitiv Suite natrénovat určité vzorce myšlení. Tyto vzorce poté vyhledává v zachycených signálech a generuje odpovídající kognitivní události (push, pull apod.).

Tento způsob ovládání pomocí natrénovaných stavů by byl asi vhodnější, protože se dá lépe kontrolovat a lze jej díky tréninku přizpůsobit konkrétnímu uživateli. Bohužel jsem již neměl čas tuto variantu implementovat.

# Závěr

Hlavním cílem práce bylo vytvoření knihovny pro práci se zařízením Emotiv EPOC a dvou aplikací využívající tuto knihovnu. Jednou z nich je aplikace Emotiv EPOC Logger pro záznam EEG a předzpracovaných signálů a druhá aplikace se jmenuje EPOC Events Generator a slouží k ovládání externích aplikací pomocí neuroheadestu Emotiv EPOC. Tyto cíle byly splněny.

V rámci diplomové práce jsem se seznámil se základy sběru informací o aktivitě lidského mozku (konkrétně se jednalo o metodu elektroencefalografie), získal základní povědomí o tom, jak fungují zařízení propojující mozek a počítač a podrobil jsem analýze neuroheadset EPOC a dodávané SDK.

Dále jsem s využitím Emotiv SDK navrhl a implementoval sadu funkcí určených pro zajištění komunikace a zpracování dat. Tuto knihovnu jsem poté využil při implementaci aplikací Emotiv EPOC Logger a EPOC Events Generator. První aplikace je určena pro záznam dat. Cílem druhé je umožnit ovládání externích aplikací a poskytnout data, na základě kterých bude možné provést analýzu zájmu uživatele o prezentované informace. Důkladným testování bylo ověřeno, že aplikace dělají to, co je od nich očekáváno.

Pro bezproblémové zprovoznění a používání implementovaných aplikací byla také vytvořena jednoduchá instalační a uživatelská příručka.

Práce s neuroheadsetem byla velmi zajímavá. Po prvním připojení přístroje a zapnutí aplikace Emotiv Control Panel jsem byl unesen možnostmi přístroje. Především mě zaujala postava robota, která opakovala výrazy mého obličeje. Postupem času jsem ale zjišťoval, že zařízení má své nedostatky. Z mého pohledu je hlavním nedostatkem nestabilní kvalita signálu z jednotlivých elektrod. Tyto elektrody je třeba před použitím navlhčit pomocí solného roztoku. Při nesprávném dávkování se ale stávalo, že některé elektrody měly velmi špatný signál, v extrémním případě nedošlo vůbec k připojení. Při delším používání také docházelo k vyschnutí roztoku a tím ke ztrátě spojení. Zřídka také docházelo k výpadkům v bezdrátovém spojení, ty byly většinou zapříčiněné slabší baterií přístroje.

# Literatura


[1] doc. MUDr. Jan Pokorný, DrSc.: Elektroencefalografie [online]. [cit. 2016-04-23]. Dostupné z: https://mynameiskavi.wordpress.com/2012/02/27/electroencephalography-or-eeg/

[2] Electroencephalography or EEG [online]. [cit. 2016-04-23]. Dostupné z: https://mynameiskavi.wordpress.com/2012/02/27/electroencephalography-or-eeg/

[3] EEG Machine [online]. [cit. 2016-04-23]. Dostupné z: http://www.madehow.com/Volume-7/EEG-Machine.html

[4] Houdková, L.: Tvorba zpětnovazebních her pro realizaci neurofeedback terapie. 2015. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/62078/F3-BP-2015-Houdkova-Lenka-Tvorba_zpetnovazebnich_her_pro_realizaci_neurofeedback_terapie.pdf

[5] Epilepsy Awareness Program - EEG Resources [online]. 2012, [cit. 2016-04-23]. Dostupné z: http://www.biomedresearches.com/root/pages/researches/epilepsy/eeg_resources.html

[6] Elektroencefalografie [online]. [cit. 2016-04-23]. Dostupné z: http://www.wikiskripta.eu/index.php/Elektroencefalografie

[7] Malmivuo, J.; Plonsey, R.: *Bioelectromagnetism*. New York: Oxford University Press, 1995, ISBN 01-950-5823-2. Dostupné z: http://www.bem.fi/book/13/13.htm

[8] Němec, P.: Využitelnost nervového ovládání počítače. 2012. Dostupné z: https://www.vse.cz/vskp/29778_vyuzitelnost_nervoveho_ovladani_pocitace

[9] Fouad, I. A.-A.; Mohammed, F. E.-Z.: Using Emotiv EPOC Neuroheadset To Acquire Data In Brain-Computer Interface. *International*

*Journal of Advanced Research 3*, , č. 11, 2015. Dostupné z: https://www.researchgate.net/publication/284178786_Using_Emotiv_EPOC_Neuroheadset_To_Acquire_Data_In_Brain-Computer_Interface

[10] Arafat, M. I.: Brain-Computer Interface: Past, Present and Future [online]. [cit. 2016-04-23]. Dostupné z: https://www.academia.edu/1365518/Brain_Computer_Interface_Past_Present_and_Future

[11] Vaněčková, T.: Vliv emoční stimulace na signál EEG. 2014. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=86508

[12] Živkov, M.: Faktory ovlivňující využitelnost nervového ovládání počítače v oblasti informačního managementu. 2013. Dostupné z: https://www.vse.cz/vskp/35812_faktory_ovlivnujici_vyuzitelnost_nervoveho_ovladani_pocitace_v_oblasti_informacniho_managementu

[13] Roman Čmejla: EEG I - textová verze 8. přednášky z BSG [online]. [cit. 2016-04-23]. Dostupné z: http://sami.fel.cvut.cz/bsg/BSG08.txt

[14] Mozkové vlny a rozšířené vědomí [online]. 2014, [cit. 2016-04-23]. Dostupné z: http://www.ac24.cz/zpravy-ze-sveta/3384-mozkove-vlny-a-rozsirene-vedomi

[15] Brain activity and how to measure it. [online]. [cit. 2016-04-23]. Dostupné z: https://mynameiskavi.wordpress.com/2012/02/28/brain-activity-and-how-to-measure-it/

[16] Hudges, J. R.: Gamma, fast, and ultrafast waves of the brain: Their relationships with epilepsy and behavior. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1525505008000127

[17] Sedlák, V.: Vývoj .NET aplikace pro EEG neuroheadset. 2014. Dostupné z: https://www.vse.cz/vskp/41751_vyvoj_net_aplikace_pro_eeg_neuroheadset

[18] Švejda, J.; Žák, R.; Jašek, R.: Zpracování mozkové aktivity v bci systémech [online]. *Trilobit - odborný vědecký časopis*, 2012. Dostupné z: http://trilobit.fai.utb.cz/zpracovani-mozkove-aktivity-v-bci-systemech_e498d494-fd80-4948-a8d6-f5f3720bba2d

[19] Komal Maloo: Brain computer interface [online]. [cit. 2016-04-23]. Dostupné z: http://www.slideshare.net/komal_maloo/brain-computer-interface-11822630

[20] Vokorokos, L.; Madoš, B.; Ádám, N.; aj.: DATA ACQUISITION IN NON-INVASIVE BRAIN-COMPUTER INTERFACE USING EMOTIV EPOC NEUROHEADSET. *Acta Electrotechnica et Informatica*, ročník 12, č. 1, 2012: s. 5–8. Dostupné z: https://www.researchgate.net/publication/269513314_Data_Acquisition_in_Non-Invasive_Brain-Computer_Interface_Using_Emotiv_Epoc_Neuroheadset

[21] Uday Saikia: Brain Computer Interfaces [online]. [cit. 2016-04-23]. Dostupné z: http://www.slideshare.net/udaysaikia/brain-computer-interfaces-by-uday-mc-gill

[22] NeuroSky: What Is BCI and How Did It Evolve? [online]. 2015, [cit. 2016-04-23]. Dostupné z: http://neurosky.com/2015/06/what-is-bci-and-how-did-it-evolve/

[23] Yadav, A. K.: Brain-Computer Interface [online]. [cit. 2016-04-23]. Dostupné z: http://files.psychickeobtezovani.webnode.cz/200000027-14cf21514c/Brain-Computer-Interface.pdf

[24] Šárka Landovská, M.: Možnosti využití nervového ovládání počítače ve sportu. 2013. Dostupné z: https://is.cuni.cz/webapps/zzp/detail/113142/

[25] Emotiv, Inc.: Wearables for your brain [online]. [cit. 2016-04-24]. Dostupné z: https://emotiv.com/

[26] NeuroSky, Inc.: NeuroSky [online]. [cit. 2016-04-24]. Dostupné z: http://neurosky.com/

[27] Lang, M.: Investigating the Emotiv EPOC for cognitive control in limited training time. 2012. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.304.3228&rep=rep1&type=pdf

[28] Group, T. E.: European Data Format. Dostupné z: http://www.edfplus.info/

[29] Pahorecký, P.: Biofeedback pomocí bezdrátového EEG. 2012. Dostupné z: http://cyberold.felk.cvut.cz/research/theses/papers/281.pdf

[30] Emotiv, Inc.: Emotiv Software Development Kit [online]. [cit. 2016-04-24]. Dostupné z: http://synapsets.etsmtl.ca/files/EmotivAPI-UserManual.pdf

[31] Ing. Petr Špaček, Ph.D.: Template Method - Strategy [online]. [cit. 2016-04-24]. Dostupné z: https://edux.fit.cvut.cz/courses/MI-DPO/_media/lectures/mi-dpo-02-template-method-strategy.pdf

[32] Ing. Petr Špaček, Ph.D.: Template Method - Strategy [online]. [cit. 2016-04-24]. Dostupné z: https://edux.fit.cvut.cz/courses/MI-DPO/_media/lectures/mi-dpo-04-iterator-composite-builder-observer.pd

[33] Kosek, J.: XML schémata[online]. [cit. 2016-05-01]. Dostupné z: http://www.kosek.cz/xml/schema/wxs.html

[34] Martin Malý: Web Sockets [online]. 2009, [cit. 2016-04-24]. Dostupné z: https://www.zdrojak.cz/clanky/web-sockets/

[35] Beust, C.: JCommander. Dostupné z: http://jcommander.org/

[36] Walnes, J.: X-Stream. Dostupné z: http://x-stream.github.io

[37] Foundation", T. A. S.: Apache Maven Project. Dostupné z: https://maven.apache.org/

Příloha A

# Seznam použitých zkratek

**API** Application Programming Interface.

**BCI** Brain–Computer Interface.

**CSV** Comma-separated values.

**EDF** European Data Format.

**EEG** Elektroencefalografie.

**EKG** Elektrokardiografie.

**EMG** Elektromyografie.

**GUI** Graphical User Interface.

**JNA** Java native access.

**JNI** Java native interface.

**JSON** JavaScript Object Notation.

**LSP** Liskov Substitution Principle.

**MEG** Magnetoencefalografie.

**REM** Rapid eye movement.

**SDK** Software Development Kit.

**XML** Extensible Markup Language.

Příloha **B**

# Obsah přiloženého CD

- `readme.txt` ..... obsah CD
- `bin` ..... adresář se spustitelnou formou implementace
- `docs` ..... dokumentace SDK a vytvořených aplikací
  - `EmotivEPOCLibrary` .. dokumentace knihovny Emotiv EPOC Library
  - `EmotivEPOCLogger` ..... dokumentace aplikace Emotiv EPOC Logger
  - `EmotivSDK` ..... dokumentace Emotiv SDK
  - `EPOCEventsGenerator` dokumentace nástroje EPOCEventsGenerator
- `lib` ..... JAR knihovny nutné pro sestavení vytvořených aplikací
- `src`
  - `EmotivEPOCLibrary` .. dokumentace knihovny Emotiv EPOC Library
  - `EmotivEPOCLogger` ..... dokumentace aplikace Emotiv EPOC Logger
  - `EmotivSDK` ..... dokumentace Emotiv SDK
  - `EPOCEventsGenerator` dokumentace nástroje EPOCEventsGenerator
- `text` ..... text práce
  - `DP_Munzar_Filip_2016.pdf` ..... text práce ve formátu PDF
- `test` ..... testovací data a výstupy testování
  - `EmotivEPOCLogger` ..... testovací data pro Emotiv EPOC Logger
  - `EPOCEventsGenerator` .... testovací data pro EPOCEventsGenerator
- `thesis_src` ..... zdrojová forma práce ve formátu LaTeX

Příloha C

# Instalační a uživatelská příručka

V této kapitole naleznete pokyny pro úspěšné nainstalování nástrojů a SDK společnosti Emotiv a aplikací Emotiv EPOC Logger a EPOC Events Generator.

**Aplikace byly spouštěny na operačním systému Linux, distribuce Ubuntu. Veškeré pokyny a příkazy jsou určeny pro tento operační systém. V jiných systémech se mohou lišit.**

**Pokud není uvedeno jinak, jsou všechny prezentované příkazy a cesty vztaženy ke kořenovému adresáři přiloženého CD.**

**Pro sestavení a spuštění aplikací Emotiv EPOC Logger a EPOC Events Generator je vyžadována Java ve verzi 8.**

## C.1 Nástroje a SDK společnosti Emotiv

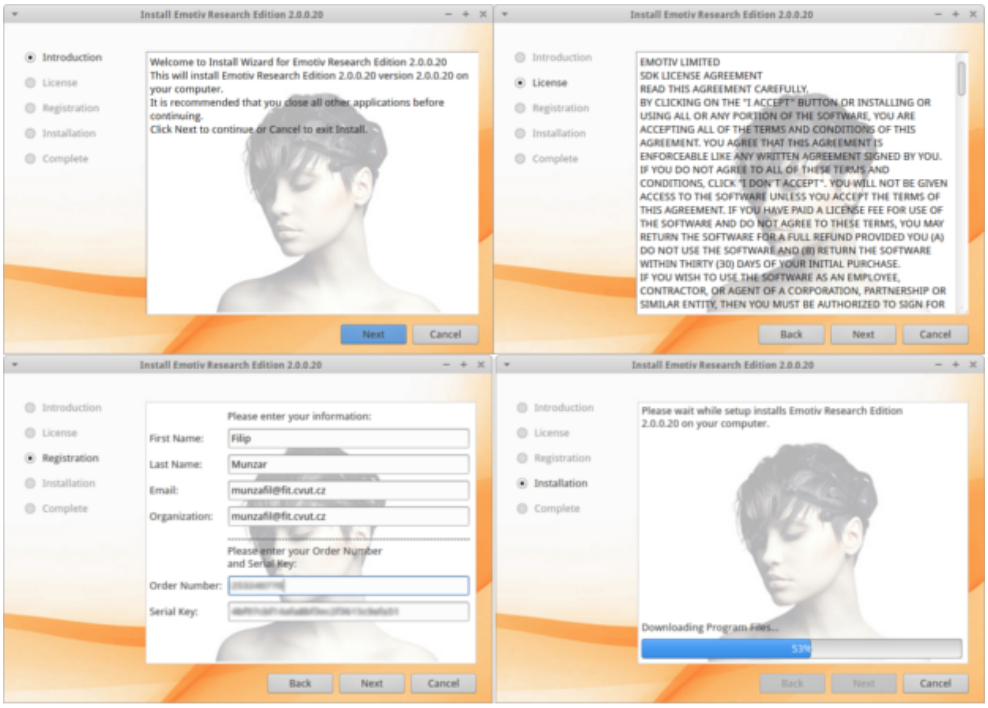
### C.1.1 Instalace

Společnost Emotiv dodává pro instalaci svých nástrojů shellový skript. Tento skript spustíme pomocí příkazového řádku. V mém případě pomocí příkazu `sudo ./EmotivResearchEditionInstall.sh` (skript je k dispozici na přiloženém CD).

Instalační skript si sám zajistí stažení a nastavení všech potřebných nástrojů a knihoven. Po nastavení prostředí se spustí instalačního průvodce (obrázek C.1), který uživatele seznámí s podmínkami použití a umožní zadat registrační údaje (jméno, příjmení, email, organizace, číslo objednávky a seriový klíč). Bez zadání těchto informací není možné instalaci dokončit! Po dokončení instalace je vše připraveno a můžeme začít s užíváním nainstalovaných nástrojů.

**Bez instalace SDK a ostatních nástrojů není možné používat aplikace Emotiv EPOC Logger a EPOC Events Generator!!**

Nainstalované nástroje a aplikace společnosti Emotiv jsou uloženy v domovské adresáři uživatele. V mém případě se jedná o adresář s názvem `Emotiv-`




Obrázek C.1: 4 kroky instalačního průvodce nástrojů společnosti Emotiv

`Research_2.0.0.20`. V tomto adresáři naleznete spouštěče jednotlivých aplikací, dokumentaci Emotiv SDK, příklad Emo skriptů a další potřebné knihovny a nástroje.

## C.2 Emotiv EPOC Library

### C.2.1 Sestavení knihovny

O sestavení knihovny ze zdrojových kódů se stará nástroj Maven [37]. Veškeré zdrojové kódy knihovny jsou ve složce `src/EmotivEPOCLibrary`. Je zde také uložen soubor `pom.xml`, který obsahuje instrukce pro nástroj Maven (s využitím jakých zdrojů provést kompilace, kam uložit výsledek apod.).

Před samotným sestavením knihovny je třeba přidat do lokálního repozitáře nástroje Maven knihovnu `j2ee-1.4.jar`, která je pro sestavení knihovny Emotiv EPOC Library nutná a není obsažena v online repozitářích. Tyto repozitáře jsou v podstatě databáze, ve kterých Maven vyhledává knihovny, které jsou potřebné pro správné sestavení. Potřebné knihovny jsou uvedeny v souboru `pom.xml`. Přidání do lokálního repozitáře provedeme příkazem, který je uveden na ukázce C.1.

Zdrojový kód C.1: Přidání knihovny j2ee do lokálního repositáře nástroje Maven

```
mvn install:install-file -Dfile=lib/j2ee-1.4.jar
    -DgroupId=javax.j2ee -DartifactId=j2ee -Dversion=1.4
    -Dpackaging=jar -DgeneratePom=true
```

Pro sestavení knihovny přejdeme složky `src/EmotivEPOCLibrary` a spustíme příkaz `mvn package`. Výsledný JAR soubor je poté k dispozici ve složce `src/EmotivEPOCLib/target`, jedná se o soubor `EmotivEPOCLibrary-1.0.jar`.

**Knihovnu není třeba sestavovat, sestavený JAR archiv je v dispozici ve složce `lib`.**

Setavenou knihovny je možné připojit k libovolnému projektu a začít využívat funkce, které obsahuje.

## C.3 Emotiv EPOC Logger

### C.3.1 Sestavení aplikace

Veškeré zdrojové kódy aplikace jsou ve složce `src/EmotivEPOCLogger`. Sestavení probíhá opět pomocí nástroje Maven.

Jelikož aplikace vyžaduje ke svému běhu knihovnu Emotiv EPOC Library, která není dostupná z online repozitářů, musíme ji pomocí příkazu na ukázce C.2 přidat do lokálního repozitáře.

Zdrojový kód C.2: Přidání knihovny Emotiv EPOC Library do lokálního repositáře nástroje Maven

```
mvn install:install-file -Dfile=lib/EmotivEPOCLibrary-1.0.jar
    -DgroupId=cz.cvut.fit.munzafil.emotiv
    -DartifactId=EPOCLibrary -Dversion=1.0 -Dpackaging=jar
    -DgeneratePom=true
```

Pro sestavení aplikace přejdeme složky `src/EmotivEPOCLogger` a spustíme příkaz `mvn package`. Výsledný JAR soubor je poté uložen ve složce `src/EmotivEPOCLogger/target`, soubor má název `EmotivEPOCLogger-1.0.jar`.

**Aplikaci není třeba sestavovat, sestavený JAR archiv je v dispozici ve složce `bin`.**

### C.3.2 Spuštění a obsluha aplikace

Aplikace je dostupná ve formě JAR achivu, který obsahuje všechny potřebné knihovny třetích stran. Ke spuštění tedy stačí pouze tento jeden soubor. Spuštění aplikace je zajištěno pomocí příkazu `java -jar bin/EmotivEPOCLogger-1.0.jar`.

Spustí se hlavní okno aplikace. Abychom mohli začít pořizovat záznam, je nutné připojit zařízení. Připojení je obsluhováno pomocí menu `Připojení`.

Pro připojení k aplikaci EmoComposer slouží položka „Připojit k EmoComposer“, položka „Připojit k EmoEngine“ je určena pro připojení k fyzickému zařízení. Ke zrušení současného spojení slouží položka „Odpojení“. Informace o současném spojení jsou zobrazeny v levé části okna.

Ve chvíli, kdy jsme připojeni, můžeme začít pořizovat záznam. Na výběr je ze 3 možností - EEG signály, Emo skript a kombinace obou možností. K volbě slouží sekce „Typ záznamu“. Po volbě typu záznamu určíme složku, kam chceme záznam uložit a název souboru (bez přípony, přípona se zvolí automaticky podle zvoleného typu). Záznam spustíme tlačítkem „Spustit záznam“. Stejným tlačítkem záznam zastavíme (po spuštění záznamu se text tlačítka změní na „Zastavit záznam“).

## C.4 EPOC Events Generator

### C.4.1 Sestavení aplikace

Veškeré zdrojové kódy jsou uloženy ve složce `src/EPOCEventsGenerator`. Sestavení probíhá stejně jako v předchozím případě pomocí nástroje Maven.

Stejně jako aplikace Emotiv EPOC Loggeer vyžaduje tato aplikace ke svému běhu knihovnu Emotiv EPOC Library. Tato knihovna by již měla být vložena v lokálním repozitáři, jelikož tam byla přidána při sestavování aplikace Emotiv EPOC Logger. Pokud ale knihovna Emotiv EPOC Library nebyla doposud vložena do repozitáře, je třeba spustit příkaz na ukázce C.2.

Pro samotné sestavení aplikace přejdeme do adresáře `src/EPOCEventsGenerator` a spustíme příkaz `mvn package`. Sestavený JAR soubor je ve složce `src/EPOCEventsGenerator/target`, soubor nese název `EPOCEventsGenerator-1.0.jar`.

Spustitelná aplikace je opět k dispozici ve složce `bin`.

### C.4.2 Spuštění a obsluha aplikace

Spuštění aplikace zajišťuje `java -jar bin/EPOCEventsGenerator-1.0.jar`. Na rozdíl od aplikace Emotiv EPOC Logger nemá tento nástroj grafické prostředí, ale je konfigurován pomocí argumentů příkazové řádky. Pro správné spuštění je tedy nutné uvést odpovídající argumenty. Argumenty, které je možné zadat, jsou uvedeny na ukázce C.3. Povinné parametry jsou označeny hvězdičkou.

Zdrojový kód C.3: Argumenty pro konfiguraci nástroje EPOC Events Generator

```
1  --connectionComposerHost
2     Adresa, na které běží EmoComposer (povinný, v případě
          connectionType=composer)
3  --connectionComposerPort
4     Port, na kterém běží EmoComposer (povinný, v případě
          connectionType=composer)
5  --connectionType *
6     Typ připojení (povolené hodnoty: composer, engine)
7  --emotionLoggerFilePath
8     Cesta pro uložení záznamu.
9  --emotionLoggerIsActive
10    Povolit záznam emocí?
11    Default: false
12 --inbeatIsActive
13    Povolit připojení k aplikaci InBeat?
14    Default: false
15 --inbeatServer
16    URL adresa websocketového serveru aplikace InBeat
17 --shortcutFilePath
18    Cesta k souboru s pravidly pro klávesové zkratky.
19 --shortcutIsActive
20    Povolit generování klávesových zkratek?
21    Default: false
```

Pro připojení k aplikaci InBeat, je třeba pomocí parametru `inbeatServer` specifikovat URL, na které běží websocketový server. V tomto případě je URL `ws://inbeat.eu/sync`.

Aplikace umožňuje generovat klávesové zkratky a tím ovládat externí aplikaci. Každý uživatel si může nadefinovat vlastní pravidla pro generování zkratek, stačí vytvořit nový XML dokument, který bude doržovat syntaxi uvedenou v kapitole . Tento dokument se poté předá aplikaci pomocí argumentu `shortcutFilePath`. Jako inspirace může posloužit dokument `shortcut_rules_test.xml` ve složce `test/EPOCEventsGenerator/shortcut`.

Poslední možností, kterou nástroj EPOC Events Generator poskytuje, je záznam emocí uživatele. Záznam je ukládán do CSV souborů. Složka, kam se mají soubory ukládat, je specifikována pomocí parametru `shortcutFilePath`.

O běhu nástroje je uživatel informován pomocí výpisů na standardní výstup (v tomto případě na příkazový řádek).

Příloha D

# Emotiv SDK

## D.1 Události EmoEvents

Tabulka D.1: EmoEngine události

| **EmoEngine události** | **HEX hodnota** | **Popis** |
|---|---|---|
| EE_UserAdded | 0x0010 | registrace nového uživatele do seznamu uživatelů |
| EE_UserRemoved | 0x0020 | odebrání uživatele ze seznamu |
| EE_EmoStateUpdated | 0x0040 | k dispozici je nový stav ke zpracování |
| EE_ProfileEvent | 0x0080 | odpověď na žádost o získání profilu uživatele |
| EE_CognitivEvent | 0x0100 | byla zaznamenána nová kognitivní událost |
| EE_ExpressivEvent | 0x0200 | byly zaznamenána nová expresivní událost (výraz obličeje) |
| EE_InternalStateChanged | 0x0400 | používá se pouze v Emotiv Contol Panel |
| EE_EmulatorError | 0x0001 | interní chyba zařízení |

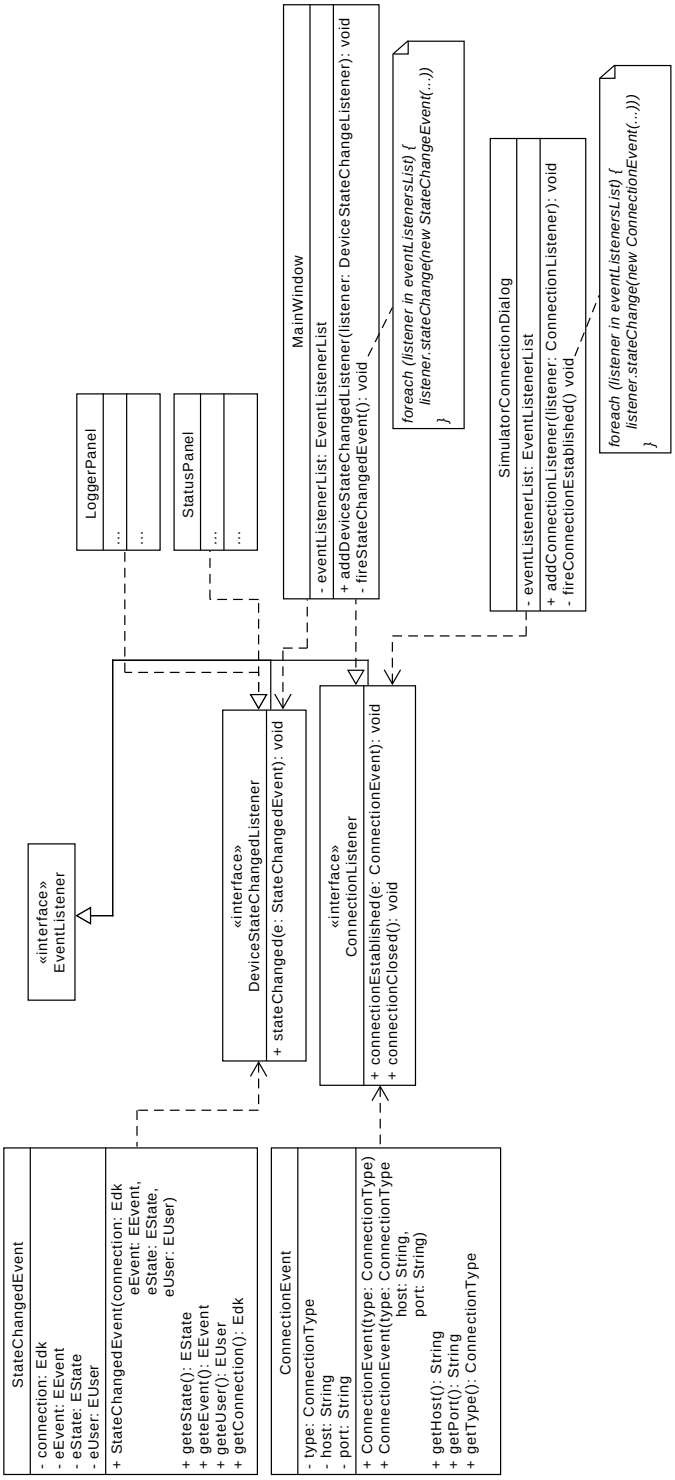
## D.2 Chybové kódy

Tabulka D.2: EmoEngine chybové kódy

| **Chybový kód** | **HEX hodnota** | **Popis** |
|---|---|---|
| EDK_OK | 0x0000 | úspěch |
| EDK_UNKNOWN_ERROR | 0x0001 | kritická chyba |
| EDK_INVALID_PROFILE_ARCHIVE | 0x0101 | načtená data neodpovídají struktuře uživatelského profilu |
| EDK_NO_USER_FOR_BASE_PROFILE | 0x0102 | vrácen při snaze získat uživatelské ID základního profilu |
| EDK_CANNOT_ACQUIRE_DATA | 0x0200 | EmoEngine není schopen získat data ze zařízení Emotiv EPOC |
| EDK_BUFFER_TOO_SMALL | 0x0300 | vrácen pokud připravený buffer není dostatečně veliký pro uložení profilu |
| EDK_OUT_OF_RANGE | 0x0301 | jeden z parametrů funkce je mimo povolený rozsah |
| EDK_INVALID_PARAMETER | 0x0302 | jeden z parametrů není platný (např. Null pointer) |
| EDK_PARAMETER_LOCKED | 0x0303 | hodnota parametru je uzamčena a nemůže být modifikována |
| EDK_COG_INVALID_TRAINING_ACTION | 0x0304 | není možné provést trénink dané akce |
| EDK_COG_INVALID_TRAINING_CONTROL | 0x0305 | špatně nastavený kontrolní příznak tréninku |
| EDK_COG_INVALID_ACTIVE_ACTION | 0x0306 | špatně zvolená akce pro trénování |
| EDK_COG_EXCESS_MAX_ACTIONS | 0x0307 | bylo dosaženo maximální povoleného počtu souběžných akcí |
| EDK_EXP_NO_SIG_AVAILABLE | 0x0308 | natrénovaný vzorek není momentálně k dispozici |
| EDK_INVALID_USER_ID | 0x0400 | špatné ID uživatele |
| EDK_EMOENGINE_UNINITIALIZED | 0x0500 | vrácen ve chvíli, kdy nebylo vytvořeno spojení se zařízením |
| EDK_EMOENGINE_DISCONNECTED | 0x0501 | spojení bylo ztraceno |
| EDK_EMOENGINE_PROXY_ERROR | 0x0502 | vrácen ve chvíli, kdy se nepodaří spojení s EmoComposer |
| EDK_NO_EVENT | 0x0600 | ve frontě není žádná neobsloužená událost |
| EDK_GYRO_NOT_CALIBRATED | 0x0700 | gyroskop nebyl zkalibrován |
| EDK_OPTIMIZATION_IS_ON | 0x0800 | probíhá optimalizace algoritmu |

Příloha E

# Diagramy




Obrázek E.1: Diagram zachycující třídy zajišťující komunikaci v aplikaci Emotiv EPOC Logger

Příloha F

# XSD schéma

Zdrojový kód F.1: XSD schéma definující strukturu XML pro definování klávesových zkratek

```
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="actions">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="action" maxOccurs="unbounded"
            minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="shortcut">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="value"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element type="xs:string" name="name" />
              <xs:element type="xs:float" name="value" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Příloha G

# Zdrojové kódy

Zdrojový kód G.1: Komunikace panelů StatusPanel, LoggerPanel a hlavního okna aplikace

```
/* Vytvoření vlákna MainRunnable - soubor MainWindow.java*/
public void connectionEstablished(ConnectionEvent e) {
  try {
    ConnectionFactory cf = new ConnectionFactory();
    ConnectionParameters params = new ConnectionParameters();

    if (e.getType() == ConnectionType.COMPOSER) {
      params.setHost(e.getHost());
      params.setPort(Short.parseShort(e.getPort()));
    }

    this.connection = cf.create(e.getType(),
        params).establishConnection();

    MainRunnable mainRunnable = new
        MainRunnable(this.connection);
    mainRunnable.addDeviceStateChangedListener(this.panel_status);
    mainRunnable.addDeviceStateChangedListener(this.panel_logger);

    this.mainThread = new MainThread(mainRunnable);
    this.mainThread.start();

  } catch (ConnectionException | UnknownConnectionType e1) {
    this.panel_status.setStatusText(e1.getMessage(), true);
  }
}

/* Zaregistrování posluchače - soubor MainRunnable.java */
public void addDeviceStateChangedListener(
    DeviceStateChangedListener listener) {
  this.eventListenerList.add(DeviceStateChangedListener.class,
      listener);
}

/* Informovaní registrovaných posluchačů - souboru
    MainRunnable.java */
protected void fireStateChangedEvent() {
  Object[] listeners = eventListenerList.getListenerList();
  for (int i = 0; i < listeners.length; i += 2) {
    if (listeners[i + 1] instanceof DeviceStateChangedListener) {
      ((DeviceStateChangedListener) listeners[i +
          1]).stateChanged(new StateChangedEvent(this.eEvent,
          this.eState, this.user));
    }
  }
}
```

Zdrojový kód G.2: Reakce na změnu stavu v panelech StatusPanel a LoggerPanel

```
/* Implementace metody stateChanged ve třídě StatusPanel */
public void stateChanged(StateChangedEvent e) {
  // načtení EState
  EState eState = e.geteState();

  // Načtení kvality připojení jednotlivých senzorů
  int [] f = eState.getContactQuality();

  // překreslení senzorů
  for(int i = 0; i < f.length; i++) {
    if ( channelSignalStatusMap.containsKey(i) ) {
      channelSignalStatusMap.get(i).updateStatus(f[i]);
      channelSignalStatusMap.get(i).setName((new
          Channels()).getChannelName(i));
    }
  }

  // překreslení ostatních informací
  this.label_wireless.setText(Integer.toString(
      eState.getWirelessSignal()));
  DecimalFormat df = new DecimalFormat("0.###");
  this.label_time.setText(df.format(eState.getRunningTime()) + "
      sec.");
  this.label_battery.setText(Integer.toString(
      eState.getBatteryLevel()));
}

/* Implementace metody stateChanged ve třídě LoggerPanel */
public void stateChanged(StateChangedEvent e) {
  if (isLogStarted) {
    try {
      // načtení struktur
      EState eState = e.geteState();
      EEvent eEvent = e.geteEvent();
      EUser user = e.getUser();

      // provedení záznamu
      logger.log(eState, eEvent, user);

      // informování uživatele pomocí textové oblasti
      this.appendLog(eState.toString());
    } catch (IOException e1) {
      this.appendLog(e1.getMessage());
    }
  }
}
```

Příloha H

# Testovací data a výstupy testování

## H.1 Emotiv EPOC Logger

Zdrojový kód H.1: Porovnání emo skriptu vytvořeného pomocí Emotiv EPOC Logger s originálním skriptem

```
<!-- pro jednoduchost jsou vynechány hlavičky a uvádím pouze čá
    st těchto skriptů -->
<!-- obsah souboru test_template.emo - originální soubor použitý
    v aplikaci EmoComposer-->
<time value="2s4t">
  <affectiv event="engagement_boredom" value="0.1" />
  <signal_quality value="1,1,0,1,0.75,0.5,0.25,1,0,
      0.75,1,1,0,1,0.5,1,1,0" />
</time>
<time value="3s6t">
  <affectiv event="engagement_boredom" value="0.2" />
  <signal_quality value="1,1,0.5,0.75,0,0.5,0,0.75,
      1,0.5,1,1,1,0.75,0.75,1,0,0.5" />
</time>
<time value="5s10t">
  <affectiv event="engagement_boredom" value="0.3" />
  <expressiv_upperface event="eyebrow_raised" value ="0.85" />
  <expressiv_lowerface event="clench" value ="0.85" />
  <signal_quality value="1,1,0.25,0.5,0.25,0.75,1,1,
      0.25,0.5,0.25,0.5,0.25,0,0.25,1,1,0.25" />
</time>


<!-- obsah souboru test_template_zaznam.emo - soubor zaznamenaný
    aplikací Emotiv EPOC Logger z dat generovaných pomocí
    skriptu test_template.emo -->
<time value="2s3t">
  <affectiv event="engagement_boredom" value="0,100000" />
  <signal_quality value="1.0,1.0,0.25,1.0,0.75,0.5,0.25,
      1.0,0.25,0.75,1.0,1.0,0.25,1.0,0.5,1.0,1.0,0.25" />
</time>
<time value="3s5t">
  <affectiv event="engagement_boredom" value="0,200000" />
  <signal_quality value="1.0,1.0,0.5,0.75,0.25,0.5,0.25,
      0.75,1.0,0.5,1.0,1.0,1.0,0.75,0.75,1.0,0.25,0.5" />
</time>
<time value="5s9t">
  <expressiv_lowerface event="clench" value="0,850000" />
  <expressiv_upperface event="eyebrow_raised" value="0,850000" />
  <affectiv event="engagement_boredom" value="0,300000" />
  <signal_quality value="1.0,1.0,0.25,0.5,0.25,0.75,1.0,
      1.0,0.25,0.5,0.25,0.5,0.25,0.25,0.25,1.0,1.0,0.25" />
</time>
```

Zdrojový kód H.2: Část emo skriptu vytvořeného z dat získaných ze zařízení Emotiv EPOC pomocí Emotiv EPOC Logger

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE EML>
<EML version="1.0" language="en_US">
  <config>
    <autoreset value ="1" group="expressiv_eye" event="blink" />
    <autoreset value ="1" group="expressiv_eye"
        event="wink_left"/>
    <autoreset value ="1" group="expressiv_eye"
        event="wink_right"/>
  </config>
  <sequence>
    ....
    <time value="102s3t">
      <expressiv_upperface event="furrow" value="0,114455" />
      <affectiv event="excitement_long_term" value="0,513602" />
      <affectiv event="excitement_short_term" value="0,505174" />
      <affectiv event="engagement_boredom" value="0,548642" />
      <signal_quality value="1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
          1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,0.0,0.0" />
    </time>
    <time value="102s6t">
      <affectiv event="excitement_long_term" value="0,513602" />
      <affectiv event="excitement_short_term" value="0,505174" />
      <affectiv event="engagement_boredom" value="0,548642" />
      <signal_quality value="1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
          1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,0.0,0.0" />
    </time>
    <time value="102s18t">
      <expressiv_lowerface event="smirk_right" value="0,460452"
          />
      <affectiv event="excitement_long_term" value="0,512386" />
      <affectiv event="excitement_short_term" value="0,493017" />
      <affectiv event="engagement_boredom" value="0,548642" />
      <signal_quality value="1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
          1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,0.0,0.0" />
    </time>
    ....
  </sequence>
</EML>
```

## H.2 EPOC Events Generator

### H.2.1 Generování klávesových zkratek

Zdrojový kód H.3: Emo skript pro testování ovládání pomocí klávesových zkratek

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE EML>
<EML version="1.0" language="en_US">
  <config>
    <autoreset value ="1" group="expressiv_eye" event="blink" />
    <autoreset value ="1" group="expressiv_eye"
        event="wink_left"/>
    <autoreset value ="1" group="expressiv_eye"
        event="wink_right"/>
  </config>
  <sequence>
    ...
    <!-- simulace klávesy VPRAVO -->
    <time value="4s6t">
      <expressiv_eye event="wink_right" value="1" />
      <signal_quality value="1,1,0.5,0.75,0,0.5,0,0.75,
          1,0.5,1,1,1,0.75,0.75,1,0,0.5" />
    </time>
    <!-- simulace klávesy VLEVO -->
    <time value="6s10t">
      <expressiv_eye event="wink_left" value="1" />
      <expressiv_upperface event="eyebrow_raised" value ="0.85"
          />
      <expressiv_lowerface event="clench" value ="0.85" />
      <signal_quality value="1,1,0.25,0.5,0.25,0.75,1,1,
          0.25,0.5,0.25,0.5,0.25,0,0.25,1,1,0.25" />
    </time>
    <!-- simulace klávesy VPRAVO -->
    <time value="8s6t">
      <expressiv_eye event="wink_right" value="1" />
      <affectiv event="excitement_short_term" value="0.5" />
      <signal_quality value="1,1,0.25,0.75,1,1,0,0,1,0.5,
          1,0,0.5,0.5,0.75,0,0,0" />
    </time>
    ...
    <!-- simulace klávesy VLEVO -->
    <time value="14s1t">
      <expressiv_eye event="wink_left" value="1" />
      <affectiv event="excitement_short_term" value="0.9" />
      <affectiv event="excitement_long_term" value="0.323" />
      <signal_quality value="1,1,0.25,0,1,1,1,1,0.5,0,0.25,
          1,0.5,0.5,0,1,0.75,0.75" />
    </time>
  </sequence>
</EML>
```

Zdrojový kód H.4: Soubor s definicí pravidel pro generování klávesových zkratek

```
<?xml version="1.0" ?>
<actions>
    <action>
        <shortcut>
            <value>RIGHT</value>
        </shortcut>
        <name>wink_right</name>
        <value>0.0</value>
    </action>
    <action>
        <shortcut>
            <value>LEFT</value>
        </shortcut>
        <name>wink_left</name>
        <value>0.0</value>
    </action>
</actions>
```

### H.2.2 Komunikace s aplikací InBeat

Zdrojový kód H.5: Emo skript pro testování komunikace s aplikací InBeat

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE EML>
<EML version="1.0" language="en_US">
  ...
  <sequence>
    <!-- událost pro spuštění videa -->
    <time value="1s15t">
      <expressiv_eye event="blink" value="1" />
      <signal_quality value="1,1,1,0,0.5,1,0.75,0,
          0,0.75,0.5,0.25,0.5,0,0.25,0,0.5,0.5" />
    </time>
    <!-- událost pro spuštění zastavení videa -->
    <time value="5s4t">
      <expressiv_lowerface event="clench" value ="0.71" />
      <signal_quality value="1,1,0,1,0.75,0.5,0.25,
          1,0,0.75,1,1,0,1,0.5,1,1,0" />
    </time>
    <!-- sníží hlasitost -->
    <time value="7s6t">
      <expressiv_eye event="wink_left" value="1" />
      <signal_quality value="1,1,0.5,0.75,0,0.5,0,
          0.75,1,0.5,1,1,1,0.75,0.75,1,0,0.5" />
    </time>
    ...
    <!-- zvýší hlasitost -->
    <time value="11s26t">
      <expressiv_eye event="wink_right" value="1" />
      <signal_quality value="1,1,0,1,0.5,1,0.25,0.75,
          1,0,0.75,0.75,0.75,0.5,1,0.5,0.75,0.75" />
    </time>
    <!-- událost pro spuštění videa -->
    <time value="12s8t">
      <expressiv_eye event="blink" value="1" />
      <signal_quality value="1,1,0.75,0.5,0.25,0.75,0.25,
          1,0.75,0,0.75,0.25,0.5,1,0.5,0,0.25,0.75" />
    </time>
    ...
    <!-- další video -->
    <time value="18s4t">
      <expressiv_lowerface event="smirk_right" value ="0.75" />
      <signal_quality value="1,1,0,1,0.75,0.5,0.25,1,
          0,0.75,1,1,0,1,0.5,1,1,0" />
    </time>
    ...
  </sequence>
</EML>
```

### H.2.3 Záznam emocí uživatele

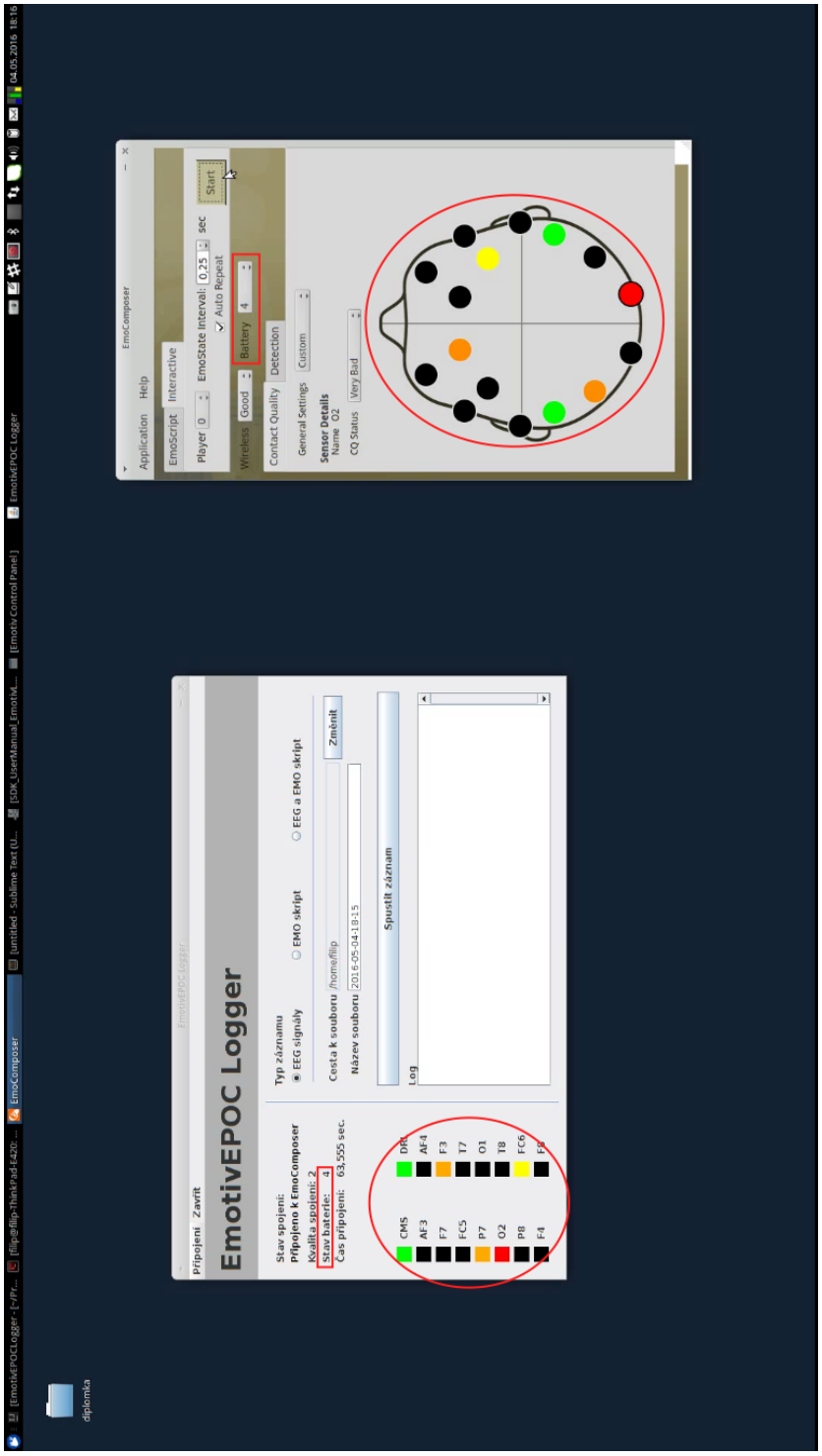
Zdrojový kód H.6: Část emo skriptu generující testovací data pro záznam emocí uživatele

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE EML>
<EML version="1.0" language="en_US">
  <sequence>
    <time value="2s4t">
      <affectiv event="excitement_short_term" value="0.25" />
      <affectiv event="excitement_long_term" value="0.02" />
      <signal_quality value="1,1,0,1,0.75,0.5,0.25,1,0,
          0.75,1,1,0,1,0.5,1,1,0" />
    </time>
    <time value="4s6t">
      <affectiv event="excitement_short_term" value="0.6" />
      <affectiv event="excitement_long_term" value="0.25" />
      <signal_quality value="1,1,0.5,0.75,0,0.5,0,0.75,
          1,0.5,1,1,1,0.75,0.75,1,0,0.5" />
    </time>
    <time value="6s10t">
      <affectiv event="excitement_short_term" value="0.5" />
      <affectiv event="excitement_long_term" value="0.2" />
      <signal_quality value="1,1,0.25,0.5,0.25,0.75,1,1,
          0.25,0.5,0.25,0.5,0.25,0,0.25,1,1,0.25" />
    </time>
    ...
    <time value="11s8t">
      <affectiv event="excitement_short_term" value="0.9" />
      <affectiv event="excitement_long_term" value="0.323" />
      <signal_quality value="1,1,0.75,0.5,0.25,0.75,0.25,
          1,0.75,0,0.75,0.25,0.5,1,0.5,0,0.25,0.75" />
    </time>
    <time value="12s16t">
      <affectiv event="excitement_short_term" value="0.75" />
      <affectiv event="excitement_long_term" value="0.289" />
      <signal_quality value="1,1,0,0.25,0.25,0.25,0,1,0,
          0.75,0.5,0.75,0.25,1,0,0.75,0.5,0" />
    </time>
    <time value="14s1t">
      <affectiv event="excitement_short_term" value="0.75" />
      <affectiv event="excitement_long_term" value="0.28" />
      <signal_quality value="1,1,0.25,0,1,1,1,1,0.5,0,
          0.25,1,0.5,0.5,0,1,0.75,0.75" />
    </time>
  </sequence>
</EML>
```

Příloha I

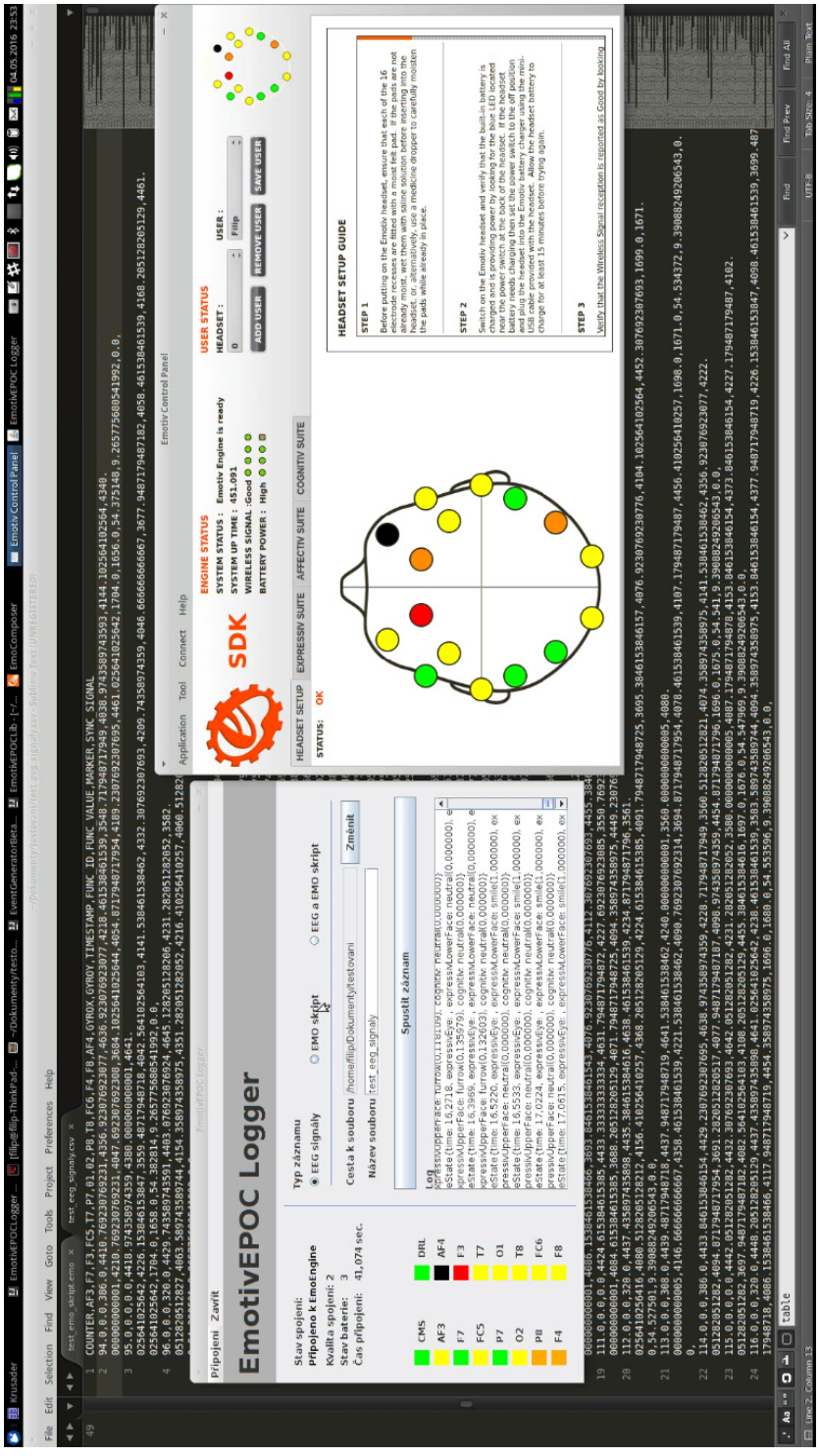
# Ukázky video záznamů z testování

Video záznamy byly pořizovány na monitoru s rozlišením 1920x1080 bodů, proto prosím omluvte sníženou kvalitu obrázků. Videa v plné kvalitě jsou na přiloženém CD.

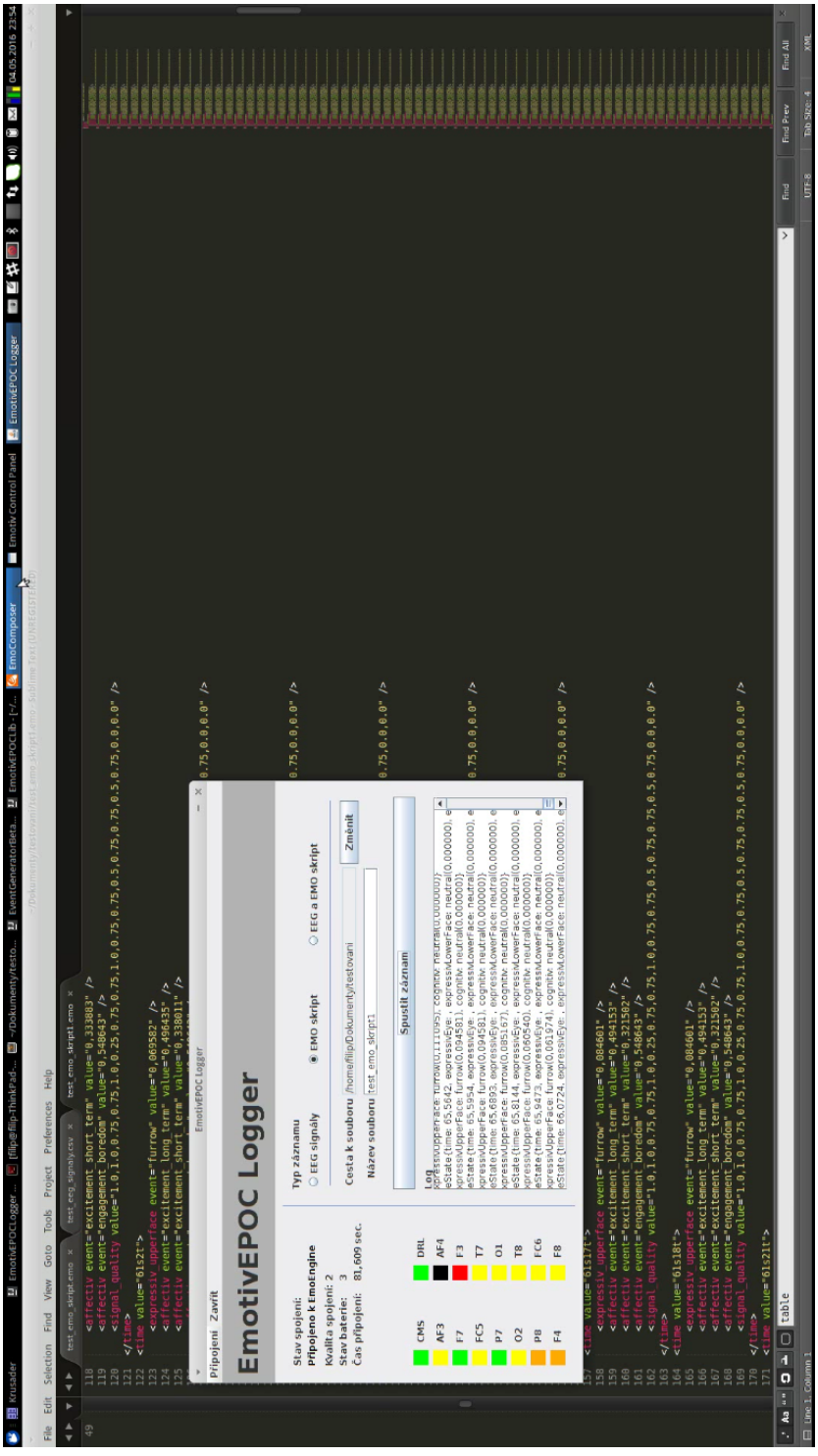



Obrázek I.1: Kontrola správnosti informací o naváženém spojení v aplikace Emotiv EPOC Logger - připojení k EmoComposer

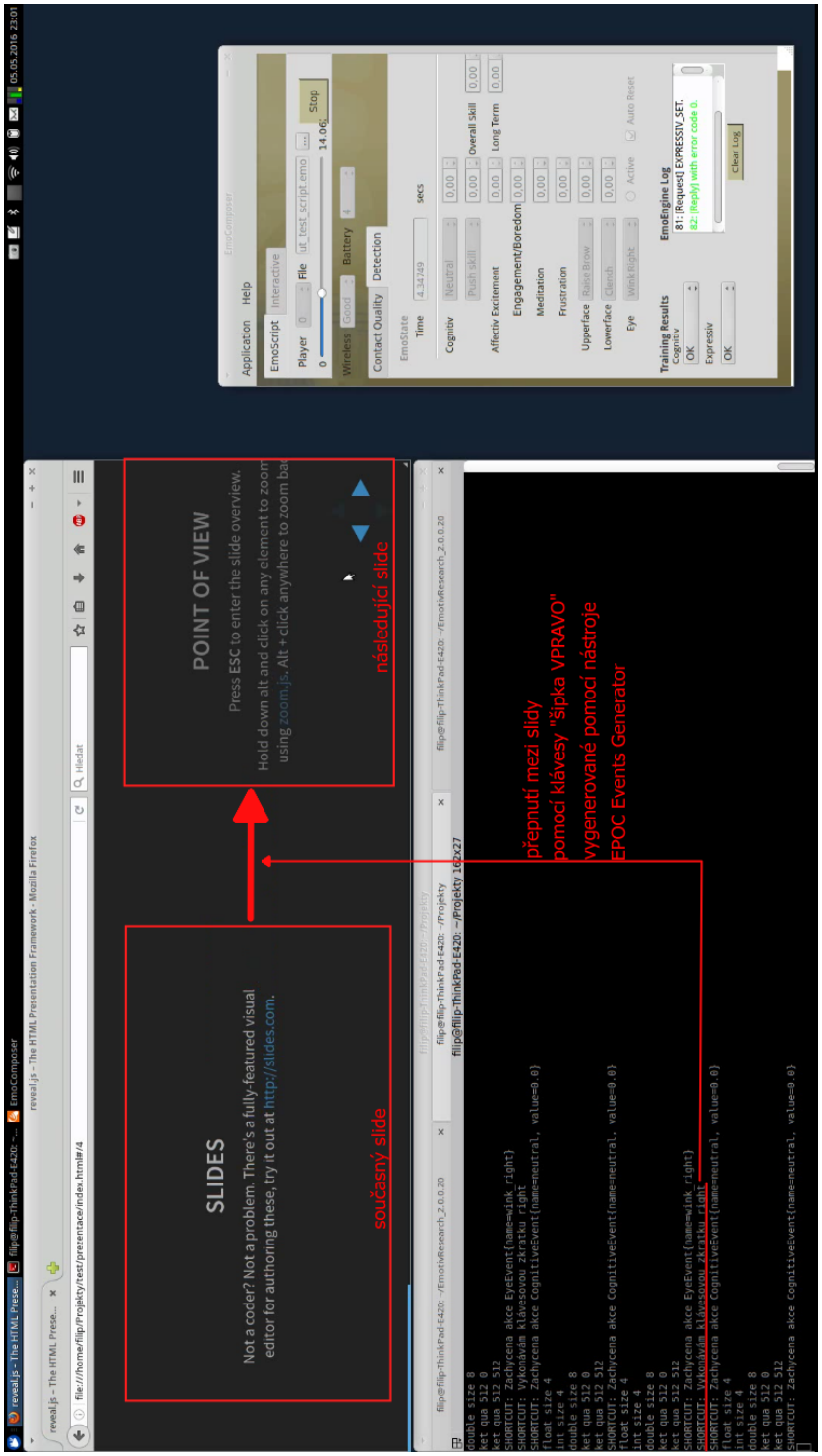

Obrázek I.2: Kontrola správnosti informací o naváženém spojení v aplikace Emotiv EPOC Logger - připojení k Emotiv EPOC



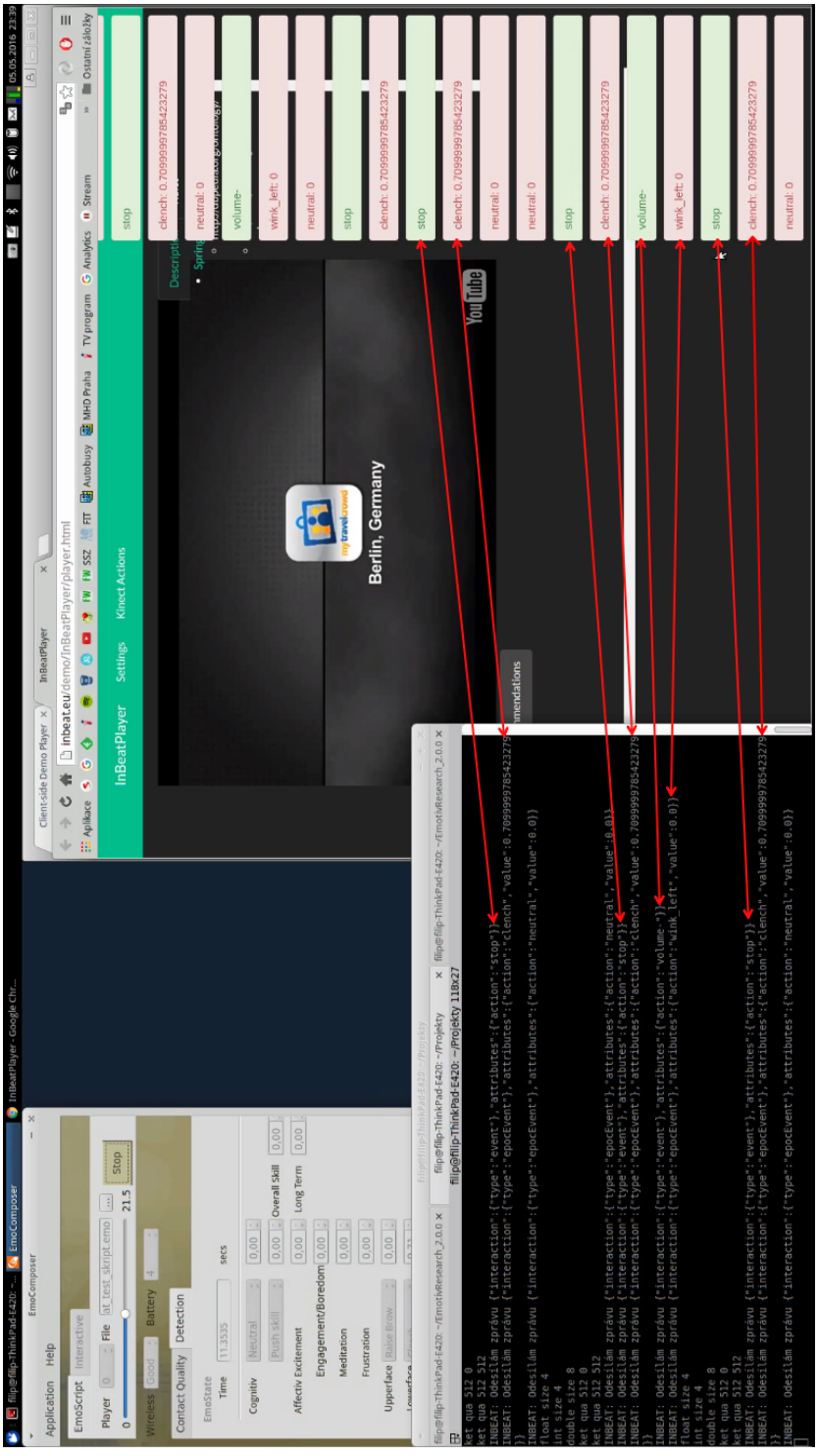

Obrázek I.3: Záznam EEG dat pomocí Emotiv EPOC Logger (v pozadí je soubor se zaznamenanými EEG daty)




Obrázek I.4: Záznam Emo skriptu pomocí Emotiv EPOC Logger (v pozadí je soubor se zaznamenaným Emo skriptem)




Obrázek I.5: Ovládání prezentace pomocí klávesových zkratek generovancých nástroje EPOC Events Generator




Obrázek I.6: Odesílání zpráv aplikaci InBeat