Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Realizace jednoduché BCI hry pro trénink hokejistů s podporou akcelerometru

Plzeň 2020 Martin Folejtar

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin FOLEJTAR**
Osobní číslo:	**A17B0206P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Realizace jednoduché BCI hry pro trénink hokejistů s podporou akcelerometru**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se:
   a) S měřením mozkové aktivity v neuroinformatické laboratoři na KIVu.
   b) Se sledováním pohybové aktivity v hokejovém kempu.
2. Prostudujte:
   a) Vlastnosti snímače Mindwave Mobile od firmy Neurosky.
   b) Vlastnosti dostupného snímače pro měření akcelerace.
3. Navrhněte:
   a) Jednoduché rozhraní mozek počítač (BCI) pro zařízení uvedená v bodě 2a.
   b) Jednoduché rozhraní akcelerometr počítač (ACI) pro zařízení uvedená v bodě 2b.
4. Vytvořte aplikaci pro trénink hokejistů na základě návrhu v bodě 3.
5. Otestujte ACI a BCI aplikaci na reálných, nebo alespoň realistických scénářích a zhodnoťte dosažené výsledky práce.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Petr Brůha**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **7. října 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

Radová

L.S.

Západočeská univerzita v Plzni
Fakulta aplikovaných

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 15. října 2019

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4. května 2020

Martin Folejtar

Abstract

This bachelor thesis focuses on the use of computing systems in order to improve sport-related skills that involve a hockey stick that will take effect on sports such as floorball, hockey, or hockeyball. The thesis is composed of three parts. The first part regards theoretical knowledge in the field of science neurology and neuroinformatics, movement activity with a hockey stick, and the specification of both headband MindWave Mobile 2 and digital sensor WT61C, which is used for acceleration measurement. The second part deals with the design and implementation of a simple computer trainer, including connecting sensors BCI and ACI. The last part addresses the use of the application itself and testing it in hockey training.

Abstrakt

Bakalářská práce se zabývá propojením výpočetní techniky s tréninkem dovedností ve sportovní sféře zaměřené na sporty s hokejovou holí (hokej, florbal či hokejbal). Práce se skládá ze tří částí. První pojednává o teoretických poznatcích ve vědním oboru neurologie a neuroinformatiky, o pohybové aktivitě na tréninku s hokejovou holí, vlastnostech čelenky MindWave Mobile 2 a digitálním snímači WT61C pro měření akcelerace. Druhá část práce pojednává o návrhu a implementaci jednoduchého počítačového trenéra s propojením snímačů (BCI a ACI). Třetí část se zabývá využitím aplikace a jejím testování na hokejovém tréninku.

Obsah

1 Úvod

Dnešní doba je velice vyspělá, existuje mnoho moderních technologií, které nabízí řadu možností. Technologie se tak staly běžnou součástí života a jsou využívány v různých odvětvích. Jedním z nich je i sportovní odvětví, které zažívá s výpočetní technikou velký rozmach. Na trhu stále nejsou aplikace, které by podporovaly jak vývoj sportovních dovedností, tak zlepšení kognitivních funkcí sportovců. S jedním nápadem na aplikaci přišli hokejoví trenéři a manažeři Lukáš a Ondra Zdrhovi z LZ hokejové školy. Rádi by poskytli svým svěřencům jednoduchou aplikaci pro **rozvoj herní inteligence i z pohodlí domova.** Aplikaci bude vizualizovat animace pohybů puků, které bude hráč provádět s hokejovou holí a pukem před obrazovkou. Zároveň bude **poskytovat zpětnou vazbu ze snímačů**, které s ní budou propojeny. **Cílem aplikace není zlepšit techniku, kterou se hráč naučí na tréninku, ale naučit myslet během pohybu.** Z jejich pohledu se na trhu neobjevila žádná podobná aplikace, která by sloužila k těmto účelům. Jejich nápad se mi zdál natolik reálný, že jsem se rozhodl, po několika konzultacích s vedoucím práce Ing. Petrem Brůhou a trenéry, si danou problematiku zvolit za práci. S vedoucím práce jsme specifikovali zadání dle našich a jejich požadavků.

Cílem je **vytvořit aplikaci s jejich požadavky, která bude hráči poskytovat zpětnou vazbu z daného pohybu, či tréninku před obrazovkou.** Každý sportovec, který bude chtít aplikaci používat, bude muset mít své přístupové údaje.

Práce se dělí na tři části. První část se zabývá měřením mozkové aktivity v neuroinformatické oblasti a zkoumáním pohybových aktivit hráčů na tréninku. Informace, která by měla z této části vyplynout, je odpověď na otázku: Jakým způsobem a co, lze sledovat na pohybu, aby hráči byla poskytnuta zpětná vazba? Dále se zaměřuji na parametry a vlastnosti vybraného snímače WT61C pro měření akcelerace a také na čelenku MindWave Mobile 2 pro měření mozkové aktivity. Kombinací těchto dvou zařízení bude vytvořen prototyp, který hráči poskytne zpětnou vazbu z dovednostního tréninku. Druhá část se zabývá samotnou implementací a návrhem prototypu aplikace. Třetí část pojednává o testování vytvořeného prototypu a zhodnocení dosažených výsledků.

2 Neuroinformatická laboratoř na KIVu

Dle zainteresování práce (snímač MindWave Mobile 2) do vědního oboru neuroinformatiky je důležité znát, jak základy v této oblasti (kapitola 3), tak seznámení s měřením mozkové aktivity v neuroinformatické laboratoři. Zmíněné informace jsou převzaté především z článku [7] od odborníků z katedry informatiky.

V laboratoři na KIVu pro neuroinformatiku dochází k návrhu a realizaci elektrofyziologických experimentů, sběru, uchování a sdílení experimentálních dat či metadat. Tyto procesy a zveřejňování jejich výsledků jsou natolik náročné, že je potřeba je podpořit vhodnou infrastrukturou, aby se zvýšila efektivita výzkumníků. Laboratoř je vybavena několika komerčními a vlastními hardwarovými zařízeními a také softwarovými nástroji. Kromě základních zařízení a softwaru se v laboratoři nachází zvuková a elektricky stíněná kabina, auto simulátor včetně kabiny, palubní desky, projektoru, volantu a pedálů připojených k počítači pro simulaci řízení.

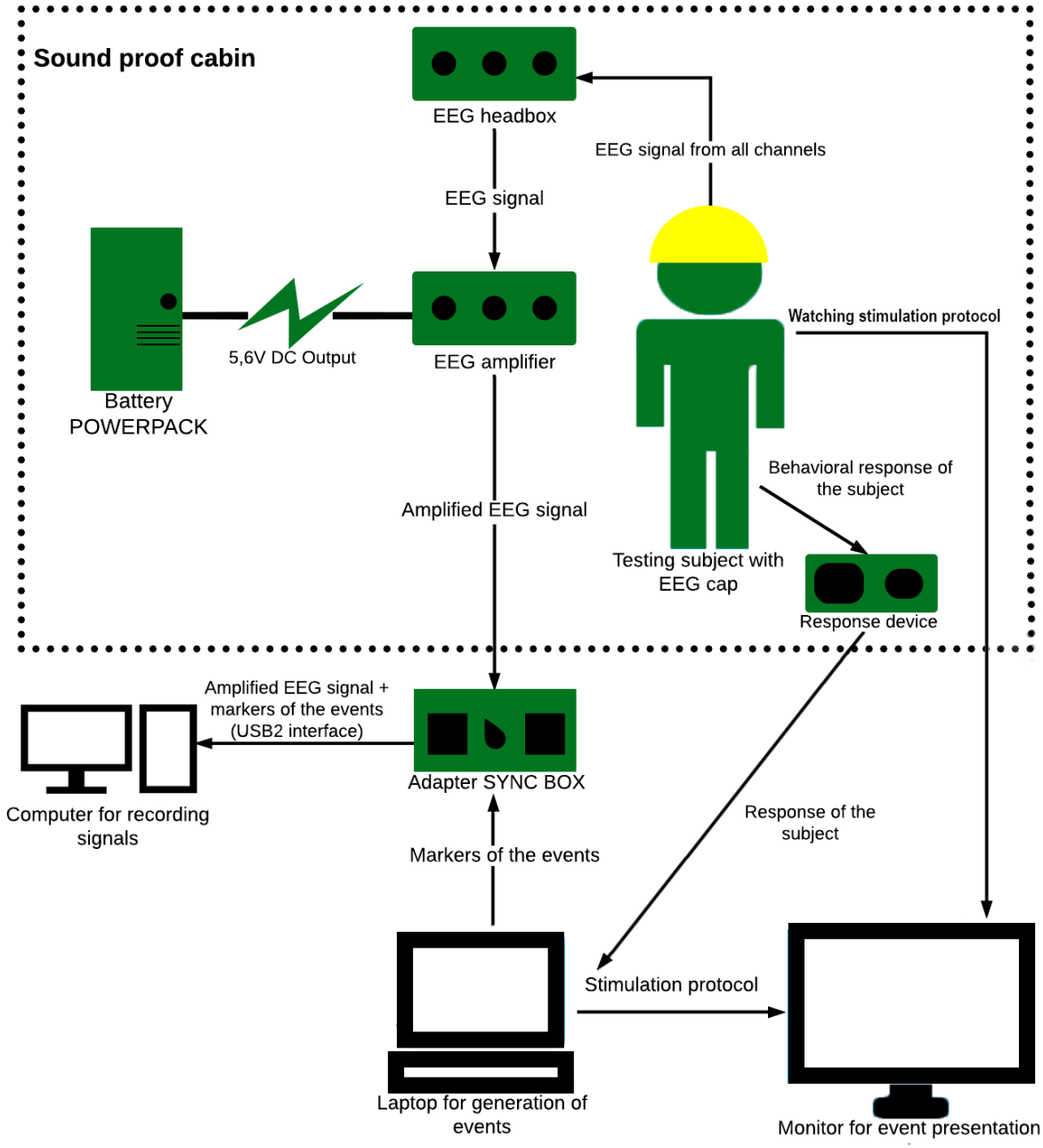
Na základě vybavení se v laboratoři provádějí například experimenty s pozorností řidiče při monotónní jízdě, pozornost dětí s poruchou vývojové koordinace, EEG či ERP experimenty (vizuální stimulace[1]). [11]

Jelikož jsem měl tu možnost být testovacím subjektem, tak vím co obnáší měření v takovéto laboratoři.

Odborníci v laboratoři, pracující v českém národním uzlu pro neuroinformatiku, jsou podporováni mezinárodní organizací INCF (International Neuroinformatics Coordinating Facility). Koordinační či výzkumné aktivity pro neurovědce jsou vedeny z této organizace, díky které se také vyvíjí softwarové standardy a programy.

Na další straně můžete vidět obrázek 2.1 znázorňující infrastrukturu neuroinformatické laboratoře na KIVu.

[1] Jev, při kterém dochází k povzbuzení jedince k určité reakci.



Obrázek 2.1: Infrastruktura laboratoře na KIVu

3 Mozková aktivita

Jak už bylo zmíněno, práce má spojitost s vědním oborem neuroinformatikou díky snímači MindWave Mobile 2. Snímač je schopen detekovat mozkovou aktivitu, která bude zprostředkovávána uživateli zpětnou vazbou v aplikaci pro rozvoj herní inteligence (praktická část), více v kapitole 4.3. Proto dále zmiňuji důležité pojmy neuroinformatiky.

3.1 Mozek

Mozek je řídícím centrem našeho těla a vnímání. Nervová soustava včetně mozku je tvořena nervovou tkání. Tkáň je složena z několika typů buněk, které se dělí na dvě základní: **gilové buňky** a **neurony**. Mozek lze také rozlišit na čtyři základní oblasti:

- **Čelní lalok** (lat. *lobus frontalis*) - je oddělen od temenního laloku centrální rýhou. Podle diplomové práce od Krafta [6] se zde nachází centrum psychické aktivity a emocí. Tato část mozku spolupracuje s ostatními laloky nad uvažováním a řešením strukturovaných úloh.
- **Temenní lalok** (lat. *lobus parietalis*) - po straně hemisféry oddělený hlubokou postranní rýhou (též jámou). Uvnitř postranní jámy se nachází skrytý menší lalok, zvaný ostrov. Zpracovává nervové impulsy smyslového vnímání.
- **Týlní lalok** (lat. *lobus occipitalis*) - analyzuje a zpracovává vizuální podněty. Dle autora Blažka by se dal přirovnat ke zrakovému analyzátoru.
- **Spánkový lalok** (lat. *lobus temporalis*) - zde se nachází sluchový analyzátor, který analyzuje rytmus, barvu tónu a šum zvuku.

Všechny informace jsou převzaty od Blažka V. [2].

3.1.1 Gliové buňky

Gliové buňky jsou přezdívané jako glie či neurologie. Jsou aparátem pro život a funkčnost neuronů. Odhaduje se, že pro jeden neuron připadá deset glií, ale jsou podstatně menší. Jedna z hlavních funkcí těchto buněk je zabezpečení výživy neuronů. Dalšími funkcemi jsou ochrana neuronů před infekčními činiteli a likvidace odumřelých buněk.[2]

3.1.2 Neurony

Neurony jsou tzv. nositelé činnosti nervové soustavy, které se podílejí na přijímání a zpracování informací. Mozek představuje přibližně 40-100 miliard neuronů. Dělí se na tři základní části: **tělo**, **dendrity** (rozvětvené výběžky) a jeden delší výběžek zvaný **axon**. Tělo buňky není bez mikroskopu viditelné, dendrity jsou dlouhé do několika milimetrů, axon může dosahovat délky až 1 metr. Neuron je zakončen koncovými váčky zvanými synaptická zakončení, díky kterým probíhá spojení s tzv. **synapsí**, která obecně spojuje dva neurony. Synapse umožňuje přenos informací z jednoho neuronu na druhý. [2]

3.2 Měření mozkové aktivity

Aktivita všech neuronů se nazývá mozková aktivita. *„Při vytvoření akčního potenciálu (vzruchu) neuronu je indukováno elektromagnetické potenciálové pole, které lze registrovat i na povrchu lebky s elektrickým napětím od 5 do 200* µV.“[6] Pro zaznamenání elektrické aktivity mozku slouží diagnostická metoda Elektroencefalogram (EEG) nebo technika evokovaných potenciálů (ERP). Podle diplomové práce od Krafta existují i další metody, které spadají do kategorie snímkování a implantované technologie.

3.2.1 Elektroencefalografie

Podle Blažka [2] je EEG neboli elektroencefalografie metoda pro zaznamenávání či snímání elektrické aktivity z různých částí mozku, která je způsobená neurony, ale i nežádoucími vlivy (artefakty). Využívá několika elektrod, které jsou elektrovodivou pastou přilepeny k povrchu hlavy. Mozkové vlny mají různou frekvenci a jejich měřením vznikne záznam, kterému se říká elektroencefalogram. Aktivita je vyjádřena buď počtem vln za sekundu (Hz) nebo ji lze vyjádřit jednou z pěti tříd, které jsou popsány níže.

První popisy metody jsou z období 19. století a dnes se jedná o jednu z nejdůležitějších diagnostických metod v neurologii. EEG napomáhá diagnostikovat i léčit epilepsii. Zpravidla se indikuje při podezření na poškození mozku, ale také při onemocněních, zánětech apod.

- **Alfa aktivita** (8 - 13 Hz) - lze ji měřit převážně v zadní části hlavy. Vyskytuje se při relaxaci, odpočinku a meditaci (pokud máme zavřené oči, jsme uvolněni a v klidu).
- **Beta aktivita** (14 - 40 Hz, nejčastěji 15 - 25 Hz) - vyskytuje se ve frontální, centrální a týlní části. Lze ji dělit na tři pásma. Jeli frekvence v prvním pásmu, můžeme říci, že je jedinec zaujatý a pozorný. Ve druhé části pásma zažívá nepříjemný stres. Třetí část je pásmo velmi škodlivé pro náš organismus.
- **Delta aktivita** (< 4 Hz) - tato aktivita se vyskytuje převážně u dětí a u dospělého jen v hlubokém spánku.
- **Theta aktivita** (4 - 7 Hz) - objevuje se při vnitřním soustředění, meditaci a modlitbách. Značí kreativitu, souvisí také s fantazií či se vzpomínkami.
- **Gama aktivita** (> 30 Hz, obvykle 36 - 44 Hz) - je přítomna po celém povrchu hlavy. Odráží stav aktivního zpracování informací. Hluboká koncentrace.

3.2.2 Artefakty

V záznamu EEG se neobjevují jen vlny mozkové aktivity, ale také nežádoucí signály, zvané artefakty, které s mozkovou aktivitou nemají žádnou souvislost. Dělí se dle původu: vyvolané svalovou aktivitou, vyvolané mrkáním, vyvolané srdeční činností (z pocení), vyvolané z pohybu jazyka a artefakty dentální. [2]

3.2.3 Evokované potenciály

Evokované potenciály (EP) jsou elektrické odpovědi nervové soustavy vyvolané vnějšími podněty či událostmi. Elektrická aktivita v mozku jedince je odrazem reakcí nervového systému. EP testují, jak dlouho mozku trvá přijmout a uvědomit si jednotlivé zprávy, které mozek vnímá ať už hmatem, zrakem, sluchem apod. (viz níže). V tomto případě mluvíme o kognitivních evokovaných potenciálech (ERP - Event-Related Potentials).

Slouží pro zhodnocení funkčního stavu neuronálních sítí či nervové dráhy (jednotlivé části nervového systému). Při poruše nervového systému trvá přenos zpráv či reakcí na podněty déle. Rychlost získání výsledků je jedna z výhod vyšetření EP. Jsou registrovány povrchovými elektrodami z kůže nad příslušnými místy hlavy (periferní nerv, míšní segment apod.)

Zde je několik typů evokovaných potenciálů:

- **Zrakové evokované potenciály (VEP)** - odpověď nervového systému na zrakové podněty např. záblesky, obrazy, barvy, změna kontrastu či jasu.
- **Sluchové evokované potenciály (AEP)** - vyvolané různými tóny, zvuky či řečí.
- **Motorické evokované potenciály (MEP)** - jedná se o odpovědi ze svalů. Tyto odpovědi jsou vyvolávány magnetickým nebo elektrickým stimulem.
- **Kognitivní evokované potenciály (ERP - Event-Related Potentials)** - jak už je zmíněno výše, jsou vyvolávány identifikovatelnými podněty a událostmi. **Příkladem je praktická část této bakalářské práce** - aplikace pro rozvoj herní inteligence.

Tvrzení jsou převzata od Bareše [1] a z institutu Masarykovy univerzity [15].

4 Zkoumání pohybových aktivit hráčů

Abych mohl danou aplikaci zprostředkovat, musel jsem se nejprve setkat se zmiňovanými trenéry, od kterých jsem získal větší informovanost. Předali mi znalosti především o mentálním rozvoji v hokejové škole. Účelem setkání bylo také pochopení jednotlivých požadavků na aplikaci, ukázaní pohybů a technik, které by se během trénování s aplikací vykonávaly.

4.1 Mentální rozvoj

Snahou hokejové školy je jak zlepšení dovedností a techniky na ledě, tak také kladení velkého důrazu na zlepšení mentální stránky sportovce.

Jeden z jejich způsobu trénování je propojení technických dovedností s myslí sportovce (v zahraničí se používá pojem "Mental Game"). Z této myšlenky bude prototyp aplikace vycházet. Tyto počítačové metody pro trénink mysli a těla jsou známé jako biofeedback a neurofeedback, které se cíleně využívají i pro měření všech tělesných modalit lidského těla.[5]

Při trénování došlo také k několika měřením z pohledu neurofyziologie, které jen potvrdilo, že tento způsob trénování prospívá. Z článku [5] můžeme zjistit, že opravdu rozvíjí:

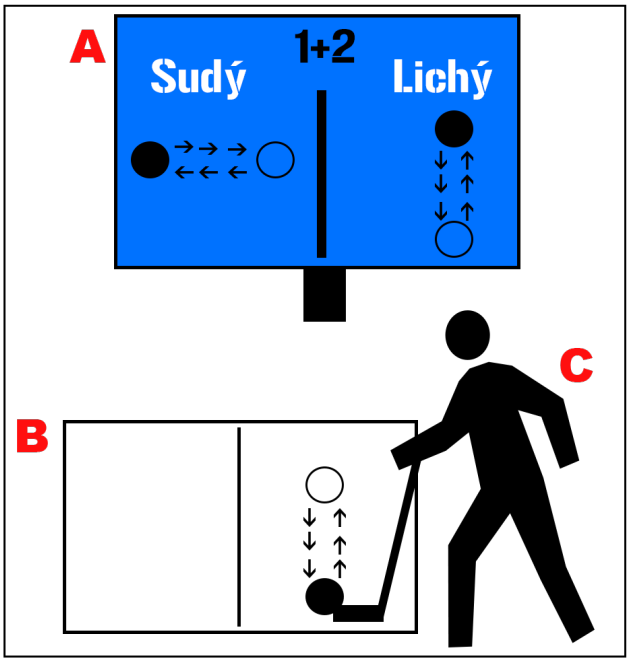
- pozornost a koncentraci,
- rychlost zpracování informací, pohotovost,
- exekutivní funkce včetně emocionální seberegulace[1],
- řeč, schopnost vyjadřování a porozumění,
- prostorovou orientaci.

Podle neuroterapeuta [5] se jedná o **zlepšení kognitivních funkcí, jako jsou zlepšení rychlosti vnímání, koncentrace, logické i verbální myšlení a kreativita**.

[1] Umět regulovat své pocity.

4.2 Aplikace pro rozvoj herní inteligence

Na základě schůzky a předchozí analýzy jsme se s trenéry shodli na několika scénářích, jak by mohl trénink s aplikací probíhat. Na obrázku 4.1 můžete vidět jednoduchý scénář, který jsme analyzovali. Níže pod obrázkem je vysvětlení celého procesu.



Obrázek 4.1: Jednoduchý scénář pro trénink s aplikací

Objekt na obrázku označený červeným písmenem **A** je obrazovka, na které běží aplikace. Podle výsledku matematického příkladu v horní části obrazovky se odvíjí pohyb (směr posunu puku), který hráč **C** bude vykonávat na pomyslně rozdělené ploše **B** před obrazovkou. Matematické příklady se budou generovat náhodně po několika sekundách. V případě obrázku by hráč měl posouvat puk shora dolů a zpět na pravé části pomyslné plochy.

Trenéři tento styl tréninku mají manuálně vyzkoušen a hráčům přináší do tréninkového cyklu nový požitek. Navíc tento způsob trénování **donutí hráče myslet během pohybové činnosti a přesouvá pozornost očí od puku směrem na obrazovku.**

Hráč se naučí zvedat hlavu a rozhlížet se během hry. Tímto je vyřešen důležitý aspekt trénování, a to je **propojení tréninkových dovedností** (pohyb s pukem) **s řešením zápasové situace** (přemýšlení během hry).

4.3 Propojení s IoT zařízením

Ze strany oddělení neuroinformatiky na KIVu byl vznesen nápad, že by se tento projekt mohl posunout o tzv. třídu výše, a to s využitím IoT zařízení, které by daný pohyb mohlo kontrolovat. A tak jsme se zamysleli nad otázkou, kterou by aplikace neřešila a to je: **Jakým způsobem hráč dostává zpětnou vazbu?** Nejprve jsem musel zjistit, co je vlastně potřeba na daném trénujícím hráči sledovat, abych dotyčnému mohl poskytnout zpětnou vazbu v rámci aplikace. V této oblasti nejsem orientován, a proto jsem se zeptal zainteresovaného zmiňovaného trenéra.
Cituji reakci na mou otázku:

> *„Na trénujícím hráči během těchto tréninků není potřeba sledovat nic. Jak má hráč přesně držet hokejovou hůl nejde definovat, každý je jinak vysoký, levák nebo pravák. Hůl drží každý sportovec trochu jinak. Cílem není technika, cílem je hráče naučit myslet, koukat před sebe (nekoukat na puk). Důležité je, aby hráči během simulací nad něčím přemýšleli (namáhali mysl) jinak nemá smysl tyto principy tréninku realizovat. Takové typy tréninků přispějí všem ať už profesionálním, tak mladším hráčům.“*

Souhrnně lze tedy konstatovat, že trenéři na hráčích z hlediska techniky nesledují nic, ale pro neobyčejný požitek z tréninku se naskytla možnost propojit aplikaci s vědním oborem neuroinformatikou. Čelenka MindWave Mobile 2 nám pomůže sledovat mozkovou aktivitu pro poskytnutí zpětné vazby. Tato problematika je zmíněna v kapitole 3.

Dále bude aplikace propojena se snímačem, který by se dal připevnit na puk a umožňoval by posílat informace o pozici či akceleraci puku. Díky těmto informacím bych hráči mohl vizualizovat například rychlost, frekvenci pohybu puku nebo také směr pohybu.

Hardwarovými či IoT zařízeními a jejich vlastnostmi se zabývám v následující kapitole 5.

5 Prostudování snímačů

Jak už bylo zmíněno na konci sekce 4.3, rozhodl jsem se pro využití čelenky od společnosti NeuroSky, díky které lze sledovat mozkovou aktivitu. Na výběr byla také čelenka na suchý zip, MindBand Brainwave EEG, která by na čele během pohybu držela pevněji. Zdála se, že bude vhodnější, ale nejednalo se o jednoduchou konektivitu přes Bluetooth technologii. Čelenka se tedy nedala příliš prakticky využít pro tuto aplikaci.

Pro pozorování pohybu puku mi byl doporučen senzor WT61C, který se dá připojit přes Bluetooth. Jeho základní funkcí je posílání dat o akceleraci nebo úhlu natočení.

5.1 Čelenka MindWave Mobile 2

MindWave Mobile 2 je zařízení vyrobené od společnosti NeuroSky, které umožňuje přes elektrody snímat elektrickou aktivitu mozku. V každém zařízení od společnosti NeuroSky je technologie ThinkGear, která propojuje čelenku s duševním stavem nositele. Zahrnuje několik senzorů a čip, který zpracovává signál z povrchu těla.

Navíc tato čelenka obsahuje patentovaný algoritmus zvaný **eSense**, který zesiluje signál a odstraňuje okolní hluk. Algoritmus má také funkci **eSense Meter**, která ukazuje, jak se uživatel **soustředí** a **medituje**.

5.1.1 Výrobce

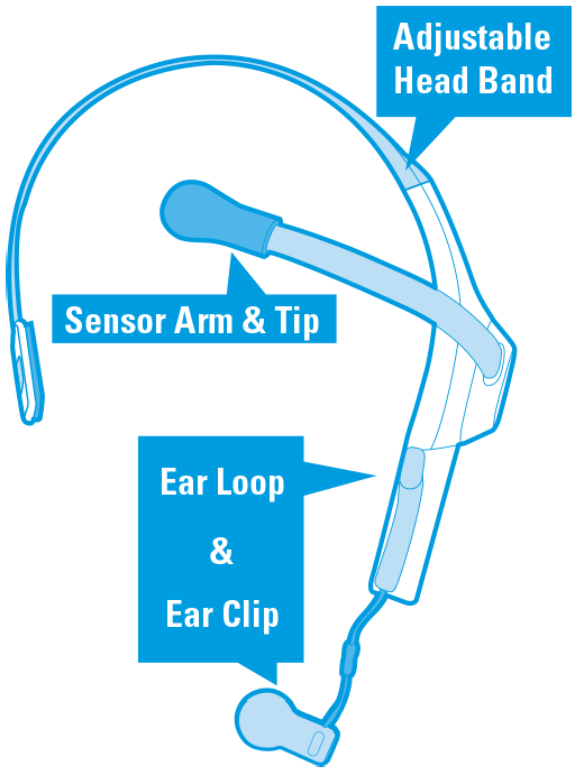
Společnost NeuroSky, Inc., sídlící ve státě Kalifornia v San José, byla založena v roce 2004. Má několik poboček po celé Asii a Evropě. Zabývá se vývojem a inovací biosenzorů, které v masovém měřítku poskytuje na trh. Jejich vedoucím postavením v biosenzorech elektroencefalogramu (EEG) přineslo stovky průlomů v monitorování mozkových vln. Díky pokrokům, které firma udělala v elektrokardiogramových (EKG) biosenzorech a biometrických algoritmech, umožňuje výrobcům a poskytovatelům služeb vyvíjet řešení, která mohou sledovat a analyzovat velké množství kardio-bio-signálů. [8]

Produkty firmy se uplatňují na trhu s širokým využitím v oblastech zábavy, automobilového průmyslu, vzdělání a také zdraví. Jedná se o výrobce

technologie Brain-Computer Interface (BCI), díky které lze měřit mozkovou aktivitu. Společnost také vyvinula BCI zařízení, jako je MindWave či MindSet, díky kterým není potřeba nanášet na hlavu vodivou tekutinu. Jedná se tak o suché snímače, které měří mozkovou aktivitu přímo od pokožky hlavy.

5.1.2 Popis zařízení

Zařízení je ve tvaru plastové čelenky, která se nasazuje na hlavu. Skládá se ze dvou elektrod. Primární elektroda se umisťuje na frontální část hlavy, nad levé oko. Další elektroda se umisťuje na levý ušní lalůček, jedná se o tzv. referenční elektrodu. Výdrž zařízení je okolo 8-10 hodin a je napájeno jednou AAA baterií. MindWave Mobile 2 lze připojit vždy k jednomu zařízení a to s platformou Windows, Mac nebo Android. Spojení probíhá přes Bluetooth technologii. [9] Na následujícím obrázku 5.1 lze vidět zmiňované zařízení.



Obrázek 5.1: MindWave Mobile 2

5.1.3 Specifikace

V následující tabulce 5.1 jsou shrnuty veškeré informace o čelence MindWave Mobile 2.

Vlastnosti MindWave Mobile 2	
Výrobce	NeuroSky, Inc.
Cena	79.99 [$]
Váha	90 [g]
Napájení	1xAAA baterie
Výdrž	8 - 10 [h]
Komunikace	Bluetooth
Maximální výkon	50 [mW]
Rádio frekvenční rozsah	2.420-2.471 [GHz]
Rádio frekvenční přenosová rychlost	250 [Kbit/s]
Dosah vysílače	10 [m]
Modulační rychlost	57 600 [Baud]
Max. rozsah vstupního EEG signálu	1 [mV] pk-pk
Modul pro zpracování signálu	TGAM1

Tabulka 5.1: Tabulka vlastností zařízení MindWave Mobile 2 [9]

5.1.4 ThinkGear modul

Jedná se o modul ThingGear TGAM1 ASIC, čip, který naměřené hodnoty zpracovává do cílových datových toků. Důležitou částí je filtrování, jehož snahou je rozlišit mozkovou aktivitu a artefakty jakéhokoliv původu např. šum. Modul dále obsahuje algoritmy zvané eSence, které byly vyvinuty laboratoří NeuroSky Inc. ve spolupráci s výzkumnými institucemi. [6] [9]

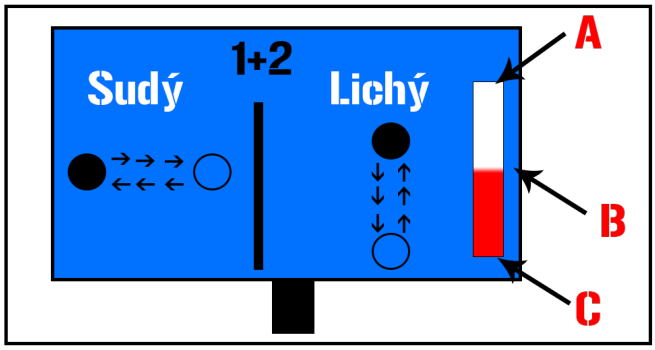
Výstupními daty tohoto modulu jsou [9]:

- detekce mrknutí,
- úroveň meditace a soustředěnosti,
- EEG rytmy,
- hrubá naměřená data,
- úroveň kvality snímané mozkové aktivity.

5.1.5 Možné využití v aplikaci

S tímto zařízením lze především sledovat soustředěnost. Je zde mnoho způsobů, jak hráči vizualizovat zpětnou vazbu:

1. Lze vizualizovat na stupnici, která je znázorněna na obrázku 5.2 níže, od **písmene C do A** (červeně zvýrazněná písmenka). Šipka od písmene **B** ukazuje na aktuální hladinu soustředěnosti. Důležitým aspektem je viditelnost, aby vše na obrazovce bylo čitelné.



Obrázek 5.2: Možná zpětná vazba z MindWave Mobile 2

2. Je zde také možnost informovat hráče o jeho soustředěnosti zvukem. Dle mého názoru by zvuk mohl hráče rozptýlit.

3. Další způsob je barvit pozadí či jiný objekt na ploše, například z tmavé barvy do světlé. Světlá barva by mohla zvýrazňovat maximální soustředěnost.

Otázkou zůstává, zda měření na sportovcích během pohybu bude přesné. Měření nejspíš budou ovlivňovat artefakty vyvolané svalovou aktivitou.

Mezi další vlastnosti, které čelenka umí rozpoznávat, je mrkání anebo meditace. Tyto veličiny lze do aplikace také zapojit najde-li se pro ně využití.

5.2 Senzor WT61C

Šestiosý senzor, který vytvořila společnost WitMotion ShenZhen Co., Ltd., obsahuje 3-osý gyroskop a 3-osý akcelerometr pro měření akcelerace a úhlové rychlosti. Se senzorem lze komunikovat přes Bluetooth.

5.2.1 Výrobce

Společnost WitMotion ShenZhen Co., Ltd, sídlící v čínské provincii Guangdong Province poblíž Honkongu, byla založena v roce 2014. Firma je orientovaná v oblasti senzorických tržních služeb a technických prací. Její senzorové technologie podporují vývoj projektů, které mohou sloužit pro rozvoj či testovací řešení. Také prošla certifikací ISO9000 pro zajištění důvěryhodnosti, bezpečnosti a optimalizace technických řešení.

Společnost vyrábí senzory sklonu, akcelerace, pohybu a mnoho dalších, které se využívají v oblastech: automobilového průmyslu, lodní, letecké a železniční dopravy, strojů, textilního průmyslu, lékařského zařízení a mnoha dalších. [16]

Podobný senzor od tohoto výrobce je využíván ve výzkumu pro zlepšení flexibility sledování pohybu. Senzor obsahuje jen jiný čip MPU-9250, který navíc nabízí 3-osý magnetometr. [18]

5.2.2 Popis zařízení

Jedná se o senzor s modulem akcelerometru a gyroskopu zvaný MPU6050, který zpracuje naměřená data a dále je pošle na výstup přes sériový port či Bluetooth. Modul obsahuje interní baterii, kterou lze nabíjet přiloženým kabelem napětím 3.3V-5V. Data procházejí také Kalmanovým filtrem - algoritmus, který se stará o přesnost dat. Na další stránce můžete vidět obrázek senzoru.

V dokumentaci o zařízení [17] je poznámka, která zmiňuje, že senzor neobsahuje magnetometr a proto úhel natočení je počítán integrací (měření nebude tolik přesné). Natočení úhlu os X a Y lze filtrovat podle gravitačního pole, tudíž tyto osy budou přesnější.

Obrázek 5.3: WT61C senzor

5.2.3 Specifikace

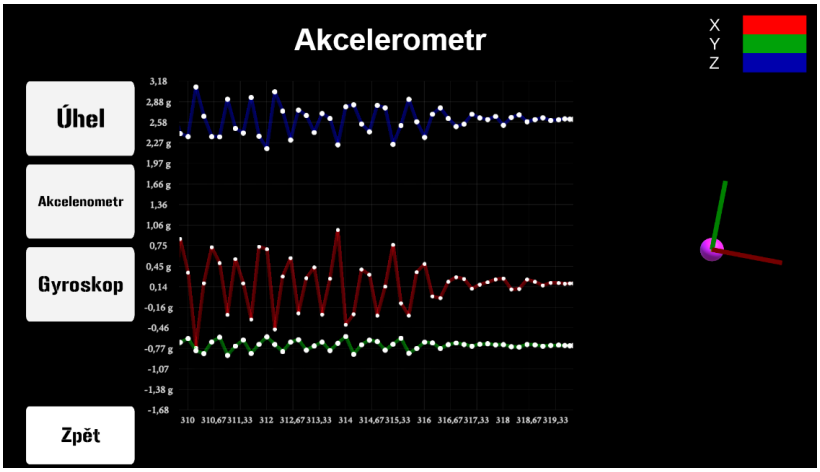
V následující tabulce 5.2 jsou shrnuty veškeré parametry o senzoru WT61C.

Vlastnosti senzoru WT61C	
Výrobce	WitMotion ShenZhen Co., Ltd
Cena	760.66 [Kč]
Váha	13 [g]
Velikost	51.3 x 36 X 15 [mm]
Napájení	3.3 - 5 [V]
Proud	<40 [mA]
Výdrž	2-3 [h]
Dosah	10 [m]
Výstup	Čas, Akcelerace, Gyroskop, Úhel natočení
Rozsah	Acc: $\pm 16g$, Gyr: $\pm 2000°/s$, Úhel: (X:$\pm 180°$, Y: $\pm 90°$)
Frekvence	100 Hz (11520 baud) / 20 Hz (9600 baud)

Tabulka 5.2: Tabulka vlastností senzoru WT61C

5.2.4 Zpracování dat a možné využití v aplikaci

Po konzultaci s vedoucím práce a pro lepší pochopení fungování senzoru, jsem se rozhodl, si jednotlivá data vizualizovat do grafu. K dokumentaci byl dodán vizualizační program, který se mi nepodařil úspěšně propojit se senzorem. V této analytické části jsem si tudíž musel navrhnout a implementovat graf, který mi jednotlivá data vizualizuje, viz obrázek 5.4. Lze si zobrazit jak akceleraci a úhel natočení, tak data z gyroskopu.



Obrázek 5.4: Graf vizualizující jednotlivá data

Záměrem této analýzy je získat co nejvíce užitečných dat pro zpětnou vazbu. První myšlenkou pro splnění cíle bylo vyzkoušení **trackování**[1] **puku**, ke kterému by byl senzor připevněn. To znamenalo vizualizovat kartézskou soustavu souřadnic a do ní hodnotu akcelerace. Nicméně se zdá, že tato analýza přesáhla možnosti senzoru.
Postupoval jsem následovně:

1. Vizualizoval jsem kartézskou soustavu souřadnic, abych věděl, vůči jakému natočenému souřadnicovému systému se puk pohybuje.

    - Osa Z: v této ose se mi podařilo porozumět hodnotám senzoru (od -180° do 180°) a tak jsem si naimplementoval pohyb otočení puku v ose Z.
    - Osa X a Y: v těchto dvou osách se mi **nepodařilo porozumět vztahu hodnot** k rotaci senzoru, protože se chovají neočekávaně.

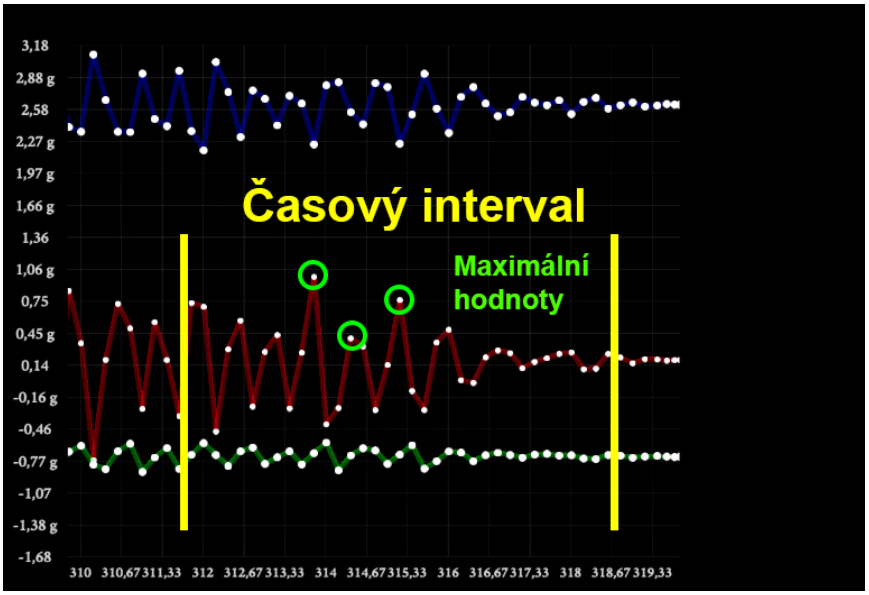
2. Vizualizace hodnoty akcelerace. Bez správného vykreslení kartézské soustavy souřadnic (1) je tato vlastnost zbytečná.

[1] Monitorování 3D pozice objektu v reálném čase.

Další vlastnost, kterou bych hráči mohl vizualizovat, je **frekvence pohybu, neboli počet změn pohybů s pukem.** Tato problematika se mi zdála triviálnější, pokud bych frekvenci kmitání s pukem vizualizoval jako tu nejvyšší hodnotu akcelerace ze všech tří os.

Opak je pravdou, protože se puk může pohybovat konstantní rychlostí a to znamená nulovou akceleraci. Dále by se také mohl vyskytnout problém s pohybem puku na malé vzdálenosti, kdy hráč nepřidává tolik sil do pohybu (malá akcelerace), ale i přes to, by jeho frekvence pohybu byla vysoká.

Řešením je sledovat počet maximálních hodnot akcelerace v grafu během stanoveného časového intervalu ve všech složkách souřadnic, viz obrázek 5.5. Problémem je, že těchto hodnot je v grafu nespočet. Jaké hodnoty tedy přesně znázorňují změnu pohybu? Nechal jsem si poradit a bylo mi doporučeno si **prostudovat Fourierovu transformaci** pro získání maximálních hodnot akcelerace za určitý časový úsek. Většinou se jedná o vstup několika naměřených hodnot (akcelerace) za sekundový časový interval. Výstupem po transformaci jsou frekvenční hodnoty.



Obrázek 5.5: Obrázek ukazující maximální hodnoty akcelerace na ose X

Frekvence může být znázorněna v aplikaci obdobně jako soustředěnost u obrázku 5.2 s čelenkou MindWave Mobile 2.

6 Návrh rozhraní mozek-počítač

Na základě kapitoly 4.2, která pojednává také i o možném návrhu, jsem se dohodl s Ing. Petrem Brůhou na realizaci BCI a ACI aplikace, která nese název Hockey Coach. Tato kapitola popisuje vize projektu a návrh aplikace. Také je zde zmíněn samotný návrh datové struktury v databázi a správa uživatelských účtů.

6.1 Vize aplikace

V této podkapitole shrnu cíle a rizika projektu. Také upozorním na zpětnou vazbu, kterou bude tato aplikace přinášet společně se statistikami.

6.1.1 Cíle aplikace

Aplikace je určena především sportovcům, kteří jsou zaměřeni na hokejové sporty (florbal, hokejbal, hokej). Zároveň je určena těm lidem, kteří se chtějí sami zdokonalovat i v domácím prostředí.

Od ostatních aplikací se liší tím, že cílí především na zlepšení herní inteligence. Uživatele donutí přemýšlet nad definovaným problémem během vykonávání herních pohybů s pukem. S tímto nápadem přišli zmínění hokejoví manažeři, kterým se tento způsob osvědčil. **Aplikace je proto navrhovaná tak, aby splňovala většinu požadavků od těchto potenciálních zákazníků.**

Požadavkem je vymyslet, jakým způsobem přineseme hráči zpětnou vazbu. Vyřešili jsme to tím, že jsme zakomponovali do aplikace zmíněné snímače v kapitole 5, které posílají zpětně data. Data jsou zpracovávána a zobrazována uživatelům. Výstup bude ve formátu, který už byl zmíněn v kapitolách 5.1.5 a 5.2.4 o možném využití senzorů. Pro vizualizaci zpětné vazby bude v aplikaci také sekce pro statistiky, která bude zobrazovat grafy, ale i srovnání s ostatními hráči v tabulkách, více v kapitole 6.1.4 o zpětné vazbě a statistikách.

Níže jsou definovány další požadavky od hokejových trenérů, které budou zakomponovány:

1. Propojení aplikace se správou účtů. Každý hráč bude mít vlastní přihlašovací údaje do aplikace, čili účet, pod kterým budou data ukládána.

2. Přepínání mezi českým a anglickým jazykem.

6.1.2 Rizika projektu

Během analýzy a návrhu jsem dospěl k názoru, že se kolem této práce vyskytuje hned několik potenciálních rizik. Mohou se týkat jak implementační části, tak reálného provozu u uživatele. Níže jsou popsané:

- **Zabezpečení aplikace a jejich dat** - v kapitole 6.3.2 se snažím popsat návrh datové struktury, ale také eliminovat osobní data, která jsou potřeba pro tento systém. Přesto je nutné, aby správce či provozovatel databáze zohledňoval nařízení GDPR. Dále by neměl uživatel nikomu sdělovat své přihlašovací údaje, neboť by mohl vystavit svůj účet riziku změn dat.

- **Zpětná vazba ze senzorů = prototyp** - rád bych upozornil na to, že IoT zařízení, která jsou využívána pro získávání dat pro zpětnou vazbu, jsou jen tzv. dodatkovým modulem. Jedná se o prototyp, který přinese nový pohled na problematiku, ale ne vždy může přijímat data, která minimálně očekáváme. Také budou na straně aplikace dopočítávána další a hodnotnější data pro uživatele, jako je počet pohybů za sekundu (frekvence) - tudíž data nemusí být přesná. Dalším rizikem jsou senzory, které je nutné připojit přes Bluetooth a tak může kdykoli dojít k přerušení.

- **"Fake"data** - kdo bude chtít, vždy si najde nějakou cestičku, jak oklamat systém. Vždy je možnost si pustit aplikaci a trénovat s tím, že si dotyčný vezme puk se senzorem do ruky a vynaloží větší frekvenci, než je možné s hokejovou holí a následně data uloží. Tudíž se tato informace může projevit na statistikách. Prozatím budeme sázet na zdravý selský rozum uživatele a na jeho uvědomění si, že tímto cesta ke zlepšení nevede.

- **Čas pro testování** - je nezbytné, aby aplikace či výsledný produkt byl otestován s několika potenciálními zákazníky (sportovci). Je potřeba mít dostatek času na ladění a vyzkoušení, než bude aplikace fungovat v provozu.

6.1.3 Konzultace s potenciálním zákazníkem

Nejprve bych měl zdůraznit, že zmiňovaní trenéři tento navrhovaný produkt budou používat a proto jsou tzv. potenciálním zákazníkem. Během návrhu aplikace došlo ke společné schůzce. Cílem této schůzky bylo shodnout se na požadavcích. Většina požadavků a samozřejmě celá myšlenka této aplikace přišla ze strany trenérů, protože na takovém způsobu trénování mají postavenou hokejovou školu. Z naší strany je požadavek na propojení aplikace s IoT zařízeními a vizualizování zpětné vazby.

Na trhu jsou také **konkurenční aplikace**, které rovněž trenéři využívají, a proto mi mohli sdělit zpětnou vazbu, čím by se tato aplikace na trhu mohla lišit, cituji:

„Rozdíl je v rozvoji inteligence. Ve většině aplikací na trhu rozvíjíš jen reakci na předem definované podněty (symboly, animace apod.). V této aplikaci také, ale je nutné ještě před tím vyřešit některý z problémů (matematický příklad), od kterého se podněty pro daný trénink odvíjí. Tím se snažíme automatizovat dovednosti a učit se přemýšlet o stavu hry (simulovat problematiku v zápase). Ve většině aplikací na trhu nic takového není. Samozřejmě bonusem je také zpětná vazba.“

6.1.4 Zpětná vazba a statistiky

Cílem je také zobrazovat v aplikaci statistiky z dat, která nám poskytují zmiňovaná zařízení. Na základě schůzky s trenéry jsme se dohodli, jaká data by byla vhodná sbírat pro přínos a zpětnou vazbu uživateli.

Ze snímače WT61C získáme hodnotu akcelerace a na základě požadavků bude po každém cvičení zobrazen počet pohybů za sekundu (frekvence) a také akcelerace každé osy v grafu. Pro statistiky bude ukládán průměrný počet pohybů (frekvencí) a doba trvání tréninku. Uživateli se v grafech zobrazí data za zvolený časový interval, např. zobrazí aktivitu za měsíc prosinec v roce 2019. V samotném tréninku bude zobrazena jen frekvence pohybů obdobně jako v kapitole 5.1.5.

Ze snímače MindWave Mobile 2 bude v tréninku zpracovávána soustředěnost obdobně jako frekvence u senzoru WT61C.

Co konkrétně je **potřeba ukládat** a také vše, co se týká databáze, je probráno **v kapitole 6.3 o správě účtů a databáze.**

6.2 Popis rozhraní

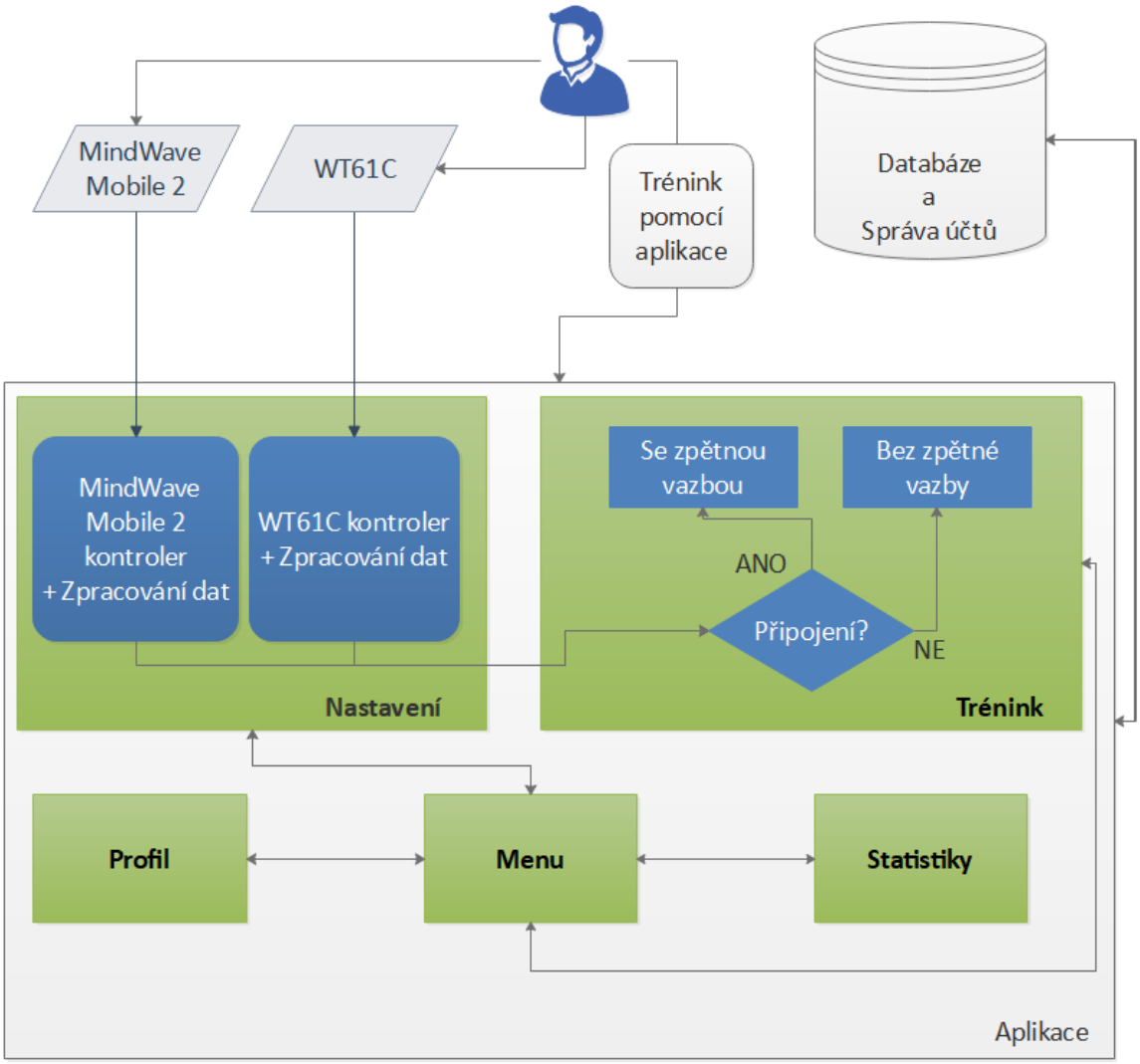
Navržené rozhraní, neboli aplikace, bude sloužit pro uživatele, který má přihlašovací údaje do daného systému. Program je navrhován především na Windows platformu.

V rámci semestrální práce z předmětu Mobilní komunikace a zařízení (MBKZ) bude implementována část aplikace, která bude sloužit pro celý tým hráčů či skupinu pod jedním účtem. Dále také bude sestavena aplikace na Android zařízení.

Hráč se po přihlášení dostane do sekce **Menu**, ze kterého bude možnost vejít do dalších sekcí aplikace:

- **V sekci Nastavení** si uživatel bude moct nastavit jazyk a především ladit připojení s IoT senzory (nastavení portů).
- **V sekci Profil** si hráč bude moci navolit svá data, pod kterými se budou statistiky ukládat - pohlaví, věková kategorie, přezdívka.
- **V sekci Trénink** bude možnost si vybrat hned z 10 cvičení, kde v každém cvičení bude jiná simulace pukem, kterou hráč bude provádět, a také jiná obtížnost matematického problému (problém, který hráč musí vyřešit a od kterého se odvíjí prováděný pohyb s pukem). 10. cvičení dle požadavků bude odlišné od ostatních. Bude se lišit tím, že se zde bude dbát na rychlost dribllingu, neboli pohybu z levé do pravé strany (frekvence). Počet těchto pohybů bude následně vyhodnocen z dat, které naměřil senzor WT61C pro měření akcelerace.
- **V sekci Statistik** se objeví grafy a tabulky, ve kterých hráč uvidí naměřená data ze všech svých cvičení. V této sekci hráč také uvidí nejen svá data, ale i data ostatních hráčů z 10. cvičení. Uživatel si díky této statistice může porovnat svojí rychlost frekvence pohybu puku s ostatními hráči.

Rozhraní navrhuji tak, aby se obešlo i bez zmiňovaných senzorů viz kapitola 6.2.1. Testování aplikace bude probíhat převážně po stránce uživatelských akceptačních testů, více v kapitole 8 o testování. Na obrázku 6.1 můžete vidět návrh celého rozhraní, kde je popsáno, jakým způsobem bude aplikace fungovat. Uživatel (ikona hlavy na obrázku níže) trénuje a zároveň může používat senzory MindWave Mobile 2 a WT61C, které komunikují s kontrolery (ovladači), kde se data zpracovávají. Aplikace bude mít 5 hlavních sekcí (zelené čtverečky) a bude komunikovat s databází.



Obrázek 6.1: Návrh systému - aplikace

6.2.1 Modularizace

Cílem je navrhnout aplikaci tak, aby nebyla závislá na zmiňovaných snímačích WT61C a MindWave Mobile 2. Pokud hráč připojí snímač, je obohacen o zpětnou vazbu a o další statistická data. Jedná se o prototyp aplikace se senzory, které nemusí být spolehlivé, a proto nechceme, aby aplikace byla na snímačích závislá. Z pohledu trenérů má i samotná aplikace využití bez senzorů. **Moduly můžete vidět na uvedeném obrázku 6.1 v předchozí kapitole. Jedná se o modul MindWave Mobil 2 kontroler a WTC61C kontroler.**

6.3 Správa uživatelů a databáze

Aplikace bude propojena se správou uživatelských účtů, aby měli hráči svá data uložena. Databáze bude uchovávat naměřená data a informace o uživateli. Pro tento projekt jsem si vybral správu uživatelů a databázi od služby Firebase.

6.3.1 Firebase

Služba Firebase je od firmy Google. Své produkty umožňuje propojit hned s několika technologiemi přes svá API (Application Programming Interface) rozhraní pro platformy iOS, WEB, a také Unity, ve kterém jsem si zvolil dělat tento projekt. Dále své služby rozděluje do tří kategorií - Buid, Improve a Grow.

Část **Build** zahrnuje hned několik důležitých služeb. Jedna z nejdůležitějších služeb je **Authentication** neboli správa uživatelů. Nabízí různé způsoby ověření, např. emailem a heslem, ale také ověření přes třetí stranu jako je Google, Facebook apod. Další službu (ve vývojové části), kterou v tomto projektu použiji, je **Realtime Database** neboli databáze synchronizovaná v reálném čase s každým připojeným klientem. Data jsou ukládána v hierarchické struktuře JSON (JavaScript Object Notation) a proto je nutné se zamyslet nad návrhem struktury, viz kapitola 6.3.2. Firebase pro vývoj nabízí také hosting pro webové aplikace, datové uložiště nebo cloudové funkce, které umožní automaticky spustit kód na backendu v reakci na události vyvolané funkcemi Firebase.

Kategorie **Improve** nabízí službu **Crashlytics** k reportování chyb, sledování stability a kvality aplikace v reálném čase. Dle mého názoru by tato

služba měla být integrována do aplikace hned od začátku používání. Dále tato kategorie obsahuje služby na monitorování výkonu a testování aplikace.

Část **Grow** obsahuje službu **Analytics**, kterou nazývají srdcem Firebase. Tato služba poskytuje přehled a informace o používání aplikace, což umožňuje činit rozhodnutí kolem marketingu a optimalizace. Dále se v této sekci vyskytují i další služby, např. předpovídání chování uživatele, posílání zpráv a notifikací k upozorňování klienta.

Důležitým faktorem pro používání těchto služeb jsou jejich **ceny**. Cenové kategorie jsou zde dvě. První kategorie **Blaze plan** nabízí všechny služby. Je zde však možnost nastavit si limit služeb, které potřebujete pro svůj projekt. Podle toho se poté odvíjí cena. Lze nastavit např. počet transakcí za den či velikost databáze, u které je cena $5 za Gigabyte. Druhá kategorie **Spark plan** je zdarma, slouží k odladění všech služeb s nižšími limity. Vlastník této cenové kategorie získá např. databázi s 1GB velikostí a počet stažení maximálně 10GB za měsíc. Pro tento projekt jsem si zvolil kategorii Spark plan, protože parametry, které nabízí, jsou pro projekt dostačující.

Před používáním jakékoli služby je nutné danou službu propojit s projektem. K tomu slouží SDK (Software development kit), balíček, který se musí naimportovat do projektu. Balíček poskytuje rozhraní, díky kterému lze komunikovat s danou službou. Všechny informace ohledně Firebase a jejich služeb jsem čerpal ze zdroje [3].

6.3.2 Návrh datové struktury

Dalo by se říci, že návrh databázové struktury procházel změnami i během vývoje. V rámci semestrální práce z předmětu MBKZ bylo nutné přidat nové atributy a také přehodnotit některé části návrhu databáze.

Na rozdíl od SQL databáze, v této databázi neexistují žádné tabulky ani záznamy. Jedná se o stromovou strukturu ve formátu JSON. Databázi je možné vnořit do hloubky až 32 úrovní a uživatele to může vést k myšlence, že má možnost data ukládat jak uzná za vhodné. Nicméně pokud si načteme data z databáze, načteme také všechny potomky tohoto uzlu. Kromě toho, když někomu udělíte přístup pro čtení nebo zápis uzlu v databázi, tak mu také udělíte přístup ke všem datům v tomto uzlu. Je potřeba návrh databáze pečlivě zvážit a rozmyslet si, jaká data budou ukládána v rámci této práce. Po několika hodinách práce a několika pokusech o návrh jsem dospěl k této

struktuře. **Hlavními uzly** jsou: type_account, users, groups.

Uzel **type_accounts** drží informace o typu uživatelského účtu. Potomkem uzlu je **UserID (identifikace uživatele - vygeneruje Firebase)**. Potomkem uzlu UserID je TYPE a hodnotou je řetězec USER nebo GROUP.

Uzel **users** drží veškerá data o účtech, která mají typ USER. Potomky tohoto uzlu jsou **personal_data** a **stats**:

- **personal_data** - potomkem je identifikátor účtu UserID a jeho potomky jsou už jednotlivé informace o účtu jako je věková kategorie, email, přezdívka, pohlaví, součet odehraného času a především identifikační číslo pro statistiky (tuto sekci představuje uzel stats).
- **stats** - uzel reprezentující statistiky jednotlivých uživatelů. Potomci jsou **progress** a **ten_exercises**:
  - **progress** - zachovává časové informace a průměrnou frekvenci v každém tréninku, který byl uživatelem spuštěn. Účelem uložení těchto dat je zobrazit uživateli pokrok o svém výkonu.
  - **ten_exercises** - uzel je určen pro porovnání výkonů z 10. cvičení napříč mezi všemi uživateli. Důvodem je to, že 10. cvičení je zaměřené na rychlost. 10. cvičení se také dělí na další 4 typy - podle času, jak dlouho hráč trénuje - 30s, 60s, 90s a 120s. Uchovávají se jen ty data, která jsou nejlepší (data se přepisují) tzv. nejvyšší průměrná frekvence (AverageFreqMax) a také nejvyšší frekvenční součet (SumFreqMax) - počet pohybů pukem z levé do pravé strany.

  Každý z těchto dvou potomků má jako následující potomek StatsID, který reprezentuje každého uživatele - jedná se o klíč napříč všemi statistikami, více v kapitole 6.3.3 o zabezpečení.

Uzel **groups** byl navržen podobně jako users. Proto zmíním pouze uzel, který v users není, **PlayerID**, který reprezentuje hráče. Na jednom účtu je možné mít až 20 hráčů. Před každým cvičením je nutné, aby si hráč zvolil přidělené PlayerID, pod kterým se jeho data budou ukládat.

Příklad navržení a uchování dat můžete vidět v příloze na obrázcích 10.1 a 10.2.

6.3.3 Zabezpečení

Výhodou konfigurace zabezpečení je její oddělení od implementace a logiky celého produktu. Klienti nenesou odpovědnost za bezpečnost a nemusí tak chránit data před světem. Konfiguraci databáze můžete vidět v příloze na obrázku 10.3.

Pravidla (zabezpečení databáze) se skládají z výrazů podobných JavaScriptu obsažených v JSON struktuře, viz zmíněný obrázek. Struktura dat musí odpovídat strukturám v naší databázi. Databáze má 4 typy pravidel:

- **.read** - definuje, kdo a za jakých pravidel může data číst.
- **.write** - definuje, kdo a za jakých pravidel může do databáze zapisovat.
- **.validate** - ověřuje konzistenci dat. Zda jsou např. data v očekávaném formátu.
- **.indexOn** - databáze umožňuje řadit a vytvářet okamžité dotazy indexováním dat.

Po přečtení a pochopení dokumentace jsem dospěl k návrhu pravidel, který **chrání uživatelská osobní data (uzel: personal_data na obrázku)**, ať už účtu typu USER nebo GROUP. Tyto osobní data si přečte jen uživatel, kterému data patří.

Co se týče **statistik (uzel: stats na obrázku)**, zapisovat je může jen ten uživatel, kterému se shoduje název uzlu s hodnotou StatsID (identifikační číslo uživatele pro statistiky). Nebylo vhodné použít UserID namísto StatsID, protože by se jednalo o narušení bezpečnosti a ostatní uživatelé by si mohli přečíst UserID jiného uživatele. Pro získávání statistických dat je nutné, aby si uzel stats s jeho potomky mohl přečíst kdokoli.

Díky **službě Authentication**, kterou nabízí Firebase, je produkt zabezpečen před nelegálním používáním. Pokud k účtu přistupuje více uživatelů, tak to není v tuto chvíli možné ověřit.

6.4 Rozhraní se snímači

V této sekci zmíním, co je nutné udělat a připravit z pohledu uživatele proto, aby senzory mohly poskytovat potřebná data. Dále zde bude řečeno, co je nutné doimplementovat v navrhované aplikaci pro připojení senzorů.

6.4.1 MindWave Mobile 2

Z pohledu uživatele je nezbytné, aby měl na svém počítači nainstalovaný software ThinkGear Connector, který lze sehnat na webových stránkách výrobce tohoto zařízení. Nadále už jen stačí zařízení připojit přes Bluetooth.

Pro využití čelenky v navrhované aplikaci je nutné si naimportovat balíček, který umožní pracovat s ThinkGear Connector aplikací. Tato aplikace už napřímo komunikuje se snímačem.

6.4.2 Senzor WT61C

Senzor pro měření akcelerace a úhlu natočení komunikuje přes Bluetooth. Před použitím je potřeba senzor zabudovat do některého z typů puků pro možné použití. Proto bylo nezbytné vyhledat puk, který má menší třecí plochu, aby bylo možné jeho použití i v domácnosti, kam daná aplikace především míří. Propracovaný puk se senzorem můžete vidět na obrázku 6.2 níže.

Snímač nemá žádný balíček, přes který by se dalo se senzorem komunikovat. Proto je nutné si napsat vlastní komunikaci. Implementace probíhala už v analytické části do prototypu (pro zobrazení a analyzování dat pro zpětnou vazbu). Senzor se připojuje přes Bluetooth a proto je nutné, aby uživatel zvolil správný port připojení v navrhované aplikaci.

Obrázek 6.2: Návrh puku se senzorem

6.5 Software

Už od samotného počátku jsem se přikláněl k tvorbě tohoto projektu v **Unity enginu** (v Unity3D editoru) s podporou scriptů v jazyce C#.

Jeden z důvodů výběru Unity enginu je tvorba aplikace pro více platforem. Je snadno pochopitelný i pro začátečníky. Další výhodou je, že má tento engine podporu od Firebase služeb, tudíž je možnost integrovat služby do produktu implementovaného v Unity. Více v kapitole 6.5.1 o Unity enginu.

Dále je potřeba balíček pro vizualizaci grafů, kdy takovéto komponenty v Unity nejsou jednoduché pro implementaci a proto jsem se rozhodl využít dvě knihovny **Dynamic Line Chart** [14] a **Awesome Charts and Graphs** [12], které umí vykreslit grafy a zobrazit potřebná data.

Také bylo potřeba najít knihovnu v jazyce C# pro Fourier Transformaci (FT), díky které spočítám frekvenci neboli počet pohybů za sekundu z dat akcelerometru WT61C. Pro tuto úlohu byla nalezena knihovna **DSPLib** ze zdroje [4]. Více o použitých knihovnách v kapitole 7.5.

6.5.1 Unity engine

Unity engine je všestranný herní engine pro vývoj 2D nebo 3D her (aplikací) pro mnoho platforem. Jedná se o volný, ale i placený produkt, který vlastní i svůj editor Unity3D (vývojové prostředí) pro návrh vlastní aplikace.

Je vytvořený společností Unity Technologies a první vydání vyšlo v roce 2005. Cílem enginu je přinést profesionální nástroje pro amatérské, ale i pro profesionální vývojáře. Jak už bylo zmíněno, Unity umožňuje tvorbu aplikací pro spoustu platforem, jako jsou konzole, zařízení s iOS, Androidem či Windows systémem a také umožňuje tvorbu her pro virtuální realitu. Unity3D editor disponuje také o obchod zvaný Asset store, který nabízí balíčky a knihovny. Balíčky jsou vytvořeny členy komunity, ale i společností Unity Technologies.

Práce s Unity3D se dělí na dvě části. První část je návrh scény z komponent (Script) a objektů (GameObject). Scéna je část softwaru, která představuje v našem případě každou obrazovku (MainMenu, Settings apod.). Druhá část je programování komponent, scriptů, které se starají o veškerou logiku a interakci s objekty ve scéně.

7 Implementace

7.1 Popis programu

Na základě návrhu je vytvořeno rozhraní neboli program vytvořený v enginu Unity v programovacím jazyce C#.

Na další stránce můžete vidět zjednodušený diagram interakcí **7.1, který je zde popsán.** Aplikace obsahuje několik obrazovek - scén, které jsou navrženy v Unity3D editoru. Scénu lze chápat jako modul tzv. View v **MVC** (Model-view-controller) architektuře. Všechny hlavní scripty scén aplikace můžete vidět na zmiňovaném obrázku 7.1. Každá ze scén má svůj kontroler, který je **vyznačen modře**, jsou pojmenovány s prefixem UIManager. Všechny tyto scripty používají pomocné knihovny či jiné scripty pro svojí potřebu.

Hlavním jádrem tohoto projektu jsou dva scripty `MPU6050Controller.cs` a `NeuroSkyController.cs` **označené zeleně**, které komunikují se senzory. S těmito kontrolery dále komunikují UIManager scripty v závislosti na interakci uživatele s aplikací.

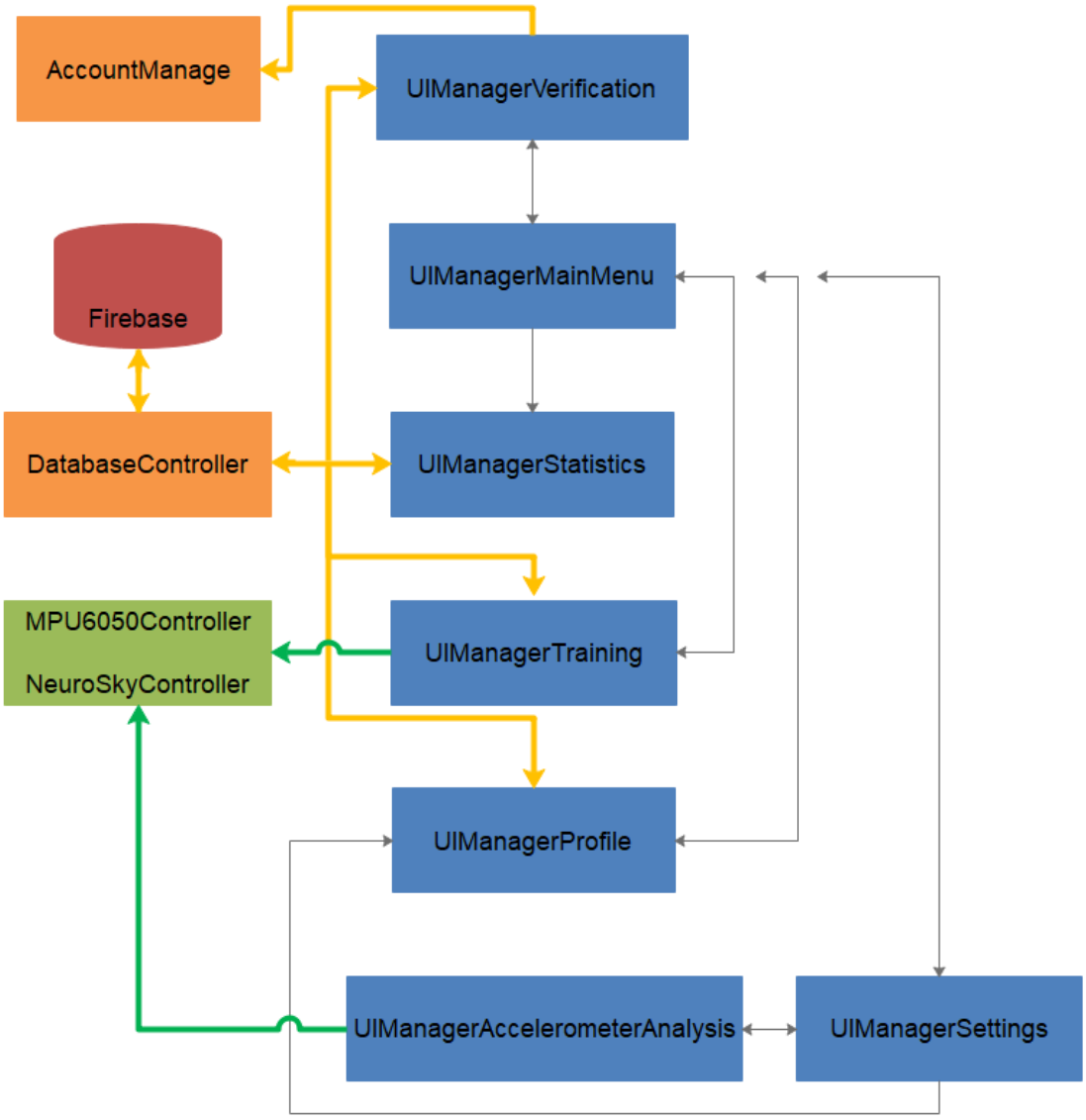
Scény:

- nastavení (`UIManagerStatistics.cs`),
- profilu (`UIManagerProfil.cs`),
- tréninku (`UIManagerTraining.cs`),
- přihlášení (`UIManagerVerification.cs`)

komunikují s databází Firebase přes třídu `DatabaseController.cs` (na obrázku **označena oranžově**) pro potřebu uložení nebo načtení dat.

Jako poslední z důležitých tříd je `AccountManage.cs` (**označená oranžově**), která vlastní odkazy na důležitá data uživatele, např. typ účtu, email, přezdívku apod.

Tento projekt obsahuje mnoho scriptů a tříd. V následujících kapitolách popíšu strukturu projektu a také důležité třídy a scripty prostřednictvím UML diagramů.



Obrázek 7.1: Zjednodušený diagram aplikace

7.2 Struktura unity projektu

Struktura Unity projektu patří mezi ty největší struktury programových nástrojů. Z tohoto důvodu popíšu jen vlastní kořenovou strukturu, ve které jsou vlastní komponenty, scripty, modely, obrázky apod. Kořenovou složkou je složka s názvem **`_HockeyCoach`** a v ní složky se soubory:

- **`_Scenes`** - složka obsahující všechny scény (obrazovky) aplikace:
  - **`VerificationScreen`** - přihlašovací scéna do aplikace.
  - **`MainMenuScreen`** - hlavní menu aplikace.
  - **`MPU6050ControlScreen`** - scéna pro připojení senzoru WT61C.
  - **`AnalysisMPU6050DataScreen`** - scéna určená především pro analýzu dat z akcelerometru WT61C, ale také pro ladění.
  - **`NeuroSkyControlScreen`** - scéna pro ladění připojení čelenky MindWave Mobile 2.
  - **`SettingsScreen`** - scéna nastavení.
  - **`ProfileScreen`** - scéna pro zobrazení uživatelských dat.
  - **`StatisticsScreen`** - scéna pro zobrazení statistik.
  - **`TrainingScreen`** - scéna, která poskytuje několik cvičení pro trénink.
- **`Tests`** - složka s integračními testy.
- **`Font`** - složka obsahující typy písma, které jsou v aplikaci využívány.
- **`Materials`** - složka obsahující materiály (barvy). Většina pro testování s 3D objekty.
- **`Prefabs`** - složka pro vytvořené a upravené komponenty (i složené z více komponent), které se mohou využívat na více místech v projektu.
- **`Sprites`** - složka pro obrázky, které jsou používány v projektu.
- **`Scripts`** - složka pro vlastní scripty. Jsou rozdělené do složek dle scén, ve kterých se používají. Pokud se používají ve více scénách, jsou rozdělené dle použitelnosti.

Scripty či třídy jsou hlavní částí tohoto projektu a proto jsou níže popsány. **Projekt obsahuje 52 scriptů a tříd.** Na úvod popisu uvedu, že **řídící scripty reagují na vstup od uživatele** a starají se o GUI (Graphic User Interfac) dané scény, která je aktivní.

- **Firebase** - složka obsahující třídy zaměřené na komunikaci s databází:
  - **RealtimeDatabaseClasses** - složka se třídami sloužící pro uchovávání dat pro uložení nebo načtení z databáze. Všechny tyto třídy nesou v názvu prefix RD (jako Realtime Database).
  - **AccountManage.cs** - třída chovající se jako správce daného přihlášeného účtu. Obsahuje instance na data uživatele, ale i např. metodu pro odhlášení z aplikace.
  - **DatabaseController.cs** - třída komunikující s databází.
- **Verification** - složka obsahující scripty scény VerificationScreen pro přihlášení do aplikace:
  - **UIManagerVerification.cs** - řídící script dané scény.
- **MainMenu** - složka se scripty používající v hlavním menu aplikace.
  - **UIManagerMainMenu.cs** - řídící script této scény.
- **Profile** - složka se scripty používající v ProfileScreen scéně:
  - **UISwitchManagerProfilePanels.cs** - script, který aktivuje příslušný řídící script dle typu účtu (GROUP nebo USER).
  - **UIManagerProfileForUser.cs** - řídící script pro typ účtu USER.
  - **UIManagerProfileForGroup.cs** - řídící script pro typ účtu GROUP.
  - **RowForGeneratePlayers.cs** - pomocný script pro generování vlastních řádků v tabulce.
- **Settings** - složka se scripty používající ve scénách AnalysisMPU6050DataScreen, MPU6050ControlScreen, NeuroSkyControlScreen a SettingsScreen:
  - **UIManagerAccelerometerAnalysis.cs** - řídící script scény AnalysisMPU6050DataScreen.
  - **UIManagerMPU6050Control.cs** - řídící script scény MPU6050ControlScreen.
  - **UIManagerNeuroSkyControl.cs** - řídící script scény NeuroSkyControlScreen.
  - **UIManagerSettings.cs** - řídící script scény SettingsScreen.
- **Statistics** - složka se scripty používající ve StatisticsScreen scéně:

  - **UISwitchManagerStatistiscPanel.cs** - script, který aktivuje příslušný řídící script dle typu účtu (GROUP nebo USER).
  - **UIManagerStatisticsForUser.cs** - řídící script pro typ účtu USER.
  - **UIManagerStatisticsForGroup.cs** - řídící script pro typ účtu GROUP.
  - **RowForGenerateTenExercise.cs** - pomocný script pro generování vlastních řádků v tabulce (pro porovnávání dat z 10. cvičení).
  - **RowForGeneratePlayedTimeOfPlayers.cs** - pomocný script pro generování vlastních řádků v tabulce (pro porovnávání odehraného času).
  - **ManagerNameAxis.cs** - pomocný script pro pojmenování os grafu.

- **Training** - složka se scripty používající ve **TrainingScreen** scéně:
  - **ManagerExercises.cs** - řídící script pro volbu zobrazených cvičení (1 až 10).
  - **UIManagerTraining.cs** - řídící script hlavního panelu zobrazeného během trénování.
  - **UIManagerBeforeExercisePanel.cs** - řídící script panelu zobrazeného před samotným tréninkem. Panel slouží především pro nastavení animací daného cvičení.
  - **UIManagerAfterExercisePanel.cs** - řídící script panelu zobrazeného po tréninku. Panel slouží pro zobrazení grafů a shrnutí proběhnutého tréninku.
  - **PointForAnimation.cs** - script pro definování animací puků mezi body.
  - **ControlSensorPanel.cs** - řídící script panelu pro zobrazení stavu senzorů.
  - **RecordData.cs** - script pro zaznamenávání dat ze senzorů pro zpětnou vizualizaci a spočítání dalších veličin pro uložení do databáze.

- **Utilities** - složka s ostatními třídami a scripty, které jsou využívány aplikací.
  - **Constants.cs** - třída s konstantami.

- **`CSVReader.cs`** - třída pro načtení dat z CSV (Comma-separated values) souboru - používáno během trénování.
- **`CSVWriter.cs`** - třída pro uložení dat z CSV souboru - používáno během trénování.
- **`DialogPanelManager.cs`** - script pro zobrazení dialogu.
- **`Exercise.cs`** - výčtový typ všech typů cvičení.
- **`FixedSizedQueue.cs`** - implementovaná vlastní struktura fronty, která má pevnou velikost.
- **`LoadingPanelManager.cs`** - script pro zobrazení panelu informující o načítání.
- **`MeasuringData.cs`** - třída představující návrhový vzor přepravka pro naměřená data během cvičení z akcelerometru, ale také pro výstupní data Fourierovy transformace.
- **`MeasuringFrequency.cs`** - třída pro počítání frekvence pohybů s pukem prostřednictvím knihovny DSPLib.
- **`SceneSwitcher.cs`** - script umožňující přepnutí do jiné scény.
- **`VisualizeFeedback.cs`** - script vizualizující zpětnou vazbu ze senzorů.
- **`Localization`** - složka s 8 scripty a třídami představující lokalizační framework. Pro snadnější popis a lepší pochopení jsou scripty a třídy popsány z pohledu jako celek v kapitole 7.4.1 o lokalizačním frameworku.
- **`HardwareControllers`** - složka s dvěma důležitými scripty pro komunikaci se senzory.
  - **`MPU6050Controller.cs`** - script s implementovanou komunikací se senzorem WT61C.
  - **`NeuroSkyController.cs`** - script komunikující s čelenkou MindWave Mobile 2 prostřednictvím knihovny NeuroSkyUnityThinkGearPlugins.

7.3 Diagramy tříd z pohledu scén

Aplikace obsahuje 52 tříd (scriptů) a 9 obrazovek (scén), do kterých se uživatel může dostat. Jedná se o přihlašovací obrazovku, hlavní menu, nastavení, nastavení čelenky NeuroSky, nastavení senzoru WT61C, analýza dat senzoru WT61C, statistiky, profil a trénink.

Tato kapitola popisuje jednotlivé scény **UML diagramy**, konkrétně diagramy tříd či scriptů (charakteristické pro Unity engine). Důvodem, proč se nejedná o jeden diagram je, že by nebyl čitelný.

Stane se, že na některých diagramech nebudou zobrazeny ostatní třídy, ale jen ty, které daný celek zastupují. **Ostatní důležité třídy budou popsány v dalších diagramech pro lepší čitelnost v následující kapitole 7.4** o diagramech tříd ostatních funkčních celků, např. lokalizační framework[1].

V diagramech budou scripty a třídy vyznačeny konkrétní barvou:

- **Šedivá barva** - script, který je charakteristický pro Unity engine. Dědí od třídy MonoBehaviour (není vyznačeno v UML diagramech pro přehlednost). Pro spuštění tohoto scriptu je nutné, aby byl připnut jako komponenta na některém objektu ve scéně.
- **Tyrkysová barva** - definuje klasickou třídu v C#.
- **Červená barva** - hlavní script, který komunikuje s dalšími scripty či třídami. V tomto případě se jedná o UIManager scripty.

Zde je také zmíněno, co je typické pro většinu scén. Většina scén používá scripty a třídy:

- **Třída `LocalizationCache`** - třída, která zastupuje lokalizační framework, který má vlastní diagram v kapitole 7.4.1 a slouží pro získání správného textu ve zvoleném jazyce.
- **Script `DialogPanelManager`** - script, který umožňuje zobrazit dialog se vstupními daty (titulek, zpráva).
- **Script `LoadingPanelManager`** - script, který umožňuje zobrazit panel s textem informující o načítání.
- **Script `SceneSwitcher`** - script umožňující přechod do jiné scény.

[1] Softwarová struktura, která slouží jako podpora při programování.

7.3.1 Diagramy hlavních scén

V této kapitole je popsána scéna přihlašovací obrazovky a scéna hlavního menu. UML diagram je znázorněn na **obrázku 10.4**, který je přiložen v příloze.

Řídícím scriptem této scény je script `UIManagerVerification.cs`, který jak už bylo řečeno, využívá panely pro zobrazení hlášky a informování o načítání. Pro přihlášení do aplikace je využívána třída `DatabaseController.cs`, která komunikuje s databází **(více v kapitole 7.4.3 o databázovém kontroleru)**.

Menu aplikace má pod kontrolou script `UIManagerMainMenu.cs` komunikující s třídou `ManageAccount.cs`, která uchovává informace o přihlášeném uživateli. Jako většina UIManager scriptů, také tento pracuje se scriptem `SceneSwitcher.cs`, který umožňuje přepínat mezi ostatními scénami.

7.3.2 Diagram scén pro nastavení

Celkové nastavení aplikace či nastavení senzorů se týká hned tří scén, kterým odpovídá diagram tříd **10.5 přiložený v příloze.** Jedná se o scénu `SettingsScreen, MPU6050ControlScreen a NeuroSkyControlScreen`. Do posledních zmiňovaných scén se uživatel dostane ze Settings scény.

Scénu Settings řídí script `UIManagerSettings.cs`, který dále komunikuje se scripty `NeuroSkyController.cs a MPU6050Controller.cs` za účelem rychlého připojení senzorů. `NeuroSkyController.cs` komunikuje s čelenkou MindWave Mobile 2 a `MPU6050Controller.cs` komunikuje s WT61C senzorem.

Scéna NeuroSkyControlScreen je především určena pro ladění připojení čelenky MindWave Mobile 2. Danou scénu řídí script `UIManagerNeuroSkyControl.cs` pracující se scriptem `NeuroSkyController.cs`, jenž pracuje s knihovnou komunikující s čelenkou.

Scéna `MPU6050ControlScreen` je především zaměřena na ladění připojení senzoru WT61C s čipem MPU6050, po kterém je také pojmenovaná. Tuto scénu řídí script `UIManagerMPU6050Control.cs` se scriptem `MPU6050Controller.cs`.

Další scripty a třídy propojené s `MPU6050Controller.cs` jsou popsány v kapitole 7.4.2 o diagramu poukazující na měření frekvence či komunikace s daným senzorem.

7.3.3 Diagram scény pro analýzu dat z WT61C

Jedná se o scénu `AnalysisMPU6050DataScreen`, která byla založena primárně pro analýzu dat, ale je přístupná i uživateli, který zde může vidět data (vizualizována v grafech), která jsou ze senzoru přijímána. Budu zde popisovat **UML diagram 10.6**.

Důležitým scriptem je `VisualizeFeedback.cs`, který je využíván i v tréninkové scéně. Script se stará o zpětnou vazbu uživateli (zobrazuje frekvenci, ale i soustředěnost v tréninkové scéně). Dalším důležitým scriptem je `MPU6050Controller.cs`, který je určen pro komunikaci se senzorem WT61C a script `MeasuringFrequency.cs`, který se zaměřuje na počítání frekvence prostřednictvím knihovny.

Další scripty a třídy propojené s `MPU6050Controller.cs` jsou popsány v kapitole 7.4.2 o diagramu poukazujícím na měření frekvence či komunikace s daným senzorem.

7.3.4 Diagram scény ProfileScreen

Scéna zobrazuje údaje o uživateli. Jedná se o popis diagramu tříd na **obrázku 10.7** v příloze.

Zde je nutné dělit funkcionalitu dle typu účtu, kde každý typ účtu obsahuje odlišná data. Pro každý typ účtu je tudíž jiný UIManager script. Pro typ účtu USER je vygenerované Graphic User Interface[2] (GUI), které řídí script `UIManagerProfileForUser.cs`. Pro typ účtu GROUP je řídícím scriptem `UIManagerProfileForGroup.cs`. Aktivování daného UIManager scriptu dle typu účtu řeší script `UISwitchManagerProfilePanels.cs`.

Oba řídící scripty používají třídu `DatabaseController.cs` pro získání dat uživatele **(více v kapitole 7.4.3 o databázovém kontroleru)**. Řídící script pro typ účtu USER používá třídu `RD_User`, která plní funkci přepravky uživatelských dat. Řídící script pro účet typu GROUP používá

[2]Grafické uživatelské rozhraní zpracovávající vstupy od uživatele či výstupy pro zobrazení dat.

script `RowForGeneratePlayers.cs`, jenž je pomocným scriptem pro vygenerování dat do tabulky.

Script `AccountManage.cs` je využíván v každém řídícím scriptu pro získání referencí na uživatelská data, která nebyla nutná stáhnout z databáze.

7.3.5 Diagram scény StatisticsScreen

Scéna zaměřující se na vizualizaci statistik do grafů či tabulek. Zde se jedná o popis UML diagramu tříd na **obrázku 10.8** v příloze.

Jako u scény `ProfileScreen` i zde se dělí scéna na dvě části a to podle typu účtu. O aktivování správného panelu a jeho příslušného UIManager scriptu se stará `UISwitchManagerStatistiscPanel.cs` script. Řídící scripty `UIManagerStatisticsForUser.cs` a `UIManagerStatisticsForGroup.cs` používají třídu `DatabaseController.cs` pro získání dat z databáze. **Třídy s prefixem RD** představují přepravky dat pro získaná data z databáze do vizualizace **(více v kapitole 7.4.3 o databázovém kontroleru)**.

Další pomocné a jednoduché scripty, které můžeme vidět v diagramu, jsou `RowForGenerateTenExercise.cs` a `RowForGeneratePlayedTimeOfPlayers.cs` pro generování tabulky. V prvním zmíněném případě se jedná o tabulku s vygenerovanými daty z 10. cvičení (používá se v obou typech účtu). Další tabulka s druhým jmenovaným scriptem zobrazuje odehranou dobu hráčů daného GROUP účtu. Pro seřazení dat dle hodnot a zobrazení v tabulkách slouží v obou případech vnořené třídy `UserTenExerciseData.cs` a `GroupTenExerciseData.cs`.

Třída `Constants.cs` obsahuje konstanty napříč celým projektem. Třída `AccountManage.cs` je využívána pro získání informací o typu účtu, ale také pro získání informací, které nejsou třeba získat z databáze (např. jméno uživatele).

7.3.6 Diagram scény TrainingScreen

`TrainingScreen` scéna je tou nejdůležitější scénou v této aplikaci. Tato scéna slouží pro trénink, při kterém je snahou vizualizovat zpětnou vazbu, pokud je k počítači připojené IoT zařízení. Zde se jedná o popis UML diagramu tříd na **obrázku 10.9** v příloze.

Scéna se dělí do tří stavů. Prvním stavem je myšleno zobrazení panelu (po vybrání cvičení) s řídícím scriptem `UIManagerBeforeExercisePanel.cs`. Tento panel slouží pro nastavení animací v tréninku a také drží odkaz na zvolené cvičení výčtovým typem z třídy `Exercise.cs`. Také se zde mírně liší GUI dle typu účtu (podle scriptu`AccountManage.cs`). Po spuštění cvičení přichází na řadu řídící script `UIManagerTraining.cs`.

Script `UIManagerTrainig.cs` se stará o animace a průběh tréninku. Scriptem `VisualizeFeedback.cs` zajišťuje vizualizaci zpětné vazby ze senzorů, jejichž řídícími scripty jsou `NeuroSkyController.cs` (čelenka MindWave Mobile 2) a `MPU6050Controller.cs` (senzor WT61C). Pro měření a počítání frekvence puku slouží script `MeasuringFrequency.cs`.

Jeden z dalších důležitých scriptů je `RecordData.cs`, který zaznamenává vizualizovaná data do souboru (prostřednictvím třídy `CSVWriter.cs`), aby nedocházelo k zahlcování operační paměti. Hlavním účelem tohoto scriptu je bezpečné a průběžné ukládání dat pro zpětné načtení do vizualizace průběhu tréninku.

Po skončení cvičení se objeví panel s řídícím scriptem `UIManagerAfterExercisePanel.cs`, jenž načte zaznamenaná data a vizualizuje je do grafů. Dálé je používán script `DatabaseController.cs` pro možné uložení vypočtených dat (např. průměrná frekvence) zaznamenaných během tréninku. **Více o databázovém kontroleru a třídách s prefixem RD v kapitole 7.4.3**.

7.4 Diagramy tříd ostatních funkčních celků

Zde popíšu diagramy důležitých funkčních celků tříd a scriptů, které jsou využívány ostatními třídami a UIManager scripty, které jsou řídícími scripty scén (popisovány v předešlé kapitole 7.3 o diagramech tříd z pohledu scén). V diagramech budou scripty a třídy vyznačeny konkrétní barvou:

- **Šedá barva** - script, který je charakteristický pro Unity engine. Dědí od třídy MonoBehaviour (není vyznačeno v UML diagramech pro přehlednost). Pro spuštění tohoto scriptu je nutné, aby byl připnut jako komponenta na některém objektu ve scéně.
- **Tyrkysová barva** - definuje klasickou třídu v C#.
- **Červená barva** - hlavní script či třída, se kterými mohou v rámci projektu komunikovat i jiné třídy.
- **Žlutá barva** - knihovna.

7.4.1 Diagram tříd - lokalizační framework

Tento popis se týká **UML diagramu 10.9**, který je přiložený v příloze o lokalizačním frameworku. Část byla nazvaná Framework, protože už od začátku implementace byl psán tak, aby nebyl závislý na tomto projektu. Jedná se o funkční celek či softwarovou strukturu, která by se dala použít i v jiných Unity projektech.

Důležitou třídou je `LocalizationCache.cs`, která drží slovník všech klíčů s příslušným textem ve vybraném jazyce. Pokud uživatel přepne do jiného jazyka, tato třída načte jiný slovník dat. Načtení probíhá ve třídě `WorkJSON.cs`, která vrací odkaz na objekty třídy `LocalizationItem.cs`. Tato třída obsahuje pole (třída `SystemText.cs`) všech klíčů s textem v konkrétním zvoleném jazyce. Díky třídě `LocalizationCache.cs` se lze přes klíč také dotázat z ostatních scriptů a tříd na text v příslušném jazyce (využívají především UIManager scripty).

Další nedílnou součástí frameworku je script `LocalizationManager.cs`, který vlastní všechna textová pole ve scéně, kterým přiřazuje správný text dle klíče. Textová pole představuje script `LocalizedText.cs` (je pověšen ve scéně na každém objektu s textovým polem), který dědí od scriptu `ALocalizationElement.cs`.

7.4.2 Diagram tříd - měření frekvence

Další důležitou součástí tohoto projektu je měření frekvence pohybů s pukem ze senzoru WT61C. Tato část je součástí jak Training scény, tak `Analysis-MPU6050DataScreen` scény. Dle mého uvážení jsem se rozhodl tuto část popisovat v této kapitole prostřednictvím UML diagramu tříd **na obrázku 10.10**, který je přiložený v příloze.

Hlavní script, který komunikuje se senzorem, je `MPU6050Controller.cs`. Senzor neposílá žádná data o frekvenci, tudíž je nutné si danou veličinu vypočítat. Počítání frekvence nám umožňuje třída `MeasuringFrequency.cs` **s knihovnou DSPLib zajišťující výpočet frekvence s využitím Fourierovy transformace**. Výpočet dané veličiny není jednoduchý, a proto je nutné si udržovat naměřená data za poslední dvě sekundy pro výpočet dané frekvence. Data jsou zaznamenávána do tří front (pro každou osu X, Y a Z) typu třídy `FixedSizeQueueDouble.cs`, která má přesnou velikost dle počtu snímků za poslední dvě sekundy. Ve třídě `MeasuringFrequency.cs` jsou dále vypočítávány frekvence ze zaznamenaných dat.

Třída `MeasuringData.cs` představuje přepravku dat, která jsou výstupem z výpočtu frekvence, za účelem předání naměřených dat scriptu `MPU6050-Controller.cs` a jejich vizualizace.

7.4.3 Diagram tříd - databázový kontroler

Za účelem většího porozumění kolem získávání a ukládání dat z databáze slouží tato kapitola, která se týká především UML diagramu **na obrázku 10.11**, který je v přiložený v příloze.

Důležitou třídou celého projektu je databázový kontroler (`Database-Controller.cs`), který jako jediný komunikuje s využitím knihovny FirebaseSDK s databází. Každá metoda v této třídě vrací nebo ukládá data primitivního typu nebo struktury (návrhový vzor přepravka), které jsou značené **prefixem RD - Realtime Database.**

Databázový kontroler dále používá konstanty a výčtový typ. Také využívá třídu `AccountManage.cs`, která poskytuje data o účtu (např. typ účtu).

7.5 Použité knihovny

Zde zmíním použité knihovny, které jsou využívány pro ulehčení práce v tomto projektu.

1. **Dynamic Line Chart** [14] - knihovna poskytující dynamický graf, využíván v řídícím scriptu `UIManagerAccelerometerAnalysis.cs` scény `AnalysisMPU6050DataScreen` pro vizualizaci živých dat ze senzoru. Jedná se o graf s instancí scriptu `DD_DataDiagram`.

2. **Awesome Charts** [12] - knihovna pro vizualizaci statického grafu. Je využívána v řídících scriptech scén `TrainingScreen` (`UIManagerAfterExercisePanel.cs`), `AnalysisMPU6050DataScreen` (`UIManagerAccelerometerAnalysis.cs`) a `StatisticsScreen` (`UIManagerStatisticsForUser.cs`).

3. **CrystalFramework** [13] - knihovna sloužící pro platformu. Ořízne obrazovku telefonu tak, aby se nepřekrývala s jinými výřezy obrazovky, jako je např. výřez na selfie kameru. Využívá se v každé scéně jedním scriptem `SafeArea.cs`, který je pověšen vždy na objektu s Canvasem (komponenta pro GUI) a řeší daný problém s výřezem.

4. **DSPLib** [4] - knihovna obsahující metody pro výpočet Fourierovy transformace. Na webu tohoto zdroje je i jednoduchá dokumentace k použití této knihovny, kterou bylo nutné samozřejmě nastudovat. Využívá se ve třídě `MeasuringFrequency.cs` s knihovní třídou `DSPLib.cs`.

5. **NeuroSkyUnityThinkGearPlugins** [10] - knihovna, která slouží ke komunikaci s čelenkou MindWave Mobile 2. Balíček je Unity knihovnou, takže se jednalo o jednoduchou integraci. Využívá se ve scriptu `NeuroSkyController.cs` a to s knihovním scriptem `TGCConnectionController.cs`.

7.6 Popis komunikace se senzorem WT61C

Zde popíšu script `MPU6050Controller.cs`, jenž slouží pro komunikaci se senzorem WT61C. Jedná se o jednu z těch zajímavějších implementací této práce.

Aplikace nevyžaduje připojení k senzoru přes sériový port vícekrát najednou, proto jsem se rozhodl implementovat třídu dle **návrhového vzoru jedináček** (singleton). Jeden z dalších důvodů je, že je nutné znát stav senzoru a mít možnost připojení i z jiných scén aplikace (přístup ke stejné instanci). Níže můžete vidět kód představující statickou metodu vracející instanci na jedináčka:

```
public static MPU6050Controller Instance {
    get {
        if (_instance == null) {
            _instance = FindObjectOfType<MPU6050Controller>()
    ;
            if (_instance == null) {
                GameObject go = new GameObject();
                go.name = "MPU6050Controller";
                _instance = go.AddComponent<MPU6050Controller
    >();
                DontDestroyOnLoad(go);
            }
        }
        return _instance;
    }
}
```

Zde jsou popsány některé důležité metody tohoto scriptu:

- **`public IEnumerator ConnectToSerialPort_Coroutine(string portName = null)`** - metoda pro možné připojení senzoru (volána z více míst programu). Na začátku se zde volá metoda `CheckOpen()`, která se pokouší o připojení přes sériový port. Následně jsou zde vytvořena vlákna pro příjem dat ze senzoru a pro počítání snímkovací frekvence (FPS), která je využívána pro měření frekvence pohybů se senzorem ve třídě `MeasuringFrequency.cs`.
- **`public bool Disconnect()`** - metoda, která odpojí sériovou komunikaci mezi zařízením a senzorem.
- **`private void CheckOpen(string portName, int boudRate)`** - metoda, která vytváří instanci třídy `SerialPort` a pokouší se o připojení s nastavením určeným parametry této metody. Pokud dojde k úspěšnému připojení, tak atribut `IsOpen` (této instance) je roven hodnotě `true`.

- **`private void Reading()`** - metoda vlákna, která v každém snímku aplikace volá metodu `SerialPort_DataReceived()` pro příjem dat.
- **`private void SerialPort_DataReceived()`** - metoda, která se stará o příjem dat ze sériového portu v daný okamžik, kdy je zavolána. Níže můžete vidět kus kódu této metody, který je v bloku `try-catch` pro možné vyhození chyb. **Jedná se o čtení bytů do třídního atributu `buffer`:**

```
1     try {
2         usLength = (UInt16)_serialPort.Read(buffer,
    bufferLength, 700);
3        ...
4
```

  Následující kód se stará o předání přečtených bytů metodě `DecodeData()`. Tato část se zmiňovanou metodou je implementována podle přiložených zdrojových kódů u dokumentace k jinému softwaru pro tento senzor (z důvodu neznámého komunikačního protokolu a komplexnosti daného problému), viz zdroj [17].

- **`private void DecodeData(byte[] byteTemp)`** - metoda převzatá z větší části z dokumentace k danému senzoru, viz zdroj [17]. Slouží pro dekódování bytů (`byte[] byteTemp`)) do formálních dat, třídních atributů, se kterými je snadnější práce, např:
  - **`Acc`** - pole obsahující data z akcelerometru,
  - **`Gyr`** - pole obsahující data z gyroskopu,
  - **`Angle`** - pole obsahující informace o úhlu natočení.

  Metoda je vcelku komplikovaná, ale je jasné, že vstupním parametrem je pole bytů a **první byte rozhoduje o jakou uloženou veličinu v poli `byteTemp` se jedná.** Pokud první byte je roven:
  - **`0x50`** - je přijímán čas, jak dlouho senzor běží,
  - **`0x51`** - je přijímána akcelerace,
  - **`0x52`** - jsou přijímána data z gyroskopu,
  - **`0x53`** - jsou přijímána data informující o úhlu natočení.

- **`public void StartMeasuringFrequency(DialogPanelManager dialog, bool showDialogAfterDisconnect)`** - po zavolání této metody dojde ke spuštění měření frekvence pohybů se senzorem, pokud je senzor připojen. Měření frekvence je komplexnější funkce (Fourierova transformace) a tudíž se nejedná o automatické spuštění hned po připojení senzoru. Klíčovou roli zde hraje zavolání metody `StartMeasuring()` ze třídy `MeasuringFrequency`, která má měření frekvence na starosti. Parametry jsou `DialogPanelManager`, který se zobrazí, pokud senzor není připojen a hodnota `showDialogAfterDisconnect` je `true`.

- **`public void StopMeasuringFrequency()`** - metoda pro zastavení měření frekvence.

7.7 Popis třídy s Fourierovou transformací

Jedna z dalších zajímavých části práce je třída `MeasuringFrequency`. Tato třída počítá jednu z těch nejdůležitějších veličin pro zpětnou vazbu uživateli ze senzoru WT61C - frekvenci pohybů se senzorem. Veličina je počítána z naměřených dat ze senzoru za poslední dvě sekundy. Odkaz na vstupní data (akceleraci) drží script MPU6050Controller. Po spočtení frekvence je hodnota nastavena instancí třídy MPU6050Controller přes setter metodu. Popis důležitých metod v popisované třídě:

- **`public static MeasuringFrequency Instance`** - get metoda, která vrací jedináčka - jedinou instanci této třídy. Třída je implementována dle návrhového vzoru jedináček (singleton).

- **`public void StartMeasuring()`** - metoda, která spouští měření neboli počítání frekvence. Vytváří a spouští se zde vlákno, ve kterém dochází k samotnému měření.

- **`public void StopMeasuring()`** - metoda pro zastavení měření frekvence.

- **`private void RunStartMeasuringFrequency(object controller)`** - metoda vlákna, která běží do doby, než bude zavolána metoda pro zastavení nebo dojde k odpojení senzoru. **Volá se zde metoda `calculateFrequencyFromData(..)` pro výpočet frekvence a to rovnou dvakrát pro každou osu zvlášť (pro naměřená data osy X a Y).** Pro 3. osu Z není nutné frekvenci počítat, protože je malá pravděpodobnost, že by hokejista dribloval na hraně s pukem a

pokud ano, tak ne dostatečně dlouho (není problém dle implementace dodat). Parametrem je objekt typu MPU6050Controller (přetypování z `object`), který obsahuje naměřenou akceleraci pro výpočet frekvence.

- **`public MeasuringData calculateFrequencyFromData (FixedSizeQueueDouble accelenometerData, float secondMeasuing, int countMembers)`** - metoda pro výpočet frekvence. Nejprve je nutné si připravit vstupní data pro funkci z knihovny. To znamená vzít přesný počet prvků (**`countMembers`**) z naměřených dat (**`accelenometerData`**) za poslední dvě sekundy (**`secondMeasuring`**). Níže můžete vidět použití knihovny DSPlib pro nalezení frekvence dle dokumentace [4]:

```
1  /*priprava vstupnich dat*/
2  System.UInt32 zeroPadding = 1024-(UInt32)
       dataForCalculating.Length;
3  double[] wCoefs = DSP.Window.Coefficients(DSP.Window.
       Type.Welch, (UInt32)dataForCalculating.Length);
4  double[] wInputData = DSP.Math.Multiply(
       dataForCalculating, wCoefs);
5  double wScaleFactor = DSP.Window.ScaleFactor.Signal(
       wCoefs);
6
7  /*aplikovani rychle fourierovy transformace*/
8  DSPLib.FFT fft = new DSPLib.FFT();
9  fft.Initialize((UInt32)dataForCalculating.Length,
       zeroPadding);
10 Complex[] cSpectrum = fft.Execute(wInputData);
11
12 /*vystup dat osy Y*/
13 double[] lmSpectrum_Y = DSPLib.DSP.ConvertComplex.
       ToMagnitude(cSpectrum);
14 lmSpectrum_Y = DSP.Math.Multiply(lmSpectrum_Y,
       wScaleFactor);
15 /* vystup dat osy X*/
16 double[] freqSpan_X = fft.FrequencySpan(((UInt32)
       dataForCalculating.Length));
17
18 /*nalezeni hledane veliciny (lomeno secondMeasuring,
       protoze chceme jen frekvenci za jednu sekundu) */
19 double maxFreq = (FindFrequency(lmSpectrum_Y, freqSpan_X
       )/secondMeasuring);
20 double maxAmplitude =  Analyze.FindMaxAmplitude(
       lmSpectrum_Y);
```

Metoda pak vrací typ třídy `MeasuringData`, který je přepravkou pro data, která jsou výstupem výpočtu.

- **`private double FindFrequency(double[] fSpan_Y, double[] inData_Y)`** - metoda, která nalezne nejvyšší hodnotu frekvence z výstupních dat. Volá se ve výše zmíněné metodě.

7.8 Popis třídy AccountManage.cs

Třída, která obsahuje metody pro inicializaci uživatelských a statistických dat do databáze. Také funguje na principu přepravky, kde třídními atributy jsou uživatelská data a typ účtu přihlášeného uživatele.
Důležité třídní atributy:

- **`FirebaseAuth Auth`** - atribut odkazující na instanci poskytující funkce s daným účtem (od Firebase knihovny).
- **`FirebaseUser User`** - atribut odkazující na instanci poskytující informace o přihlášeném účtu (od Firebase knihovny).
- **`RD_User ActuallUser`** - atribut držící informace o uživatelském účtu typu USER, který uživatel zadá prostřednictvím této aplikace.
- **`RD_Group ActuallGroup`** - atribut držící informace o uživatelském účtu typu GROUP, který uživatel zadá prostřednictvím této aplikace.
- **`RD_TypeAccount TypeAccount`** - atribut držící informace o typu účtu, který je přihlášen.

Důležité metody:

- **`public static AccountManage Instance()`** - metoda vracející jedináčka - jediná instance této třídy (vždy je přihlášen jen jeden uživatel).
- **`private AccountManage()`** - konstruktor této třídy, kde jsou inicializovány především třídní atributy Firebase instancí.
- **`private void AuthStateChanged(object sender, System.EventArgs eventArgs)`** - metoda, která reaguje na změnu stavu přihlášeného účtu (přihlášení, odhlášení).

Metody, které jsou volány z jiných částí projektu pro nastavení příslušných atributů. Ať už pro inicializování nebo pro aktualizování:

- **`public void SetUserData(DataSnapshot data)`** - parametrem je objekt z databáze, který je přetypován na `RD_User`.
- **`public void SetUserData(RD_User user)`**
- **`public void SetGroupData(DataSnapshot data)`**- parametrem je objekt z databáze, který je přetypován na `RD_Group`.
- **`public void SetGroupData(RD_Group user)`**

7.9 Popis třídy DatabaseController.cs

Tato třída se chová jako model v MVC architektuře, který komunikuje s databází. Jako předešlé zmíněné třídy i tato třída je implementována podle návrhového vzoru jedináček - není potřeba vlastnit více objektů této třídy.

Třída obsahuje spoustu krátkých metod pro uložení nebo načtení dat z databáze. Metody jsou rozděleny do tří kategorií pomocí bloků kódu `#region` pro lepší přehlednost:

- **`TYPE_ACCOUNT`** - blok obsahující metody kolem získání a uložení typu účtu.
- **`USERS`** - blok obsahující metody pro uložení a načtení dat typu účtu USER.
- **`GROUPS`** - blok obsahující metody pro uložení a načtení dat typu účtu GROUP.

Není nutné zde popisovat všechny metody, které si jsou svou strukturou podobné, a proto zde přikládám kód jedné metody pro uložení struktury `RD_User`:

- **`public IEnumerator Save_User(RD_User user, System.Action functionAfterCorrectLoading, System.Action<AuthError> errorSatetFunction)`** - metoda pro uložení instance třídy `RD_User` do databáze. Jedná se o `Coroutine`[3] funkci, díky návratovému typu `IEnumerator`. Po následném uložení se čeká na odezvu `yield` instrukcí. Pokud došlo k chybě, zavolá se metoda poskytnutá parametrem `errorSatetFunction`. Nedojde-li k chybě, zavolá se metoda `functionAfterCorrectLoading`.

[3]Funkce, kterou lze zastavit, dokud nedojde k ukončení `yield` instrukce.

8 Testování

Kapitola se zabývá testováním aplikace. Po konci vývoje aplikace je nutné ověřit funkčnost a zda aplikace splňuje zmiňované požadavky.

Výsledný produkt je testován beta testy neboli uživatelskými akceptačními testy:

- **Integrační testy** - testují interakci komponent neboli to, jak dokáží jednotlivé celky spolupracovat.
- **Uživatelské akceptační a beta testy** - beta testy jsou zaměřené na testování produktu koncovými uživateli. Uživatelské akceptační testy jsou často vykonávány podle předem domluvených scénářů, na kterých se domluvily obě strany. Důležité je ověřit, zda je produkt smysluplný a pro dané účely použitelný.

8.1 Integrační testy

Testování aplikace probíhalo s využitím `Unity Test Framework (UTF)`, což je knihovna, která poskytuje psát testy jak v **Edit módu**, tak v **Play módu**.

Testy v Edit módu jsou především určené pro unit (jednotkové) testy, které testují jednotlivé třídy a metody. Testy v Play módu, které jsem využíval, jsou zaměřeny na spolupráci jednotlivých komponent.

Bylo napsáno 6 tříd obsahující testy, které jsou samostatně zaměřené na jednu z obrazovek aplikace. Testy nalezneme ve složce `Tests`:

1. `TestVerificationScreen.cs` - jsou zde testy cílené především na přihlašování do aplikace. Je zde několikrát a vždy jiným způsobem otestováno chybné přihlášení. Následně je zde také test na přihlášení s validními daty.
2. `TestMainMenuScreen.cs` - testy zaměřené především na hlavní menu obrazovky. Je zde otestováno odhlášení uživatele a přepínání do jiných scén aplikace.
3. `TestProfileScreen.cs` - je zde napsán jeden delší test, který ověří funkčnost uložení a načtení dat v profilové scéně. Jedná se o test s

názvem `Test_SaveAndLoadData()`, který načte profilovou obrazovku, vloží data a následně uživatele z aplikace odhlásí. Dále uživatele znovu přihlásí pro zobrazení a ověření uložených dat.

4. `TestSettingsScreen.cs` - jsou zde testy pro obrazovku nastavení. Níže můžete vidět příklad testu pro ověření přepnutí do anglického jazyka:

```
[UnityTest, Order(2)] //test spusten jako druhy
public IEnumerator Test_SwitchLanguage_EN() {
    yield return new WaitForSeconds(2f);
    LocalizationManager localizationManager = GameObject
   .FindObjectOfType<LocalizationManager>();
    //prepnuti jazyka
    localizationManager.ChangeLanguage_Button("en");
    yield return new WaitForSeconds(2f);
    //1. overeni
    Assert.AreEqual("Back", LocalizationCache.
   getlocalizationText("back"));
    //2. overeni
    Assert.AreEqual("No", LocalizationCache.
   getlocalizationText("no"));
    //3. overeni
    Assert.AreEqual("Error", LocalizationCache.
   getlocalizationText("error"));
}
```

5. `TestStatisticsScreen.cs` - v této třídě je napsán jediný test pro ověření zobrazených dat v grafu.

6. `TestTrainingScreen.cs` - tato třída obsahuje test na spuštění všech cvičení a ověření, zda cvičení probíhá.

Také byl napsán test `TestCase_SaveAndLoadPalyedTimeInStats()`, ve třídě **`TestCase.cs`**, který se zaměřuje na otestování uložení a zobrazení odehraného času. Test prochází obrazovku statistik a trénování, kde navíc spustí několik cvičení a následně ověří uložení dat tím, že si nechá zobrazit graf s očekávanou hodnotou ve statistikách.

Pro většinu testů je nutné být v aplikaci přihlášen, a proto byla vytvořena třída **`TestParametr.cs`** obsahující konstanty s přihlašovacími údaji. Než dojde k testování, je nutné v této třídě zadat přihlašovací údaje.

8.1.1 Vyhodnocení integračních testů

Díky těmto testům došlo jak k otestování grafického rozhraní, tak k otestování interakcí modulů jako jsou metody pro zobrazení grafů, databázový kontroler (pro uložení a čtení dat z databáze) nebo přihlášení do aplikace.

Všechny napsané testy proběhly úspěšně a zároveň se staly do budoucna prvním testovacím scénářem pro ověření stávajících funkcí. Není však vyloučené, že rozšiřováním aplikace se testy mohou stát neplatné.

8.2 Uživatelské akceptační a beta testy

Pro zhodnocení výsledků bylo potřeba vytvořit jednotný scénář, který obsahuje 18 úkolů (označení PC) pro desktopové zařízení a 1 nepovinný úkol pro mobilní zařízení (označení MOB). Úkoly jsou následující:

PC-1
Spusťte aplikaci Hockey Coach.exe a ověřte, že jste připojeni k internetu.

PC-2
Po prvním spuštění by se měl zobrazit dialog pro volbu jazyka. Proto zvolte jazyk, ve kterém budete schopni aplikaci používat.

PC-3
Přihlaste se prostřednictvím přidělených přihlašovacích údajů.

PC-4
Po přihlášení se ocitnete v hlavním menu aplikace. Přejděte do sekce profil přes ikonku postavy vpravo dole.

PC-5
Vyplňte potřebné údaje - přezdívku, pohlaví, věkovou kategorii a uložte! Poté klikněte na tlačítko OK v informujícím dialogu o uložení a vraťte se zpět do menu.

PC-6 – Vlastníte-li senzor WT61C pro měření akcelerace

1. Přejděte do nastavení přes ozubené kolečko (tlačítko).
2. Přejděte do nastavení senzoru WT61C přes tlačítko MPU6050 controller.

3. Propojte zde aplikaci se senzorem. Předtím je nutné propojit senzor s počítačem přes Bluetooth a znát port připojení. (připojení je popsáno v manuálu - v příručce).

4. Po připojení se vraťte zpět do obecného nastavení. Stav připojení můžete vidět v obecném nastavení.

PC-7– Vlastníte-li čelenku MindWave Mobile 2

1. Připojte čelenku přes tlačítko Připojit v tomto nastavení anebo v nastavení dané čelenky, do kterého se dostanete přes tlačítko NeuroSky controller.

2. Po připojení se vraťte zpět. Stav připojení můžete vidět v obecném nastavení.

PC-8
Vraťte se do hlavního menu a klikněte na tlačítko Trénink, kde se vám zobrazí sekce s deseti možnými cvičeními.

PC-9
Vyberte 1. cvičení.

PC-10
Zobrazí se vám dialog s volbou nastavení animací. Nastavení zvolte dle své věkové kategorie:

Věková kategorie	Čas, po kterém se změní matematický problém	Rychlost puku
Profesionál nebo U20	10	5
Ostatní	15	5

Tabulka 8.1: Tabulka kategorií

PC-11

1. Pokud vlastníte senzory, ověřte jejich připojení přes ovládací panel.

2. Klikněte na tlačítko start a trénujte po dobu 1 minuty.

PC-12
Zobrazí se statistiky z proběhnutého cvičení. Klikněte na tlačítko uložit!

PC-13
Vyberte 5. cvičení a opakujte kroky PC-10, PC-11 a PC-12.

PC-14 - Vlastníte-li senzor WT61C pro měření akcelerace

1. Vyberte 10. cvičení.
2. Před zahájením se vám ukáže podobný dialog, jako v předchozím případě. Je nutné zde vybrat typ cvičení.
3. Vyberte typ 60s.
4. Před kliknutím na tlačítko start ověřte připojení senzorů přes ovládací panel, pokud senzory vlastníte.
5. Klikněte na tlačítko start.

PC-15 - Vlastníte-li senzor WT61C pro měření akcelerace
Po uběhnutí 60 sekund se vám zobrazí statistiky. Klikněte na tlačítko uložit.

PC-16 – Vlastníte-li senzor WT61C pro měření akcelerace
Vyberte 10. cvičení s typem tréninku 30s a opakujte kroky dle předchozích bodů PC-15 a PC-16.

PC-17
Přejděte do nastavení ze sekce hlavního menu, kde se vám načte graf s aktuálním odehraným časem v tomto měsíci.

PC-18 – Pokud proběhl trénink se senzorem WT61C
Zobrazte si kategorii 10. cvičení, vyberte svojí věkovou kategorii (nebo vše), vyberte svoje pohlaví (nebo vše), vyberte typ cvičení 60s nebo 30s a zobrazte statistiky.

MOB-1 – Vlastníte-li mobilní aplikaci
Lze tento scénář projít a vyzkoušet s mobilní aplikací bez připojení senzorů.

8.2.1 Vyhodnocení uživatelských testů

Vzhledem k situaci virové pandemie Covid-19 nedošlo k otestování aplikace hokejisty ve zmíněném hokejovém kempu. Aplikace bez senzorů byla otestována hokejovými trenéry, kteří si ji pustili na svém počítači. Ostatní jedinci, kteří k hokeji nemají tak blízko, testovali aplikaci na počítači (s připojenými senzory) s těmito parametry:

- Operační systém - Windows 10.
- Grafická karta - NVIDIA GeForce GTX 1080 - 4GB.
- CPU - Intel(R) Core(TM) i7-7700HQ CPU - 2.80GHz
- RAM - 16GB.
- Rozlišení obrazovky 1920 x 1080.

Pro výstup testů dle scénáře sloužil formulář, ve kterém jsou kromě zmiňovaných bodů z předchozí kapitoly i textová pole pro názor na aplikaci.

Níže jsou vypsané **záporné reakce** od těch, kteří mohli aplikaci otestovat:

- „V sekci profil bylo možné kliknout na tlačítko zpět, i v případě, že byl otevřený modál informující o úspěšném uložení profilu.“

  Řešení - dialogy byly potřeba přesunout do popředí, aby zakryly ostatní uživatelské rozhraní.

- „Nedalo se připojit přes Bluetooth k čelence od NeuroSky.“

  Řešení - výměna baterie.

- „Čelenka NeuroSky se odpojovala.“

  Řešení - výměna baterie nebo přesunutí tréninku na klidnější místo, kde nebude docházet k rušení spojení.

- „Déle jsem čekal na načítání statistik.“

  Řešení - nejedná se o vážný problém, ale s rozšířením práce by bylo vhodné tento problém vyřešit lepším návrhem databáze nebo jinou optimalizací.

- „Pokud uživatel zapomene heslo pro přihlášení, není zde možnost tento problém vyřešit“

  Řešení - doprogramován dialog (v přihlašovací obrazovce) pro možné resetování hesla - dojde k zaslání odkazu na email.

Zde jsou zmíněné **kladné reakce**, ale i návrhy pro rozšíření:

- „Uvítal bych vyhledávání statistik ostatních hráčů podle přezdívky.“
- „Přidal bych více modifikací cvičení.“
- „Zpětná vazba, díky které se hráči mohou porovnávat napříč kategoriemi.“

Níže můžete vidět obrázek zobrazující testování aplikace.

Obrázek 8.1: Testování aplikace dle scénáře

9 Zhodnocení dosažených výsledků

BCI hra s podporou akcelerometru neboli aplikace zvaná Hockey Coach, zaměřena na rozvoj herní inteligence, byla úspěšně navržena a implementována.

Nejprve bych poznamenal, že bylo plánováno poskytnout aplikaci k otestování hráčům hokejové školy, nicméně vzhledem k současné situaci kolem pandemie Covid-19 vznikla opatření, kvůli kterým nemohlo testování proběhnout. A tak došlo k otestování a akceptování aplikace ze strany samotných trenérů. Lze tedy konstatovat, že byly splněny všechny požadavky na tuto práci. Uživatelské testy dle scénářů proběhly vzhledem k situaci jen na pár jedincích, kteří poskytli důležitou zpětnou vazbu. Nebyly reportovány žádné vážnější chyby. Šlo jen buď o drobné nedokonalosti, které byli následně opraveny, anebo o nedostatky, které se mohou stát předmětem pro rozšíření práce.

Podstatné připomínky a nápady na vylepšení aplikace, které byly zmíněné v hodnocení (od těch, kteří měli možnost aplikaci otestovat), cituji zde:

- „Přidal bych možnost resetování svého dosaženého výsledku z 10. cvičení.“
- „Přidal bych více modifikací cvičení.“
- „Déle jsem čekal na načítání statistik.“

Také bych rád upozornil na integrační testování, které proběhlo vcelku uspokojivě, ale veškeré funkce systému nejsou testy pokryté. Hlavním důvodem nedostatků implementovaných testů je jejich náročnost a rozsah práce. Snahou bylo otestovat ty funkce, které se zdají být kritické. Budoucím rozšiřováním aplikace se testy mohou stát neplatné, ale některé naopak mohou složit pro opětovné ověření stávajících funkcí.

S čím jsem nebyl spokojen a doposud nejsem je jak analytická, tak implementační část kolem senzoru WT61C pro měření akcelerace. Z počátku se mi práce se senzorem zdála triviální, ale když došlo na to, vymyslet jaká data se budou uživateli vizualizovat, tak už jsem tak optimistický nebyl. Data, která

senzor poskytuje, nebyla natolik vhodná, a tak se analýza odvíjela i mimo rámec této práce. Po několika konzultacích jsem dospěl k závěru, že je nutné si prostudovat nepoznanou oblast, Fourierovu transformaci, pro výpočet potřebné frekvence pohybů se senzorem. Implementace této funkce nepřipadala v úvahu z důvodu náročnosti a rozsahu práce, a tak bylo potřeba nalézt vhodnou knihovnu a správně ji použít. Další záležitostí, se kterou jsem se potýkal v závěru bakalářské práce, byl problém s napájením senzoru. Tento problém se bohužel dotkl i testování, protože senzor musel být při testování připojen do napájení.

Jako výtku bych uvedl plynulost aplikace, která je negativně projevena připojenými senzory a následovným zpracováváním dat (počítání frekvence). I přesto, že došlo k implementaci vláken, dochází k občasnému přerušení kontinuálnosti animací, a proto by bylo vhodné se do budoucna seznámit s paralelním programováním v Unity enginu.

Dle zpětné vazby a především kvůli dominantní platformě na trhu, bych se chtěl do budoucna upínat především na mobilní aplikaci. Pro Android platformu je nutné doimplementovat komunikaci s IoT zařízeními.

Tato práce mi také přinesla nové zkušenosti s používáním služeb od Firebase, která též nabízí jednu ze služeb, kterou by bylo vhodné do budoucna integrovat. Jedná se o službu Crashlytics, která zaznamenává informace o padání aplikace a tak slouží především pro opravování problémů se stabilitou. Nejspíš je také nutné, aby systém prošel auditem z hlediska právního. Jsou zde totiž nejasnosti ohledně uchovávání osobních údajů v databázi od Firebase, přestože jsou data zabezpečená před vnějším zneužitím. Dále je do budoucna možnost přidat nový způsob ověření a zaregistrování se do aplikace prostřednictvím poskytovatelů třetích stran, např. Facebook, Google. Zároveň je také nutné, aby správce přidával uživatelské účty do databáze manuálně, a proto by bylo vhodné daný systém do budoucna propojit s automatizovanou formou, např. samotná registrace uživatele, webové rozhraní (po koupi aplikace) apod.

K programu byl také vytvořen **návod pro sestavení aplikace, systémové požadavky a uživatelská příručka**. Všechny tyto dokumenty jsou přiložené v příloze.

Vytvořená aplikace je otestována a připravena pro rozvoj herních dovedností.

10 Závěr

Cílem této práce bylo navrhnout a vytvořit jednoduchou aplikaci pro trénink hokejistů s podporou senzoru pro měření akcelerace a čelenky MindWave Mobile 2 pro měření kognitivních funkcí. Většina požadavků přicházela ze strany hokejových trenérů, které jsem si dovolil zmínit v úvodu této práce.

V analytické části jsem se seznámil s tréninkem v hokejové škole, prostudoval jednotlivé snímače a především analyzoval to, co budeme hráči vizualizovat jako zpětnou vazbu. Pro zobrazení zpětné vazby, z čelenky MindWave Mobile 2, jsem zůstal jen u jedné veličiny a tou je soustředěnost trénujícího hráče. Nad vizualizací zpětné vazby, ze snímače pro měření akcelerace, padlo více úvah, ale nakonec jsem dospěl k návrhu, který počítá frekvenci pohybů se senzorem. Oba návrhy pro zobrazení zpětné vazby z jednotlivých snímačů byly úspěšně implementovány.

Dle požadavků také došlo k propojení aplikace se systémem pro ověření identity uživatele. Tato služba byla integrována knihovnou Firebase, která navíc díky znalosti indentity poskytuje bezpečně ukládat data do databáze. Proto také došlo k návrhu a implementaci ukládání statistických dat pro možnost porovnávání uživatelů mezi sebou.

Program byl otestován jak integračními testy, tak uživatelskými testy dle scénáře několika osobami. Výsledkem testování jsou převážně kladné reakce a nové návrhy pro rozšíření. Při zhodnocení dosažených výsledků jsem nalezl drobné nedostatky navržené implementace, které jsem následně opravil. Při návrhu a implementaci byl největší komplikací senzor pro měření akcelerace. Nedisponoval vlastní knihovnou pro komunikaci a posílal jen základní veličiny, ze kterých bylo potřeba výslednou hodnotu frekvence dopočítat Fourierovou transformací.

Je podstatné zmínit, že byly splněny jak všechny požadavky ze strany hokejových trenérů, tak body zadání.

Literatura

[1] Bareš, M. *Kognitivní evokované potenciály* [online]. I. neurologická klinika, LF MU a FN u sv. Anny v Brně, 2011. [cit. 2020/16/1]. Dostupné z: https://www.csnn.eu/casopisy/ceska-slovenska-neurologie/2011-5-1/kognitivni-evokovane-potencialy-36052.

[2] Blažek, V. *Základy neurofyziologie a neuroanatomie člověka.* Aleš Čeněk, 2006. ISBN 80-86898-63-6.

[3] Google. *Firebase* [online]. Google, 2020. [cit. 2020/23/3]. Dostupné z: https://firebase.google.com/.

[4] Hageman, S. *DSPLib - FFT / DFT Fourier Transform Library for .NET 4* [online]. Steve Hageman, 2018. [cit. 2020/29/3]. Dostupné z: https://www.codeproject.com/Articles/1107480/DSPLib-FFT-DFT-Fourier-Transform-Library-for-NET-6.

[5] Kolářová, E. *Hra mozku a co na to říká samotný mozek?* [online]. 2019. [cit. 2019/28/12]. Dostupné z: http://www.biofeedback-rhb.cz/hra-mozku-a-co-na-to-rika-samotny-mozek/.

[6] Kraft, V. Systém pro distribuci a správu kognitivních her. Master's thesis, University of West Bohemia, Czech Republic, 2018.

[7] Mouček, R. et al. Software and Hardware Infrastructure for Research in Electrophysiology. *Frontiers in neuroinformatics*. 03 2014, 8, s. 20. doi: 10.3389/fninf.2014.00020.

[8] Neurosky, I. *About NeuroSky* [online]. Silicon Valley, Kalifornie: Neurosky, Inc., 2019. [cit. 2019/27/12]. Dostupné z: http://neurosky.com/about-neurosky/.

[9] Neurosky, I. *MindWave Mobile 2: User Guide* [online]. Silicon Valley, Kalifornie: Neurosky, Inc., 2018. [cit. 2019/27/12]. Dostupné z: https://store.neurosky.com/products/pc-developer-tools.

[10] Neurosky, I. *NeuroSkyUnityThinkGearPlugins* [online]. Silicon Valley, Kalifornie: Neurosky, Inc., 2017. [cit. 2019/10/9]. Dostupné z: https://github.com/NeuroSkyWuxi/NeuroSkyUnityThinkGearPlugins.

[11] NTIS. *Our Laboratory* [online]. NTIS - New Technologies for Information Society, 2020. [cit. 2020/10/1]. Dostupné z: http://neuroinformatics.kiv.zcu.cz/texts/read/our-laboratory_2015-04-02.

[12] Pixels, H. *Awesome Charts and Graphs* [online]. Happy Pixels, 2020. [cit. 2020/29/3]. Dostupné z: https://assetstore.unity.com/packages/tools/gui/awesome-charts-and-graphs-138153.

[13] Pug, C. *Safe Area Helper* [online]. Crystal Pug, 2020. [cit. 2020/29/3]. Dostupné z: https://assetstore.unity.com/packages/tools/gui/safe-area-helper-130488.

[14] Shi, Y. *Dynamic Line Chart* [online]. Yun Shi, 2020. [cit. 2020/29/3]. Dostupné z: https://assetstore.unity.com/packages/tools/gui/dynamic-line-chart-108651.

[15] Univerzita, M. *Vyšetření evokovaných potenciálů* [online]. Institut biostatistiky a analýz Masarykovy univerzity, 2020. [cit. 2020/16/1]. Dostupné z: https://telemedicina.med.muni.cz/pdm/detska-neurologie/index.php?pg=neurologicke-vysetreni--vysetreni-evokovanych-potencialu.

[16] WitMotion ShenZhen Co., L. *About Wit* [online]. WitMotion ShenZhen Co., Ltd, 2017. [cit. 2019/27/12]. Dostupné z: http://www.wit-motion.com/english.php?m=text&a=index&classify_id=36500.

[17] WitMotion ShenZhen Co., L. *WT61C user manual* [online]. WitMotion ShenZhen Co., Ltd, 2019. [cit. 2019/28/12]. Dostupné z: http://wiki.wit-motion.com/english/doku.php?id=inclinometer.

[18] Zhonghao Han, N. G. B. Y. H. L. Y. W. L. H. A flexible motion tracking system based on inertial sensors. *MATEC Web of Conferences*. 08 2018, 198, s. 19. doi: 10.1051/matecconf/201819804010.

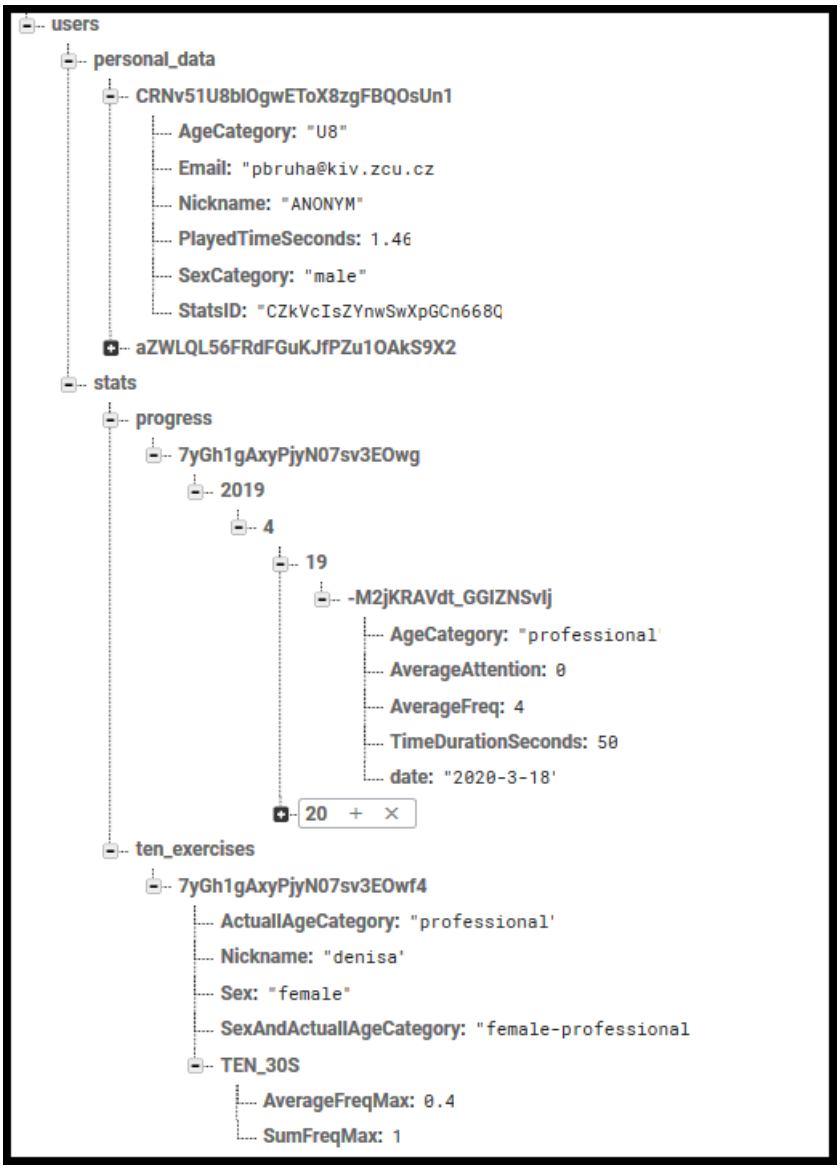
Seznam zkratek

ACI - Accelerometer-Computer Interface
AEP - Akustické (sluchové) evokované potenciály
BCI - Brain-Computer Interface
CSV - Comma-separated values
EEG - Elektroencefalografie
EKG - Elektrokardiografie
ERP - Event-Related Po-tentials
FPS - Frames per second
GUI - Graphic User Interface
IoT - Internet of Things
JSON - JavaScript Object Notation
KIV - Katedra informatiky a výpočetní techniky
MBKZ - Mobilní komunikace a zařízení
MEP - Motorické evokované potenciály
MVC - Model-View-Controller
SDK - Software development kit
SQL - Structured Query Language
VEP - Vizualní (zrakové) evokované potenciály

Přílohy

Příloha A - Návrh databáze

Obrázek 10.1: Příloha A - Návrh databáze - skupiny a typ účtu



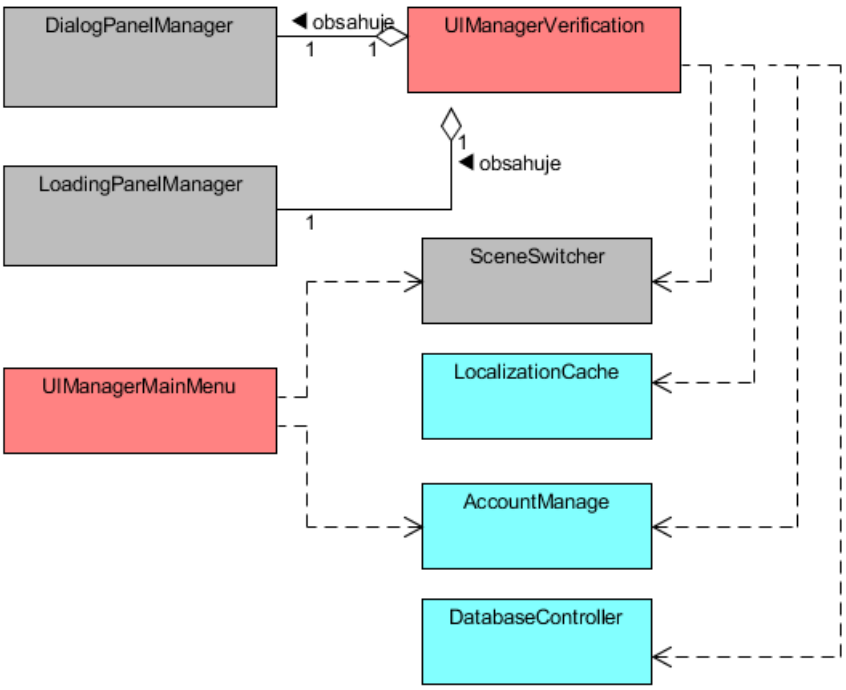
Obrázek 10.2: Příloha A - Návrh databáze - user

Příloha B - Zabezpečení databáze

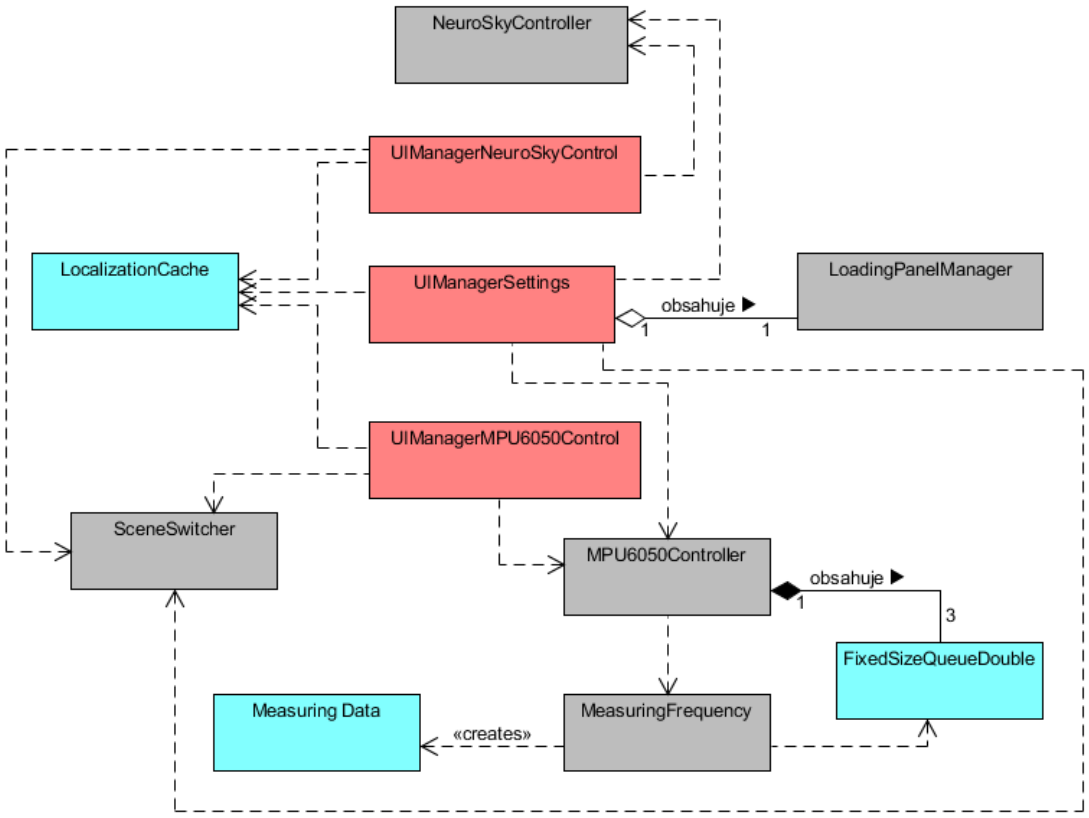
```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
{
  "rules": {
    "type_accounts": {
     "$uid": {
       ".read": "$uid === auth.uid",
       ".write": false
     }
    },
    "users": {
        "personal_data": {
          "$uid": {
            ".read": "$uid === auth.uid",
            ".write": "$uid === auth.uid"
          }
        },
        "stats": {
          "progress": {
            ".read": true,
            "$statsid":{
              ".read": true,
              ".write": "$statsid === root.child('users/personal_data/'+auth.uid+'/StatsID').val()"
            }
          },
          "ten_exercises": {
              ".read": true,
              "$statsid":{
                ".read": true,
                ".write": "$statsid === root.child('users/personal_data/'+auth.uid+'/StatsID').val()"
               }
            }
        }
    },
        "groups": {
        "personal_data": {
          "$uid": {
            ".read": "$uid === auth.uid",
            ".write": "$uid === auth.uid"
          }
        },
        "stats": {
          "ten_exercises": {
              ".read": true,
              "$statsid":{
                ".read": true,
                ".write": "$statsid === root.child('groups/personal_data/'+auth.uid+'/StatsID').val()"
               }
            }
        }
    }
  }
}
```

Obrázek 10.3: Příloha B - Pravidla zabezpečení databáze

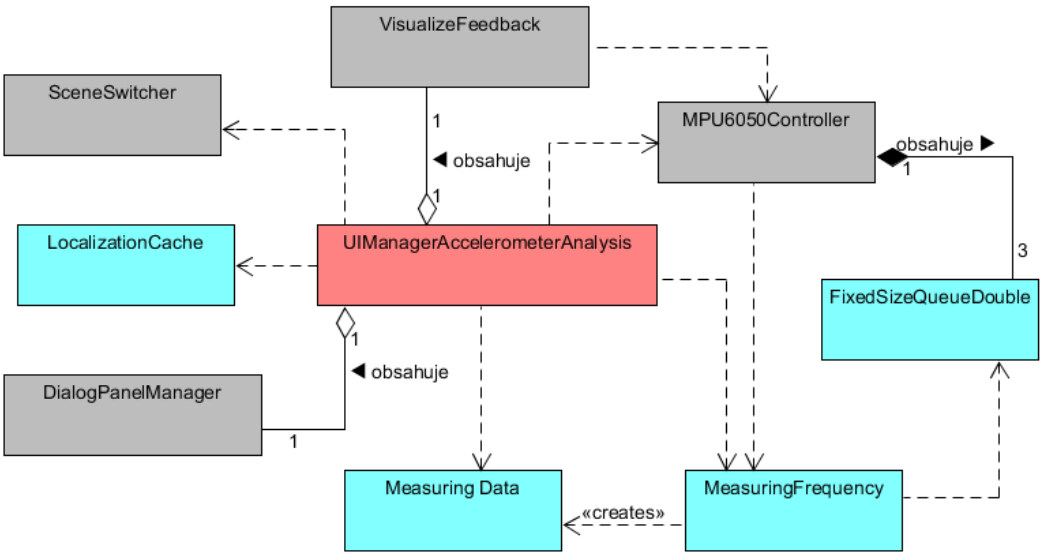
Příloha C - Diagramy tříd



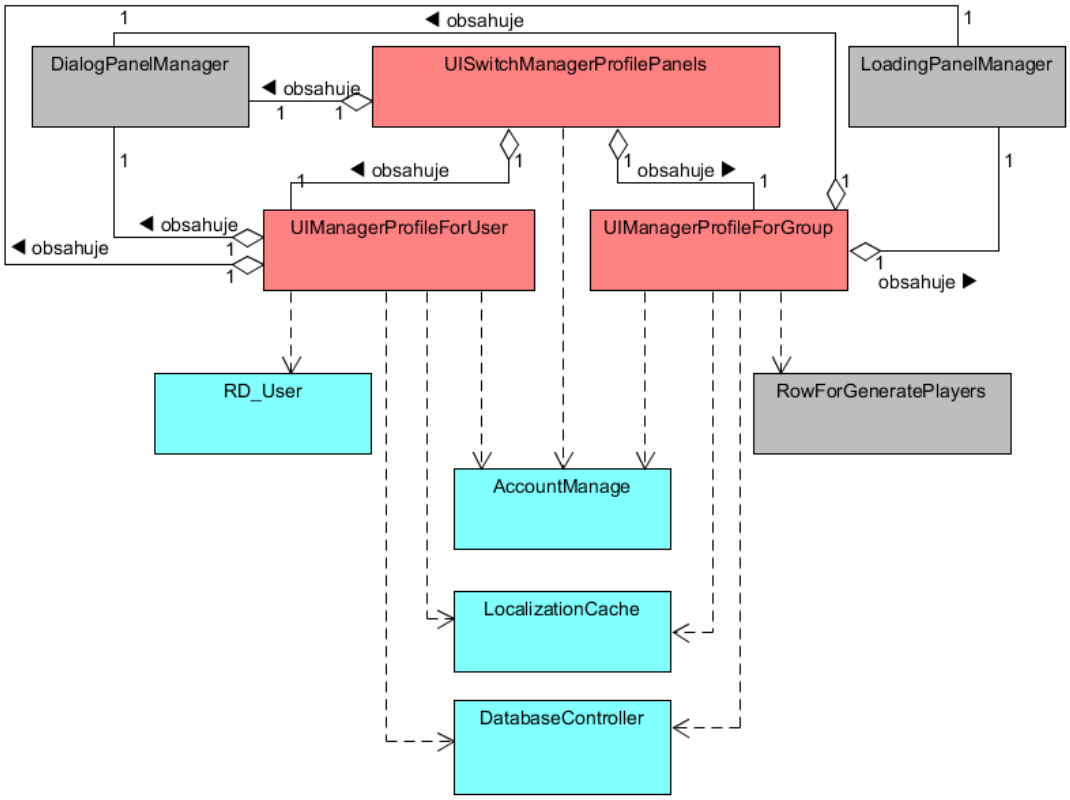
Obrázek 10.4: Příloha C - Diagram scény VerificationScreen a MainMenuScreen



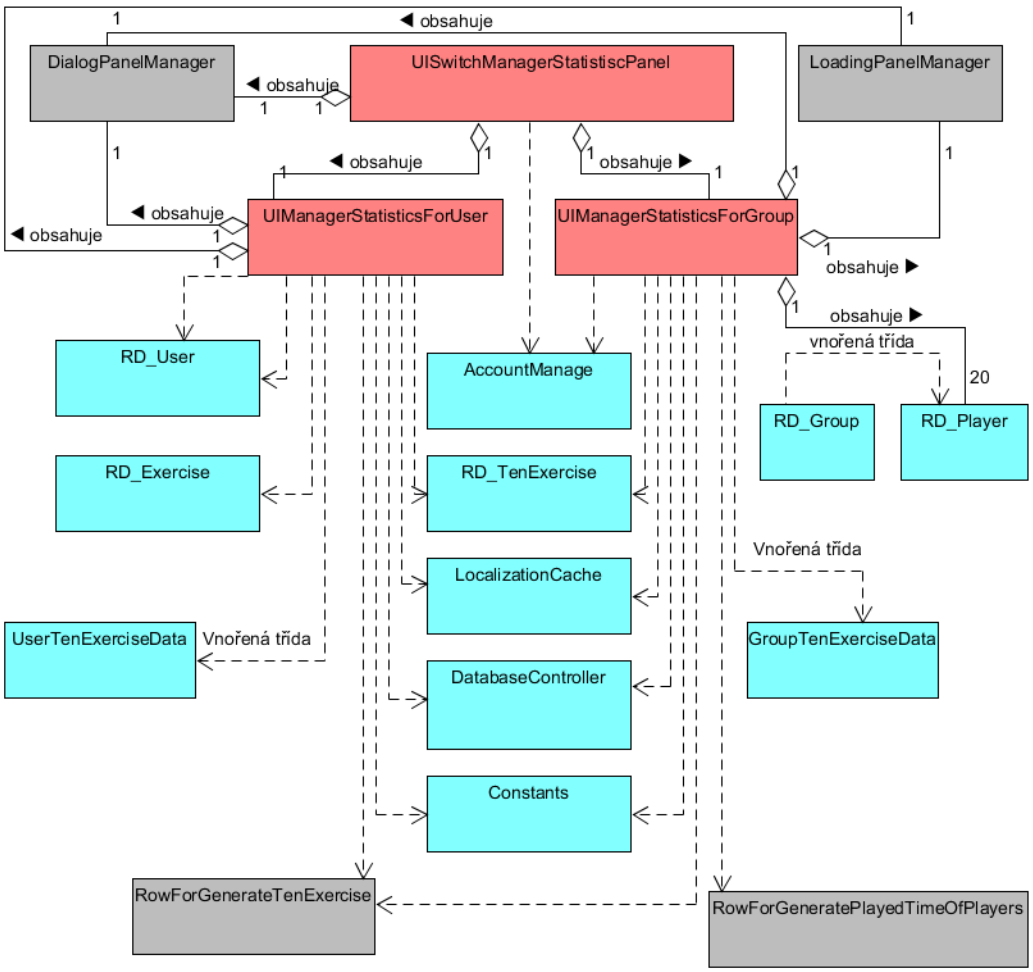
Obrázek 10.5: Příloha C - Diagram scény SettingsScreen a dalších dvou scén pro ladění senzorů



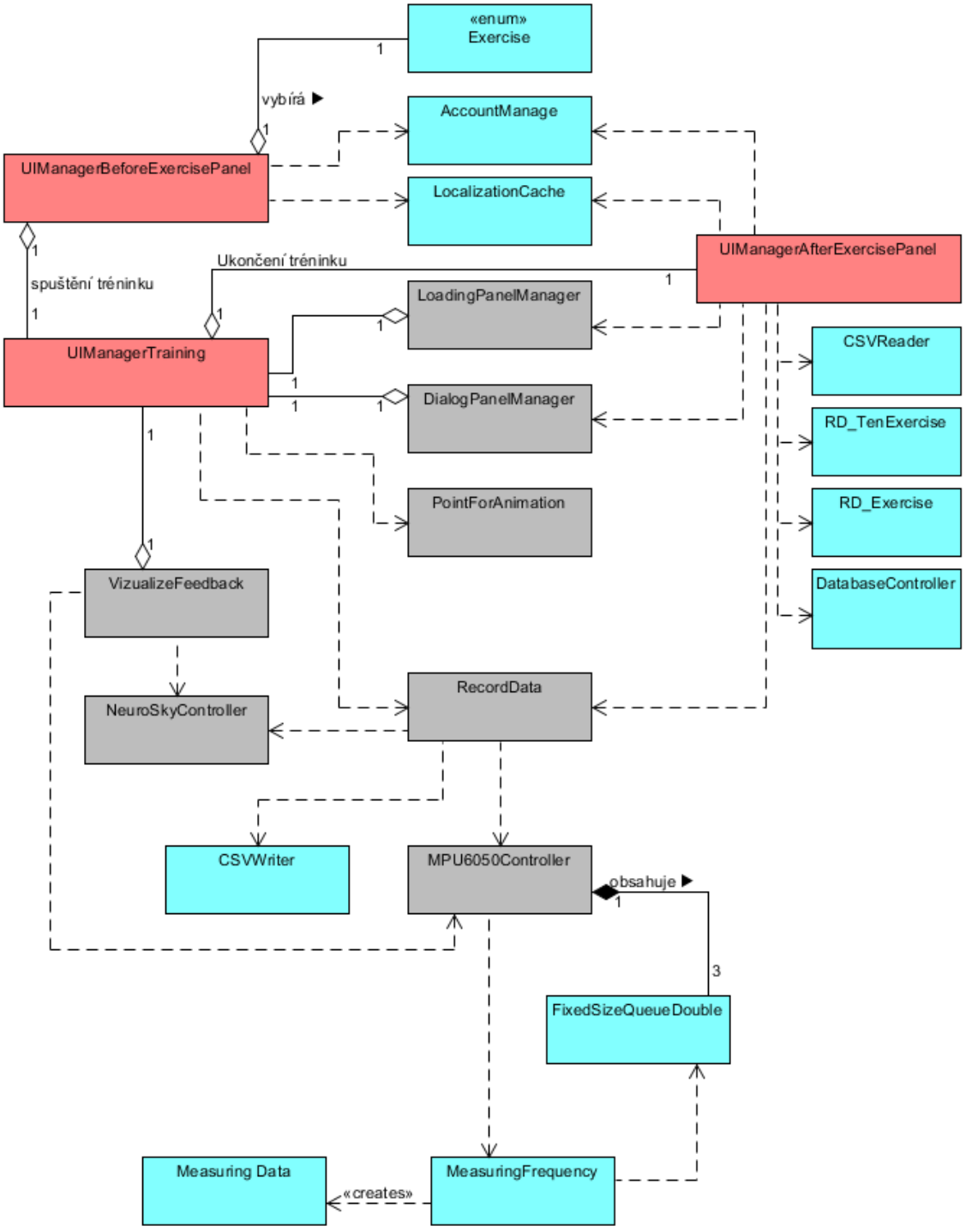
Obrázek 10.6: Příloha C - Diagram scény sloužící pro analýzu dat ze senzoru WT61C



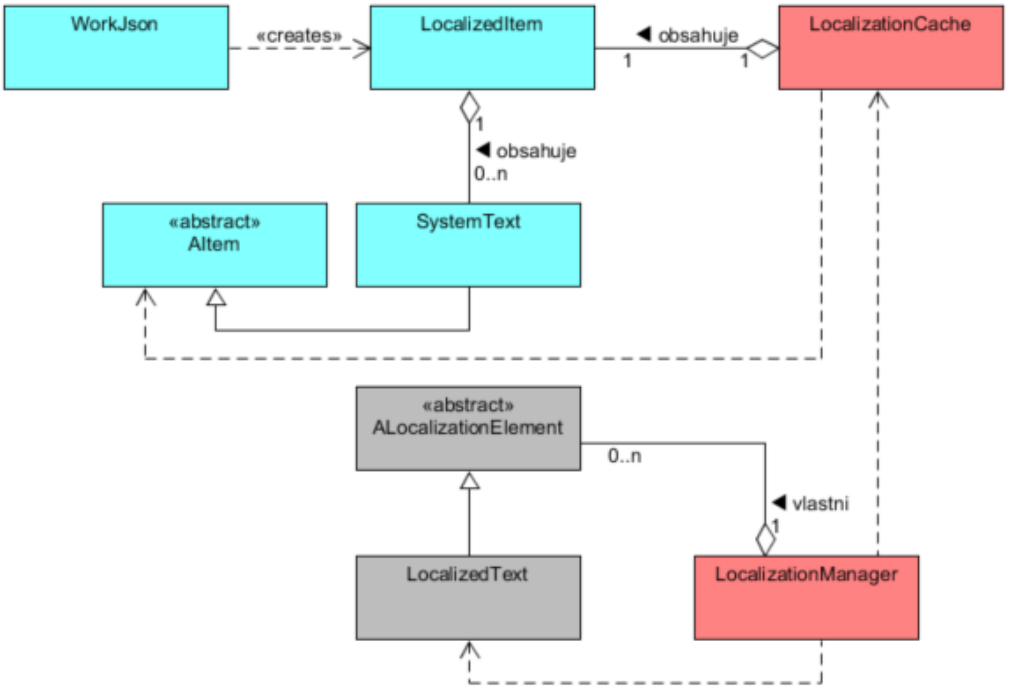
Obrázek 10.7: Příloha C - Diagram profilové scény se třemi UIManager scripty



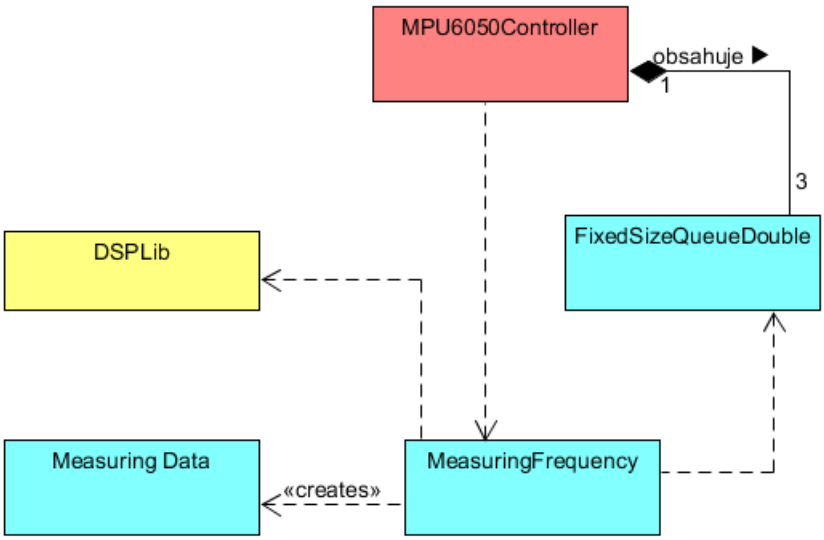
Obrázek 10.8: Příloha C - Diagram scény pro statistiky se třemi UIManager scripty



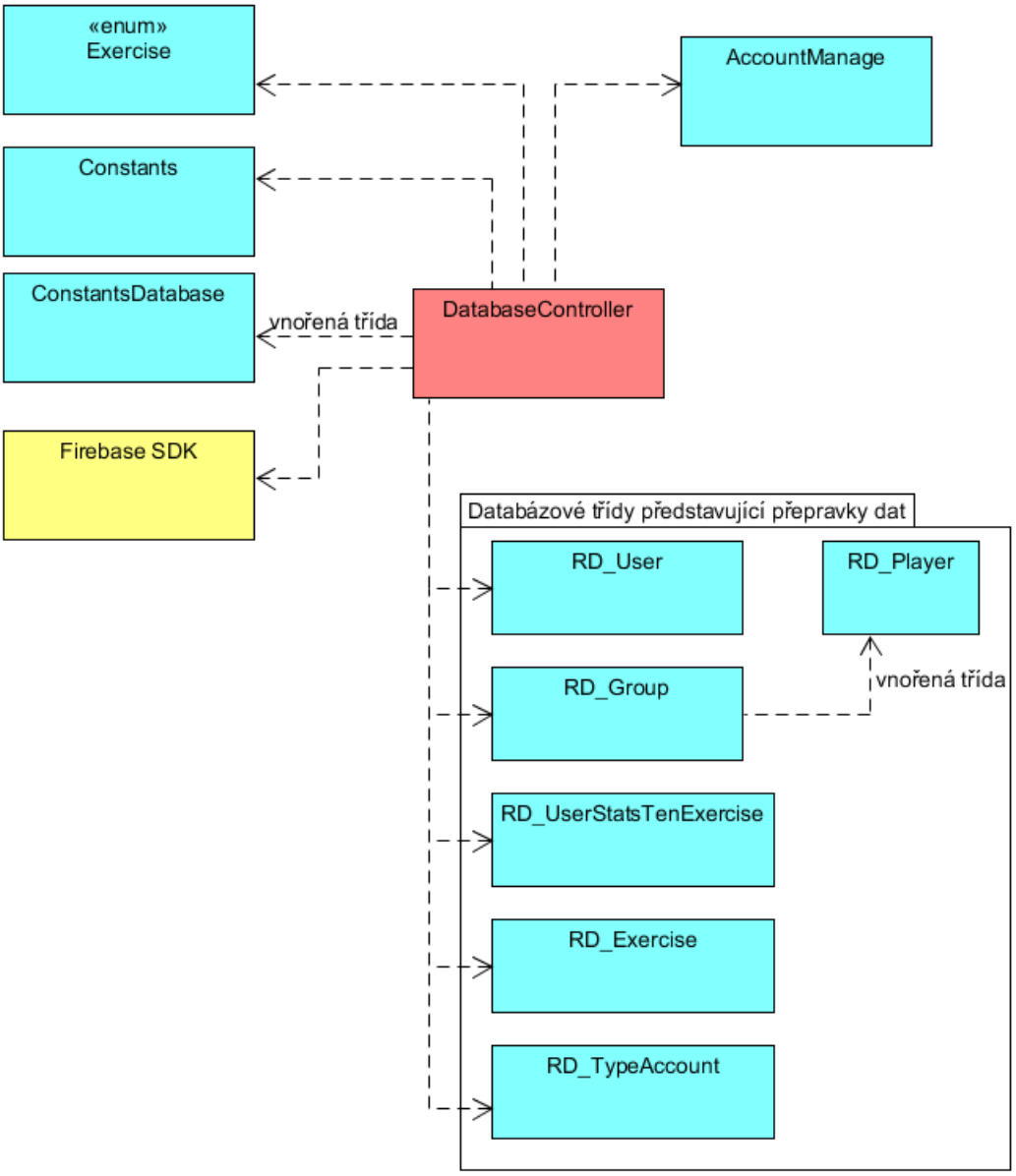
Obrázek 10.9: Příloha C - Diagram scény pro trénink se třemi UIManager scripty



Obrázek 10.10: Příloha C - Diagram představující lokalizační framework



Obrázek 10.11: Příloha C - Diagram tříd k měření frekvence



Obrázek 10.12: Příloha C - Diagram tříd pracující s `DatabaseController.cs`

Příloha D - Sestavení aplikace

Pro spuštění programu na platformě Windows je potřeba mít build aplikace. K **sestavení aplikace je vytvořen `built.bat`** soubor, který je potřeba modifikovat podle vašeho prostředí. Než dojdeme k samotnému sestavení je potřeba předtím splňovat tyto požadavky:

- **Mít k dispozici složku s projektem tohoto programu.**
- **Nainstalován Unity3D editor 2019.2.15f1** - editor je potřeba k sestavení aplikace. V této verzi editoru byla aplikace vyvíjena, a proto je doporučena.
- **64bitová architektura systému** - výstupem bude aplikace cílena především na 64bitovou architekturu systému.

Teď přejdeme k samotné modifikaci souboru **build.bat** pro přizpůsobení k vašemu prostředí:

1. Otevřete si zmiňovaný soubor v textovém editoru.
2. Upravte první řádek - zadejte správnou cestu k nainstalovanému Unity3D editoru. Níže můžete vidět příklad:

```
1 set path_editor_19_215f1=D:\Unity3D\Editors\2019.2.15f1\
     Editor\Unity.exe
2
```

3. Upravte druhý řádek - zadejte správnou cestu k projektu aplikace. Níže můžete vidět příklad:

```
1 set project_path=D:\Unity3D\Projects\
     HockeyCoach_Unity2019.2.15f1\HockeyCoach
2
```

Dále stačí soubor spustit přes příkazový řádek a čekat dokud nedojde k sestavení a puštění aplikace. Sestavená aplikace s názvem **`build_DATUM.exe`** se bude nacházet ve složce **`C:\output_hockey_coach`**.

Příloha E - Systémové požadavky aplikace Hockey Coach

Aby bylo možné aplikaci Hockey Coach spustit a používat, počítač musí splňovat níže uvedené minimální technické specifikace:

1. Operační systém - Windows 8.1 / Windows 10.
2. Grafická karta - 2GB grafická karta.
3. CPU - 64bitový procesor (x86-64).
4. RAM - 4GB.
5. Místo na pevném disku - 100MB.
6. Rozlišení obrazovky - monitor s rozlišením 1024 x 768 (doporučuje se rozlišení 1920 x 1080).
7. Internet - ověření identity - přístup do aplikace.

Příloha F - Uživatelský manuál

Uživatelská příručka, která Vám poskytne základní informace o používání produktu. Pro přihlášení do aplikace je nutné mít přihlašovací údaje, které vyplníte v přihlašovací obrazovce, viz níže.

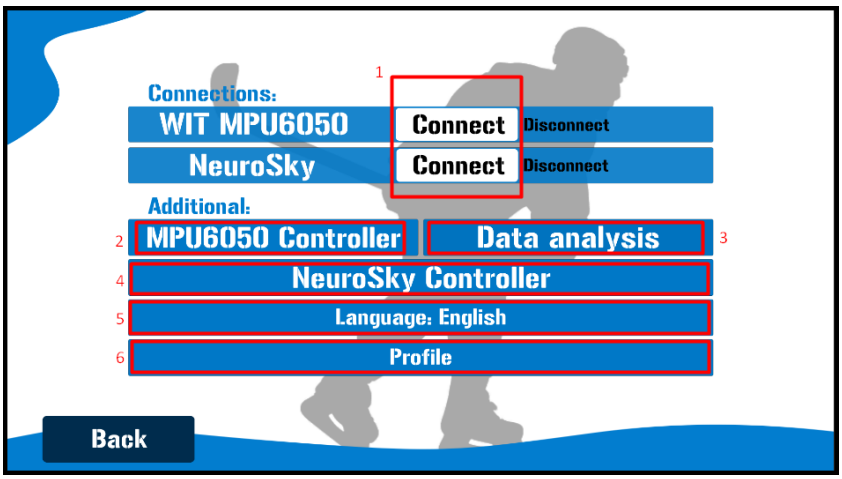
Obrázek 10.13: Příloha F - Přihlašovací obrazovka

Po přihlášení můžete vidět hlavní menu aplikace, ze kterého se můžete dostat do nastavení, statistik, sekce trénování a také do sekce profilu, kde naleznete své uživatelské údaje. Menu obrazovka je na obrázku níže.

Obrázek 10.14: Příloha F - Hlavní menu

Příloha F - Sekce nastavení

Aplikace má široké nastavení a to už jen tím, že lze k programu připojit dvě IoT zařízení, přesněji senzor WT61C (pro měření frekvence a akcelerace puku) a čelenku EEG MindWave Mobile 2 (pro měření soustředěnosti). Níže můžete vidět sekci nastavení a pod obrázkem popis obrazovky.



Obrázek 10.15: Příloha F - Sekce nastavení aplikace

Zvýrazněné bloky na obrazovce znamenají:

1. Tlačítka pro rychlé připojení senzorů s nastavením předchozího připojení.
2. Tlačítko, které vás dostane do nastavení senzoru WT61C.
3. Tlačítko, které vás dostane do sekce analýzy dat senzoru WT61C (uživatel může vidět přijímaná data v grafu).
4. Tlačítko, které vás dostane do nastavení čelenky EEG MindWave Mobile 2.
5. Tlačítko pro změnu jazyka v celé aplikaci.
6. Tlačítko, které vás dostane do profilové sekce.

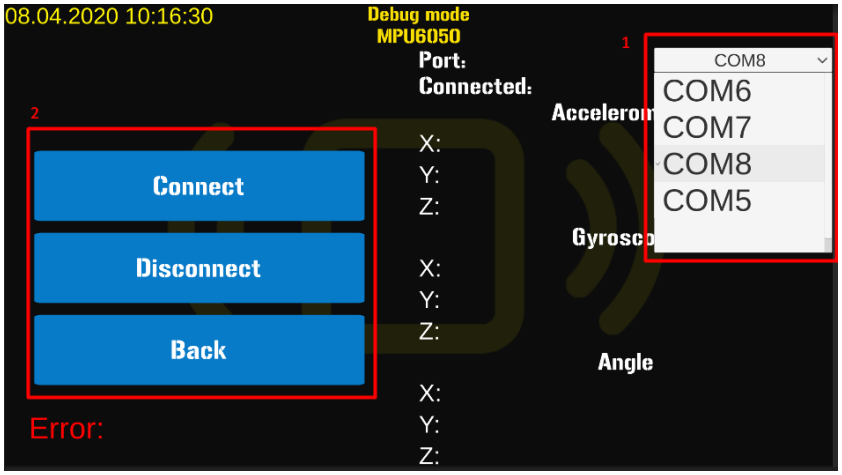
Příloha F - Sekce nastavení připojení WT61C senzoru

Níže můžete vidět obrazovku pro nastavení senzoru, který je potřeba **připojit přes Bluetooth k vašemu zařízení**. Do této obrazovky se dostanete přes sekci **Nastavení a tlačítko `MPU6050 controller`**.

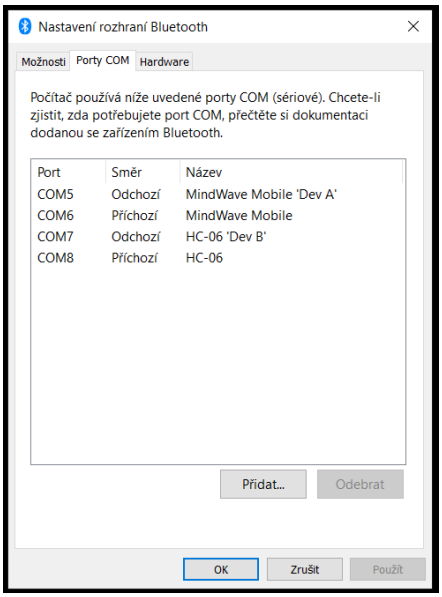
Před připojením senzoru k aplikaci, je nutné senzor sprárovat a připojit přes Bluetooth k počítači. Název senzoru je **HC-06** a heslo k připojení senzoru je **1234**.
Bloky na obrzovce znamenají:

1. Volba portu připojeného senzoru. Pro zjištění portu je nutné si otevřít **Nastavení rozhraní Bluetooth**, viz obrázek 10.17 (systém Windows). V tomto případě vás zajímá název **HC-06 'Dev B' a jeho port.**

2. Tlačítka pro obsluhu senzoru, ale také pro vrácení se zpět do nastavení.



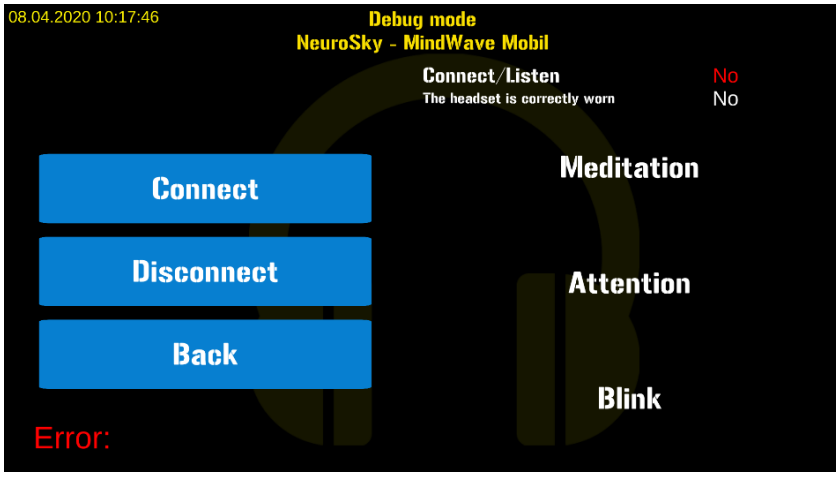
Obrázek 10.16: Příloha F - Nastavení připojení senzoru WT61C



Obrázek 10.17: Příloha F - Nastavení rozhraní Bluetooth

Příloha F - Sekce nastavení připojení EEG MindWave Mobile 2

K připojení čelenky je nutné mít na počítačí **nainstalovaný program** `ThinkGear Connector`, který lze stáhnout z oficiálních stránek výrobce. Následně je nutné připojit čelenku přes Bluetooth k vašemu zařízení a ujistit se, že se `ThinkGear Connector` k čelence připojil. Pak už jen stačí se k čelence připojit pomocí aplikace a to buď v nastavení anebo v nastavení tohoto senzoru.



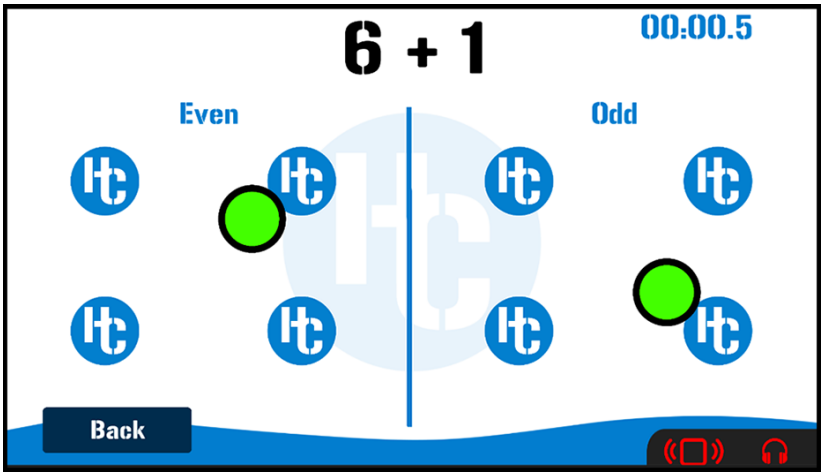
Obrázek 10.18: Příloha F - Nastavení připojení čelenky EEG MindWave Mobile 2

Příloha F - Sekce trénování

V sekci trénování je 10 cvičení. Je zde také kontrolní panel v pravém dolním rohu, který informuje o připojení senzorů (zelená barva - připojen, červená barva - odpojen). **Existují dva typy cvičení. První typ se vyskytuje v 1. - 9.cvičení. Druhý typ se vyskytuje jen v 10. cvičení.**

Po zvolení 1. - 9. cvičení se objeví panel, kde lze zvolit rychlost puku a také zvolit časový interval, po kterém se změní problém v horní části obrazovky. Typ účtu GROUP má navíc volbu ID hráče, který jde zrovna trénovat.

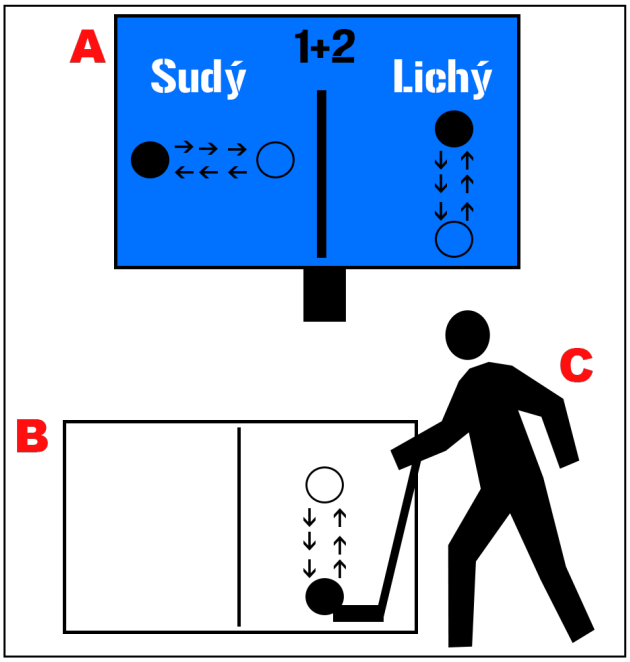
Obrázek 10.19: Příloha F - Panel před cvičením účtu typu GROUP



Obrázek 10.20: Příloha F - Tréninkový panel s animacemi

Princip trénování s 1. - 9. cvičením je popsán zde:

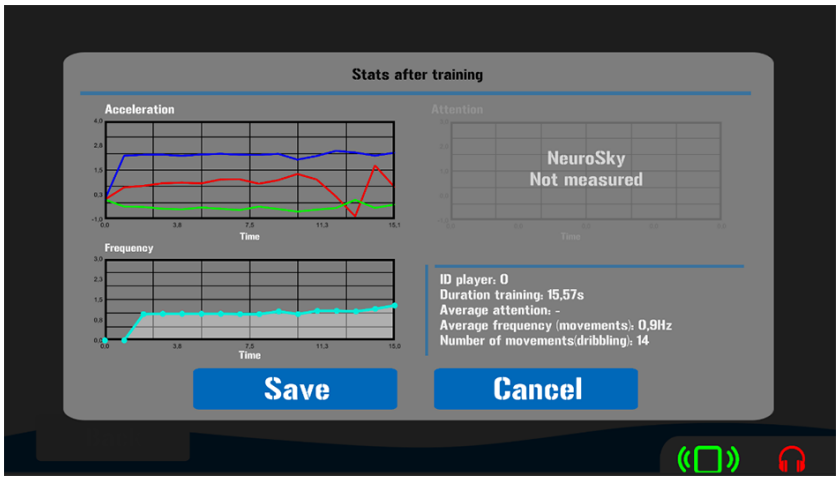
Objekt na obrázku níže označený červeným písmenem **A** je obrazovka, na které běží aplikace. Podle výsledku matematického příkladu v horní části obrazovky se odvíjí pohyb (směr posunu puku), který hráč **C** bude vykonávat na pomyslně rozdělené ploše **B** před obrazovkou. Matematické příklady se budou generovat náhodně po několika sekundách. V případě obrázku by hráč měl posouvat puk shora dolů a zpět na pravé části pomyslné plochy.



Obrázek 10.21: Princip tréninku s 1. - 9. cvičení

10. cvičení je vyjímečné v tom, že je zaměřené především na rychlost frekvence pohybů s pukem. Aby cvičení splňovalo účel, **je nutné mít k aplikaci připojen senzor WT61C**, který bude trenujícímu jedinci měřit frekvenci. Před startem tohoto cvičení si hráč zvolí, jak dlouhý chce mít trénink (30s, 60s, 90s, 120s) a následně pro start tréninku klikne na tlačítko start

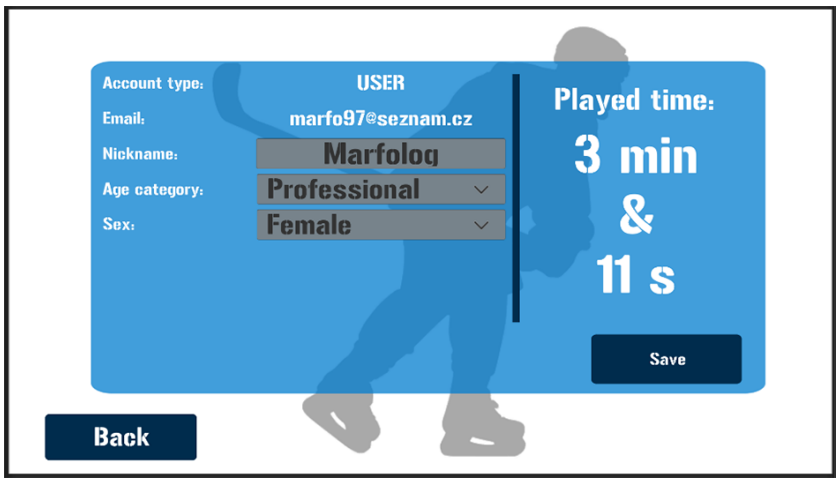
Po skončení tréninku (kliknutí na tlačítko zpět) se vám objeví panel se statistikami proběhnutého tréninku (cvičení). Je zde také volba pro uložení těchto dat do celkových statistik, viz obrázek na další straně.



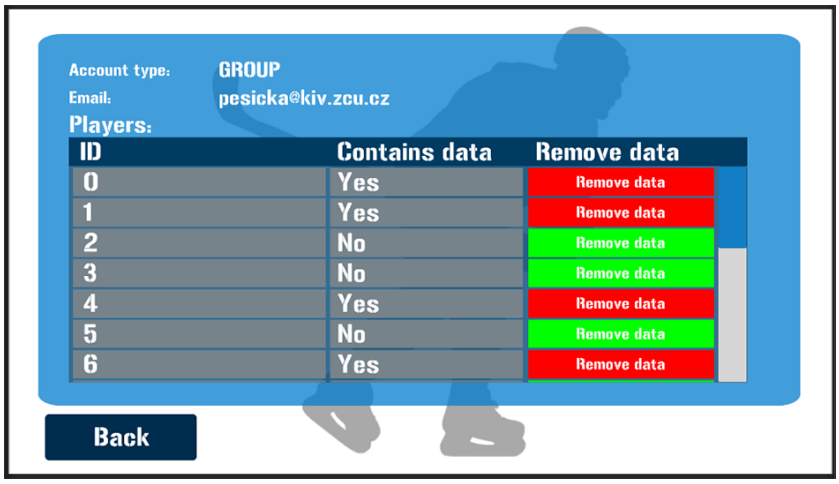
Obrázek 10.22: Příloha F - Panel se statistikami

Příloha F - Profilová sekce

Profilová sekce se liší typem účtu. Uživatel **typ účtu USER** zde může vyplnit svoji přezdívku, věkovou kategorii a pohlaví. Účelem vyplnění těchto dat je srovnávání hráčů mezi sebou v celkových statistikách. Uživatel **typu účtu GROUP** zde vidí jen tabulku všech možných hráčů a jejich stavy. V tabulce je také možnost odstranit data hráče (když bychom identifikátor chtěli přiřadit novému hráči). Identifikátor hráče slouží pro možné trénování a uložení dat pod tímto identifikátorem pro možné zobrazení statistik.



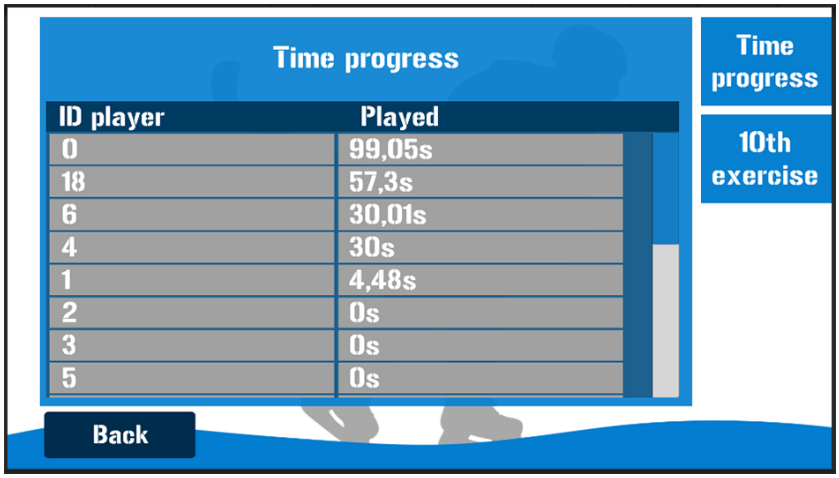
Obrázek 10.23: Příloha F - Profil typu účtu USER



Obrázek 10.24: Příloha F - Profil typu účtu GROUP

Příloha F - Statistiky

Sekce statistiky se také liší typem účtu. **Typ účtu GROUP** zde má 2 kategorie. V první kategorii statistik může uživatel vidět všechny hráče a jejich odehrané časy. V druhé kategorii je 10. cvičení, kde naleznete tabulku. V tabulce jsou porovnávány hráči podle součtu pohybů (frekvencí) za celou dobu tréninku dle typu (30s, 60s, 90s, 120s).



Obrázek 10.25: Příloha F - Statistiky účtu typu GROUP

Uživatel **účtu typu USER** může vidět, kolik má časově odehráno za určité období a také jakou průměrnou frekvenci v tomto časovém intervalu měl. Dále zde může vidět statistiky 10. cvičení, kde lze vidět své umístění v porovnání s ostatními hráči v součtu pohybů (frekvencí) v jednotlivých typech (30s, 60s, 90s, 120s).