Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Medgyesi Zsolt, Pomázi Krisztián Dániel

HASZNÁLHATÓSÁG VIZSGÁLATA HORDOZHATÓ BIOFEEDBACK RENDSZER SEGÍTSÉGÉVEL

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2015

Tartalomjegyzék

Összefoglaló

A mobil és asztali alkalmazások használhatósági vizsgálata egyre hangsúlyosabb kutatási terület, ahogy növekszik az elektromos eszközök hétköznapi életben betöltött szerepe. Mind a felhasználói hűség, mind a használhatóság mértéke kulcsfontosságú szempont sikeres alkalmazás fejlesztéséhez, figyelembe véve a piac dinamikus növekedését. Egy megfizethető és hordozható monitorozó környezet asztali és mobil platformra kielégítheti a felhasználói élményt kutató tudósok igényeit. Ebben a dolgozatban bemutatunk egy rendszert, amely monitorozza a felhasználó fiziológiai jeleit bármilyen alkalmazás használata közben. Ennek eredményeképp képesek vagyunk a szemmozgás, az EKG és EEG jelek rögzítésére, és a szenzorok által kibocsátott adatok együttes elemzésére. Az eredmények megjelenítése oly módon valósul meg, hogy a felhasználó élményt kutatók könnyen tudják követni a felhasználó alkalmazáshasználatát és kognitív folyamatait. Ebben további segítséget nyújt az adatokat csoportosító és adaptívan osztályozó integrált algoritmus. Így lehetséges a felhasználói felület további fejlesztése a produktivitás növelése és a frusztráció csökkentése érdekében.

Abstract

Evaluating mobile and desktop applications from usability aspects is becoming trending as the usage of electronic devices increases every day. Both user loyalty and usage efficiency are key factors in developing successful applications, given the wide variety of the market. An affordable and portable monitoring environment for desktop and mobile platforms could fulfill the specific needs of user experience researchers. In this paper we present a system that monitors the physiological signals of the user while using any application installed on the devices. As a result, we are able track the eye movement, ECG and EEG signals, and analyze the combined data provided by these sensors. Results are visualized in a way that user experience experts can follow effortlessly application usage and cognitive processes of the user, which makes it possible to improve even further the user interface to boost productivity and user loyalty, and lower frustration.

1 Bevezetés

Az okostelefonok gyors terjedése miatt a szoftverfejlesztő cégek és startupok egyre nagyobb figyelmet fordítanak a mobil és multiplatform alkalmazásokra. Mindegyikük számára fontos lenne, hogy háttértudást szerezzenek a felhasználó szoftverhez fűződő viszonyáról és ezen keresztül az applikáció használhatóságáról is. A frusztráció a felhasználó elégedetlenségének egy fő forrása, és akkor jelenik meg, ha az elvárásai nem valósulnak meg a szoftver használatakor. Ez történik, valahányszor a cselekvéseik kevesebb vagy a várttól eltérő eredményeket hoznak. A megfelelően „kézreeső" szoftver akár a flow élményét is adhatja a felhasználónak. Az ilyen és hasonló érzelmi változások, a befektetendő mentális erőfeszítés mérhető bizonyos fiziológiai jelek vizsgálatával, mint például az EEG, EKG jel, vagy a szemmozgás követése.

A ma elérhető széles szoftveres kínálat mellett döntő lehet, hogy egy adott alkalmazást frusztráció-mentesen tudunk használni. Mind az idő, mind a pénz korlátozott rendelkezésre állása a használhatóság tesztelésének elmaradásához vezethet. Jelenleg a fiziológiai szenzorok többsége nehezen megfizethető, és csak nagyon kevés rendszer képes megfelelően megjeleníteni egyszerre több szenzor által nyújtott adatokat.

A célunk az volt, hogy egy megfizethető és hordozható, használhatóságot megfigyelő környezetet hozzunk létre mobil és asztali alkalmazásokra. A rendszer által kényelmesen tesztelhetők alkalmazások, alacsony költségű fiziológiai eszközök segítségével. Mivel korlátozott a külső hardverek kábeles csatlakoztatási lehetősége mobil eszközökhöz, kiemelt fontosságú, hogy a szenzorok vezeték nélküli módon kommunikáljanak.

Általában a különböző biofeedback eszközök által gyűjtött adatok a hozzájuk készült szoftverek egyedisége miatt csak önmagukban elemezhetőek könnyen, ami rendkívül sok időt igényel, és gyakran szinkronizációs problémákhoz vezethet a további szenzorok értékeivel való összevetés során. Az eredmények összehasonlítása további adatfeldolgozást és magas szintű szaktudást igényel. Ezt a problémát megoldandó, a különböző szenzorokat egy rendszerbe integráltuk. Ez által lehetséges a különböző eszközök által mért adatok elemzése további feldolgozás és nagymennyiségű munka nélkül. Az adatok egymás mellett, összefüggően vannak megjelenítve.

A platformok közötti megoldás megvalósítása érdekében egy web-alapú, online felületet alakítottunk ki a felhasználói élményt vizsgáló szakemberek számára. További előnye ennek a megvalósításnak, hogy az elemzés bármely távoli helyszínről lehetséges.

A dolgozat további részében elsőként a felhasznált technológiákat mutatjuk be, úgy, mint a fiziológiás eszközöket és a keretrendszert, illetve azt, hogy munkánk során hogyan használtuk fel ezeket. Ezt követik a tervezés lépései, majd a megvalósítás folyamatának érdekesebb, fontosabb részei. A következő fejezetben bemutatjuk a rendszer által elért eredményeket és a mérések tanulságait. Végül összefoglaljuk és áttekintjük az elvégzett munkát, és számba vesszük a további fejlesztési lehetőségeket.

2 A biofeedback alapú rendszerek háttere

2.1 A szoftvertesztelésről

A szoftver felhasználóbarátabbá és intuitívabbá tételére sok lehetőség áll a fejlesztők rendelkezésére. Már 1980-ben megjelentek az első felhasználói élményt kutató tanulmányok, azonban a felhasználók igényei és elégedettsége folyamatosan változik, és így egyre nagyobb szükség van a megfelelő felhasználói felület megalkotására [1].

A tesztelési módszerek többségét a fejlesztési fázisban használjuk, mint például a GOMS (Goals, Operations, Methods and Selection rules) [2], CONJOINT [3] és "Design Space" [4] módszereket [1]. A szoftver tesztelésének fő szempontja a használhatóság. A szoftvertesztelési metódusok két fő típusra oszthatjuk: analitikus és az empirikus módszerekre. A biofeedback az utóbbi kategóriába esik. Tehát a felhasználó egy prototípust tesztel, és a használat során követjük és rögzítjük a felhasználó fiziológiás jeleit [5]. A szenzorokból gyűjtött adatokat elemezve a szoftver módosítható a jobb felhasználói élmény és jobb fiziológiai eredmények elérésére.

2.2 Biofeedback eszközök

A BCI (Brain Computer Interface – Agy Számítógép Interfész) kutatók az agy és számítógépek közötti direkt kommunikációt vizsgálják [6]. Rendszerünkben ez a szenzorokon keresztül jelenik meg. A BCI a kognitív infokommunikáció kutatási terület része, mely a kapcsolatot vizsgálja az infokommunikáció és kognitív tudományok területe között, valamint a különböző mérnöki megoldásokat, melyek e két tudomány összefonódásával jöttek létre [7]. A kognitív alkalmazások célja, hogy az emberi agy lehetőségeit növeljük infokommunikációs eszközök segítségével, tekintet nélkül a földrajzi távolságra, bármilyen mesterséges vagy biológiai kognitív rendszerek kombinációjával [8][9].

Az ember-számítógép interakciót (Human-Computer Interaction, HCI) kutató tudósok folyamatosan dolgoznak olyan technikákon, melyek lehetővé teszik számunkra a felhasználó kognitív állapotainak mérését, úgy, mint például kognitív és memória terhelés, bevonódás, meglepettség, elégedettség vagy frusztráció [10]. Továbbá ismert, hogy az EKG (Elektrokardiogram) és EEG (Elektorenkefalográfia) jelek kifejeznek általános felindultságot és az agy éberségét [11]. Tehát ezen technikák használatával

képesek vagyunk a felhasználó agyának közvetlen monitorozására. A viselhető és adott esetben vezeték nélküli szenzorok használata a mérést hordozhatóbbá és időtakarékosabbá teszi. Az általunk választott EEG és EKG szenzorok emiatt viselhető eszközök.

2.3 Szenzorok és adatgyűjtés

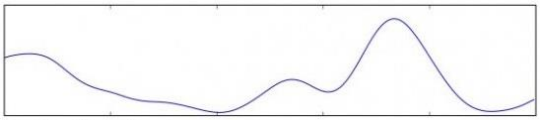
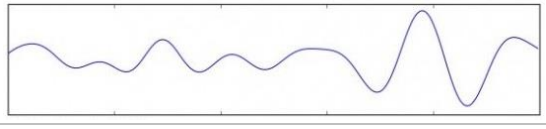
Az eszközök választásakor célunk az volt, hogy egy bárhol használható, egyszerűen beszerezhető és kedvező árú, könnyen hordozható mérési környezetet alkossunk meg, mely segítségével a lehető legtermészetesebb körülmények között tudjuk vizsgálni a felhasználó szokásait és tevékenységét. Az olcsóság és a könnyű hozzáférhetőség további előnye, hogy ezen rendszer és az eszközök birtokában bárki könnyedén végezhet méréseket, és értékes eredményeket kaphat pár tízezer forintnyi berendezés megvételével.

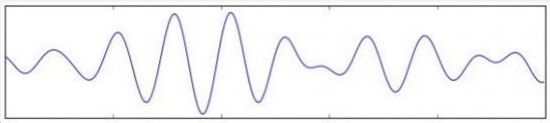
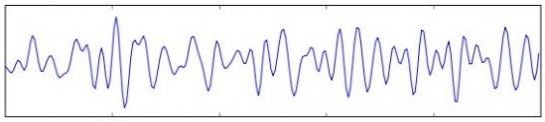
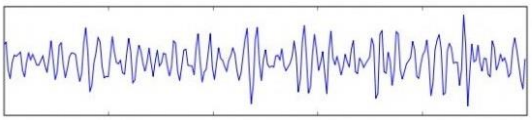
2.3.1 EEG eszköz

2.3.1.1 Elméleti háttér

Az elektroenkefalográfia (EEG) tágabb értelemben véve egy pszichofiziológiai mérőeljárás, melynek segítségével a pszichés működés élettani hátterét vizsgálhatjuk meg, szűkebb értelemben pedig egy elektrofiziológiai mérőeszköz, mely a neuronok elektromos aktivitásának regisztrálására szolgál valós időben. Az EEG-vel elvezethető jel az elektroenkefalogram, amely egy komplex, több komponensű periodikus görbeként írható le. Az EEG az agyi elektromos aktivitást méri, melynek alapja a központi idegrendszer elektorfiziológiás megközelítése. A különböző éberségi állapotoknak megfelelően az EEG hullámait kategóriákra osztjuk.

A hullámokat frekvencia szerint a következő kategóriákba soroljuk be [12]:

Név	Frekvencia	Spektrum
Delta hullámok	0,5-4 Hz	
Theta hullámok	4-8 Hz	

Alfa hullámok	8-13 Hz	
Béta hullámok	13-30 Hz	
Gamma hullámok	30-100 Hz	

A béta hullámoknál még szokás az alacsony béta (13-15 Hz), közepes béta (15-18 Hz) és a magas béta (18-30 Hz) megkülönböztetése. A leginformatívabb adatokat az alfa, béta és theta hullámok adják, melyek direkt információkat hordoznak a felhasználó teherbírásáról [13]. Az alfa hullámok részletes adatot nyújtanak a felhasználó kognitívteljesítményéről, és további szakértői elemzés tárgyai lehetnek [14]. A felhasználó figyelméről a béta hullámok adják a legpontosabb információt. A theta hullámok megbízhatóan mutatják a fáradtság növekedésével járó kognitív folyamatok során bekövetkező változásokat [15].

2.3.1.2 Az eszköz bemutatása

A Neurosky által gyártott Mindwave Mobile névre hallgató EEG készülék az egyik legolcsóbb és legkönnyebben elérhető, fejlesztési célokra használható EEG eszköz a piacon. Az eszköz a tanszéken elérhető, és a korábbi projektekben való használata miatt célszerű volt, hogy ebben a feladatban is ezt használjuk. Tervezésekor úgy alkották meg, hogy könnyen használható legyen mind okostelefon (iOS és Android), mind számítógép (Win és Mac) platformokra. Egyik legnagyobb előnye az olcsó hozzáférhetőség, mely kiemelten fontos szempont volt a projekt megvalósítása során.

Az eszköz visszaadja a nyers agyhullámokat, ezeket feldolgozza és kiadja a spektrumát 8 tartományra bontva (0-70 Hz intervallumban), valamint speciális, a Neurosky által fejlesztett algoritmus alapján számolja a figyelem (Attention) és a nyugodtság (Meditation) mértéket. Emellett az eszköz kimutatja, ha az alany pislog és erről is jelzést küld.

Az EEG a többi neuroimaging technikánál annyiban hatékonyabb, hogy a magas temporális felbontás következtében dinamikus leírást ad a folyamatokról, online monitorozást végez, és valós idejű visszacsatolásra ad lehetőséget [16] a figyelemre és

arousal-ra építve. Mint a legtöbbet alkalmazott BCI eszköz, a felhasználó mentális teherbírásáról ad információt miközben ő egy problémát old meg vagy feladatot hajt végre. A mentális teherbírás követésével állítani tudunk a végrehajtandó feladatok nehézségén, és így alkalmazkodni tudunk a felhasználó képességeihez [17]. Az általunk használt környezetben lehetséges a mentális terheltség követése a supervisor alkalmazáson keresztül, mely valós idejű visszacsatolást biztosít a fiziológiai jelekről a felügyelő számára.

1. ábra Mindwave Mobile EEG eszköz

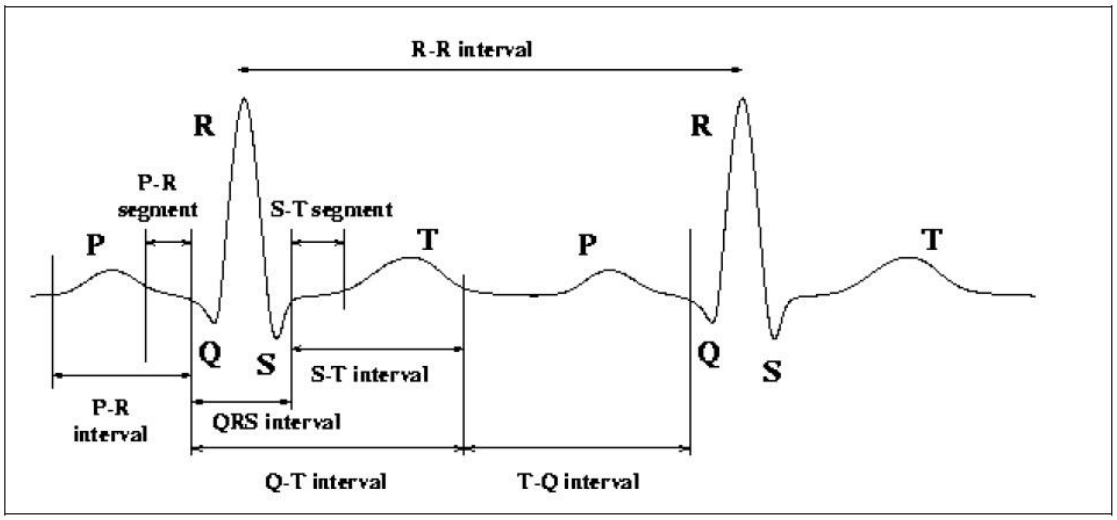
2.3.2 Zephyr HxM szívritmusmérő

2.3.2.1 Elméleti háttér

Az EKG hullámokat regisztráló szívritmusmérő eszköz működésének alapja az, hogy az emberi szív összehúzódása elektromos inger hatására jön létre, és ezeket az ingereket a testre helyezett elektródákkal érzékelni tudjuk.

Az általunk használt, EKG-ből kinyerhető adatok a pulzusszám, illetve a HPV (Heart Period Variance – Szívdobbanások között eltelt idő variabilitása), mely a HRV (Heart Rate Variance - Szívfrekvencia variabilitás) értékeiből származik. A HRV a szív neurovegetatív aktivitásának és autonóm funkcióinak mérésnagysága. Leírja a szív adottságát arra vonatkozóan, miként képes a belső és külső környezet megváltozott terheléseire reagálva a szívveréstől szívverésig eltelt időtartamot folyamatosan megváltoztatni (RR-távolságok), és ezzel kimutatja a szív alkalmazkodását az adott

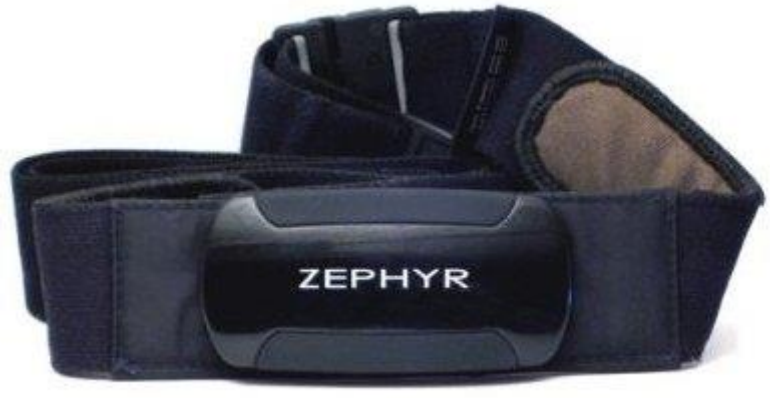
ingerekhez[18]. Ezért a HRV, és ezáltal a HPV értéke az egyszerű szívritmusnál jobban mutatja az alkalmazás által a felhasználóban keltett idegesség, feszültség változását.



2. ábra Két szívdobbanás között eltelt idő

2.3.2.2 Az eszköz bemutatása

A szívritmusmérő eszköz kiválasztásakor a Zephyr cég HxM Bt Heart Rate Monitor eszközét választottuk. Ennek több oka is volt, egyrészt ez állt könnyen rendelkezésre a tanszéken, másrészt mind Androidos, mind PC-s platformon könnyen használható és illeszthető az eszköz a Java SDK segítségével. A Zephyr HxM egy nyílt API-on keresztül Bluetooth-szal kommunikál, így könnyen összekapcsolható mindkét platformot használó eszközzel.



3. ábra Zephyr HxM Bt szívritmusmérő

2.3.3 EyeTribe szemmozgás-követő

2.3.3.1 Elméleti háttér

Egy okostelefon vagy számítógép használatakor nemcsak az alapján kaphatunk visszajelzést a felhasználóról, hogy hova kattint, vagy hol érinti meg a képernyőt, hanem értékes adatokat kaphatunk, ha tudjuk, mikor hova tekintett.

A szemmozgás-követő eszköz két fő komponense egy kamera és egy magas felbontású infravörös LED, mely okostelefonok esetében könnyedén integrálható a telefonba. Az eszköz a kamera segítségével érzékeli a felhasználó pupillájának legkisebb mozdulatát is, képet készít és a megfelelő algoritmusok segítségével átalakítja koordinátákká. Az EyeTribe használható különböző környezeti- és fényviszonyok között, de használata elsősorban beltérben javasolt [19].

Aktív használat során a szemmozgás segítségével tudjuk irányítani az alkalmazást, illetve programot, ezáltal helyettesítve számítógép esetén a kattintás vagy billentyűlenyomást, okostelefon esetében pedig az érintést.

Számunkra az eszköz passzív használata fontosabb, azaz nem az irányítás, hanem az eszköz által gyűjtött adatok tárolása és megjelenítése. A gyűjtött adatok segítségével hőtérkép rajzolható bármilyen képernyőre, mely rendkívül hasznos visszajelzés a felhasználó tevékenységéről a fejlesztők illetve oktatók számára.

2.3.3.2 Az eszköz bemutatása

Az eszköz választásakor fontos szempont volt az olcsóság, a méret és az elérhetőség. Az EyeTribe terméke megdöntötte a rekordot a világ legkisebb szemmozgáskövető eszközeként, és nem igényel külön tápellátást, USB 3.0 segítségével csatlakoztatható számítógéphez, tablethez, vagy okostelefonhoz.

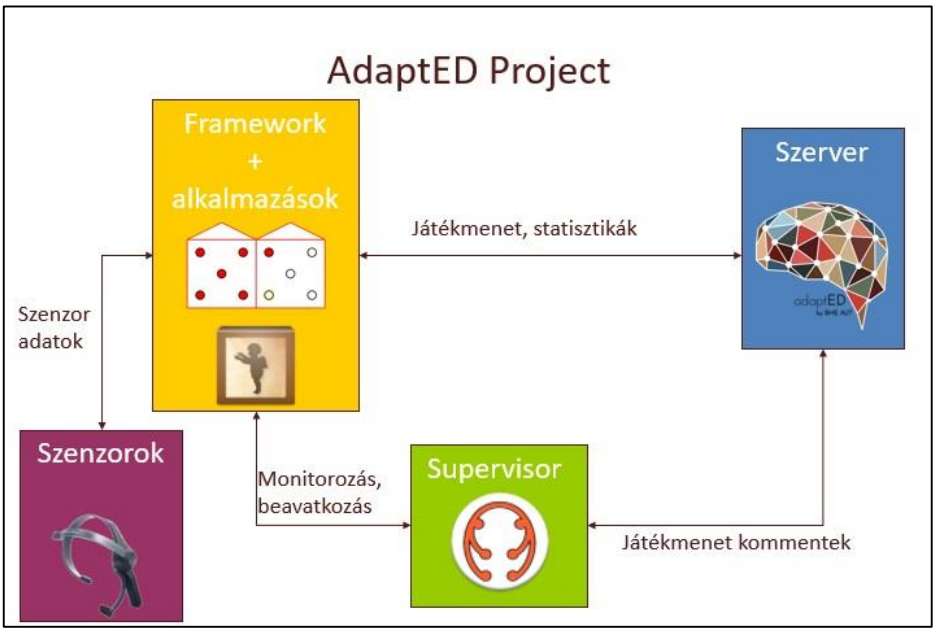
4. ábra Az EyeTribe működése

Az eszköz használatához szükség van annak kalibrálására, mely a gyártó által biztosított szoftver segítségével pillanatok alatt elvégezhető. A kalibrálás során és a szenzor használata közben célszerű fejünket a szenzorhoz képest ugyanabban a pozícióban tartani, mivel a kalibráció során ehhez a távolsághoz és szöghöz igazítva állítja be a szemünk érzékelését.

Mellékfunkcióként az EyeTribe eszköz biztosít adatokat a felhasználó pupillaméretéről, illetve, hogy a szem fixált vagy szakkád-mozgást végez-e épp. Az előbbi adat szintén felhasználható a kognitív aktivitás vizsgálatára [20].

2.4 A keretrendszer

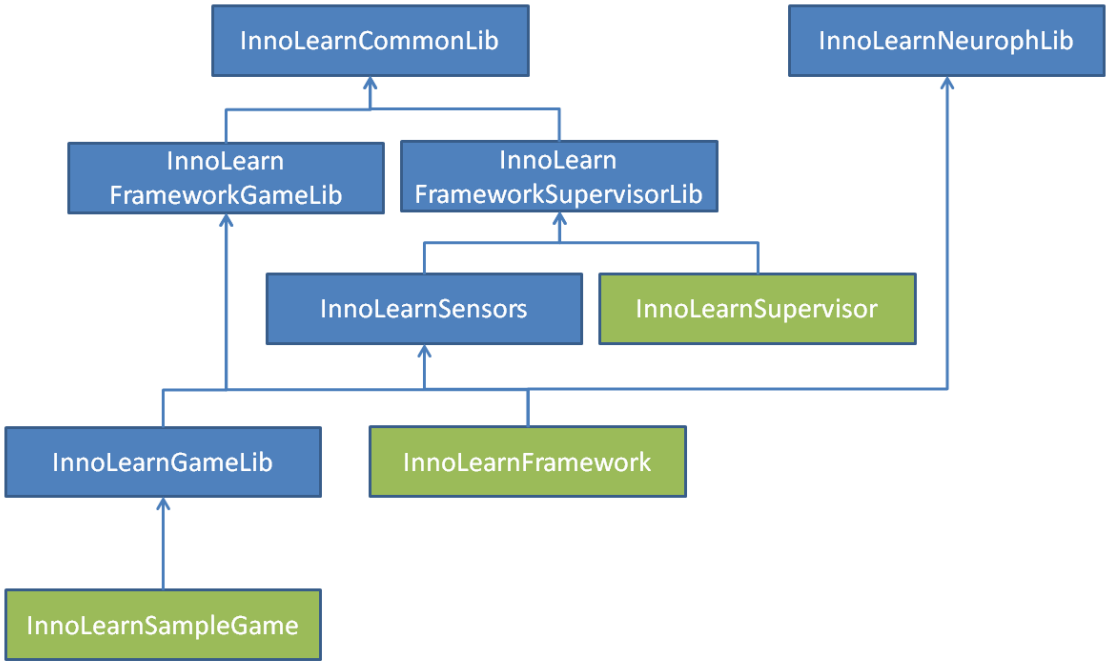
Megoldásunkat az AdaptEd keretszerre építettük [21]. Ez a rendszer Android projektek és játékok számára lett létrehozva biofeedback használata mellett. Kiterjesztettük a keretrendszert, oly módon, hogy képes legyen bármilyen mobil alkalmazás monitorozására, és Windows platformra is portoltuk. Megterveztük és megvalósítottuk a szemmozgáskövető funkciót. A szoftver rendszernek négy fő felülete van: (1) a biofeedback szenzorok, (2) a különböző játék vagy szoftver komponensek számára, (3) a felügyelő eszköz számára és (4) a szerver számára [21]. Az alkalmazás adatokat gyűjt a szenzoroktól és egy háttérfolyamat segítségével elküldi azokat a szervernek.



5. ábra A keretrendszer architektúrája: Az architektúra négy fő szerepre osztható: felügyelő, az szenzorok, a készüléken futó keretrendszer és alkalmazások, valamint a szerver. [21]

Az Android és PC megoldások ugyanazt a keretrendszert és adatbázis struktúrát használják az adatok feltöltésére és tárolására. Az adatbázis a szerver oldalon van implementálva. A mérési adatok regisztrálása és feltöltése a keretrendszerben események kiváltásán keresztül történik. Külön események kezelik a képernyőkép küldését, a HPV és EEG értékeket, valamint más szenzor inputokat. Az események összeköttetésben vannak az adatbázissal és felhasználónként tárolódnak. Minden esemény rendelkezik egy ID-val, típussal, gameplay ID-val, időbélyeggel, diszkriminátorral és egy opcionális komment mezővel.

A megoldásunk az AdaptEd keretrendszer felügyelő alkalmazás funkciójával is bővíti a használhatósági vizsgálat rendszerét [22]. Valós időben képes a felhasználó tevékenységét megfigyelni, és lehetővé teszi a felügyelő számára a mérés közben markerek elhelyezését, valamint a következtetett adatokat is bemutatja. Felület biztosít a keretrendszerhez kapcsolódó alkalmazások távirányítására.



6. ábra A keretrendszer felépítése - zölddel a futtatható programok [23]

Az ábrán szereplő *InnoLearnSampleGame* osztály a példajáték, azaz az alkalmazás, amit monitorozni szeretnénk, illetve e dolgozat esetében az a program, amelyből elindíthatjuk a készülékre telepített egyéb alkalmazások megfigyelését. Az *InnoLearnFramework* futtatható projekt valósítja meg a szerverrel való kommunikációt, és tölti fel az adatokat a háttérben. Az *InnoLearnSupervisor* projekt telepítésével külön

alkalmazáson keresztül felügyelhetjük a játékmenetet, illetve alkalmazáshasználatot. Itt megjelenek a különböző játékesemények és fiziológiás jelek, és a felügyelő a beállítások módosításával a játék közben is tudja módosítani a játék beállításait.

Az *InnoLearnGameLib* projecten keresztül a játék kapcsolódik a keretrendszerhez. Az *InnoLearnFrameworkSupervisorLib* a keretrendszer és a felügyelő alkalmazás által közösen használt osztályokat, az *InnoLearnFrameworkGameLib* pedig a játék és a keretrendszer által közösen használt osztályokat tartalmazza. Az *InnoLearnCommonLib* azokat az osztályokat tartalmazza, melyeket mindhárom futtatható alkalmazás használ. Az *InnoLearnSensors* projekten keresztül kapcsolódunk a fizikai szenzorokhoz, itt történik a jelfeldolgozás, és innen kapja meg a keretrendszer a fiziológiás adatokat. Az *InnoLearnNeurophLib* egy neurális hálót tartalmazó projekt, melyet a keretrendszer használ.

A keretrendszer több hallgató több éves munkájának eredménye, és folyamatosan alakul és fejlődik a fellépő igények és új funkciók szerint.

3 Tervezés

3.1 EEG eszköz illesztése

Az EEG eszköz illesztésekor a fő célunk az volt, hogy a gyártó által előállított *Attention (Figyelem)* és *Meditation (Nyugodtság)* értékekhez férjünk hozzá. Ezek adják számunkra a leghasznosabb információt a felhasználó tevékenységéről. A továbbiakban meghagyjuk a lehetőséget arra, hogy az eszköz nyers adatainak segítségével (különböző típusú hullámok) még részletesebb és specifikusabb információt nyerjünk.

Az adatok kinyerésén kívül fontos volt még számunkra a frekvencia megválasztása, azaz, hogy milyen sűrűn kerüljenek feltöltésre ezek az értékek. Elegendőnek találtuk a másodpercenkénti, azaz 1 Hz-es mintavételt.

3.2 HxM illesztése

A Zephyr HxM szívritmusmérő szenzorból szintén két értéket szerettünk volna kinyerni, és felküldeni a szerverre. Az egyik érték a Pulzusszám (Heart Rate), melyet az eszköz másodpercenként állapít meg, és számol ki percre vonatkoztatva az elmúlt szívverések alapján.

A másik érték a felhasznált technológiáknál is említett HPV (Heart Period Variance) érték, melyet a legutóbbi 15 szívverés timestamp-jéből számolhatunk. Erre a számolásra már létezett a rendszerben egy megoldás, mely FFT algoritmust használ, és Fekete András hallgató írta meg. Emellett az alkalmazásba került a Horváth Laura hallgató által implementált, Burg-algoritmus alapú megoldás is.

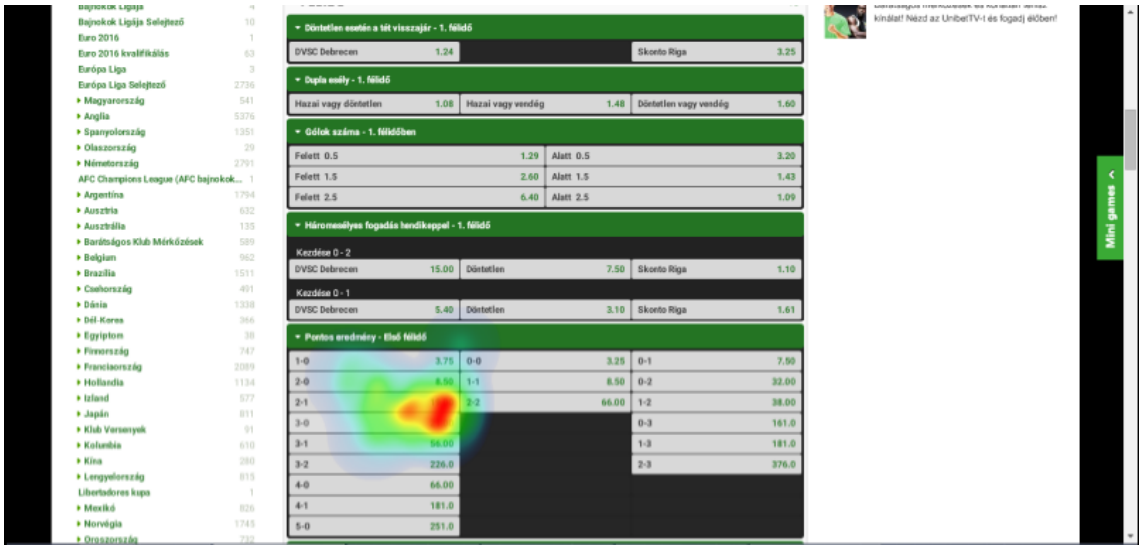
A HxM-hez az EEG-vel szemben nem állt rendelkezésre működő kód az adatok feltöltésére, így ezt is meg kellett valósítanunk. Ehhez fel kellett térképezni a keretrendszer működését, onnantól, hogy a szenzor miként fogadja és alakítja át az adatokat a szerverre való feltöltésig. Szerencsére mintaként rendelkezésre állt az EEG megvalósítása, mely már működött a tanszéki projekt keretében, így a megvalósításban fejezetben leírt funkciók megírásakor azt vettük alapul.

Byte/Bit	7	6	5	4	3	2	1	0	Field
0	STX								STX
1	0x26								Msg ID
2	55								DLC
3	Firmware ID								Payload
5	Firmware Version								
7	Hardware ID								
9	Hardware Version								
11	Battery Charge Indicator								
12	Heart Rate								
13	Heart Beat Number								
14	Heart Beat Timestamp #1 (Newest)								
16	Heart Beat Timestamp #2								
18	Heart Beat Timestamp #3								
20	Heart Beat Timestamp #4								
22	Heart Beat Timestamp #5								
24	Heart Beat Timestamp #6								
26	Heart Beat Timestamp #7								
28	Heart Beat Timestamp #8								
30	Heart Beat Timestamp #9								
32	Heart Beat Timestamp #10								
34	Heart Beat Timestamp #11								
36	Heart Beat Timestamp #12								
38	Heart Beat Timestamp #13								
40	Heart Beat Timestamp #14								
42	Heart Beat Timestamp #15 (Oldest)								
44	Reserved								
46	Reserved								
48	Reserved								
50	Distance								
52	Instantaneous speed								
54	Strides								
55	Reserved								
56	Reserved								
58	CRC								CRC
59	ETX								ETX

7. ábra A HxM által küldött csomag adatstruktúrájának felépítése [24]

3.3 EyeTribe illesztése

Az EyeTribe által gyártott eszköz illesztésekor a cél a következő volt: a szemmozgás-követő segítségével az adatok koordinátáit összegyűjtve hőtérképet készítsünk az adott idő alatt lezajlott szemmozgásokról. Így meg tudjuk jeleníteni azt, hogy a felhasználó az eltelt idő alatt a képernyő melyik területére fókuszált jobban, mi keltette fel a figyelmét. Ez által pontosan be lehet azonosítani azt a funkciót, amely a következtetett mentális állapotot, erőfeszítést igényelte. Emellett ebből kiszűrhetőek az adott alkalmazás ergonómiai, használhatósági hibái is. A hőtérképet az adott screenshotra illesztjük, és opcionálisan az érintési eseményeket is megjelenítve egy „overlay”-ként feltöltjük a szerverre.



8. ábra Hőtérkép az online adatbázisban

A monitorozáshoz és a szemmozgáskövetéshez fontos, hogy az Android eszköz vagy PC, és az EyeTribe egymáshoz képest fix pozícióban legyen, és hozzájuk viszonyítva a folyamat alatt a fejünket se mozgassuk el túlságosan a kalibrációt követően.

3.4 Monitorozó alkalmazás tervezése Android platformra

A monitorozó alkalmazás tervezésekor a cél az volt, hogy a keretrendszert kihasználva a meglévő szenzorokkal olyan alkalmazások használata közben is tudjunk adatokat szolgáltatni a felhasználóról, mely alkalmazások nem illeszkednek a keretrendszerhez. Tehát bármilyen, a telefonra telepített alkalmazás *(Activity)* használatát tudjuk monitorozni.

A tervezéskor mindenképpen fel szerettük volna használni a keretrendszer meglévő szolgáltatásait, hogy a munkánk az új területekre és a hozzáadott új funkciókra, értékekre koncentráljon. Így a keretrendszeren keresztül valósul meg a szenzorok adatgyűjtése, adatfeldolgozása és a szerverrel való kommunikáció. A monitorozó alkalmazás feladata pedig, hogy kapcsolatot teremtsen a keretrendszer és az éppen futó alkalmazás között, és az alkalmazás adatait a keretrendszer számára elérhetővé és feldolgozhatóvá tegye. Így tehát az EEG és HxM szenzor esetében csupán el kell indítanunk a monitorozást, és a keretrendszer elvégzi az adatok begyűjtését és feldolgozását, amennyiben az eszközök csatlakoztatva vannak.

Az alkalmazás tervezésekor a fő megvalósítandó funkció így az volt, hogy egy háttérben futó applikációból tudjunk a jelenleg futó activity-ről screenshotot készíteni, és

erre a képernyőképre overlay-ként elhelyezni az érintési eseményeket, illetve a szemmozgáskövető által gyűjtött adatokat.

3.5 Monitorozó alkalmazás tervezése Windows platformra

A Windows platformú monitorozó tervezésénél, kereszt-kompatibilitási szempontokból, fontos szempont volt az Android platformon futó AdaptEd keretrendszer minél kevesebb módosítással történő portolása, vagy szellemiségében hasonló megoldások használata, valamint az ott használt mérőeszközök alkalmazása.

Az korán nyilvánvalóvá vált, hogy az Android környezetben meglévő és működő szenzorillesztők desktop környezetben nem alkalmazhatóak, ezért itt szükség volt a szenzorok adatküldési protokolljának mélyebb megismerése. Az Android verzióban említett eszközök adatain kívül itt is szükség volt képernyőképek periodikus készítésére, valamint az egér pozícióinak és akcióinak rögzítésére, a keretrendszer háttérben futtatása esetén is. Ily módon biztosítva, hogy a keretrendszer és a megfigyelt program futása egymástól teljesen szeparált, lehetővé téve bármely alkalmazás vizsgálatát.

3.6 Az adatok struktúrája

Az adatokat a következőképpen terveztük megjeleníteni:

a) EEG
   - A mért jelek power spektruma
   - Származtatott értékek:
     i. Attention (Figyelem)
     ii. Meditation (Nyugodtság)

b) HxM
   - Pulzusszám
   - HPV érték

c) EyeTribe
   - Hőtérkép a koordinátákból számítva a képernyőképre kerülő overlay formájában
   - Pupillaméret

Az EEG és a szívritmusmérő adatait, mivel számértékek, a keretrendszerben eseményként kezelve tudjuk felküldeni. Azaz minden alkalommal, amikor a szenzor új értéket küld a keretrendszer felé, generálunk egy eseményt, amelyet feltöltünk és megjelenítünk, és ezen esemény paramétereként adjuk meg a feltöltött értékeket.

A származtatott értékek, mint amilyen az *Attention* és *Meditation* megjelenítése már biztosítva volt a szerveren, grafikonos formában, folytonos függvényt rajzolva a diszkrét mérések pontjaira.

A szívritmusmérő adatait szintén hasonlóképpen volt logikus megjeleníteni, tehát folytonos függvényként, az idő függvényében. Ám a grafikus négyzetben négy különböző színű vonal esetén már nehéz volt különbséget tenni köztük, így a szerver oldalon dolgozó kollégával úgy döntöttük, hogy a HxM számunkra kevésbé fontos és reprezentatív adatát, a pulzusszámot nem így jelenítjük meg, hanem csak akkor jelenik meg a grafikus nézetben, ha változás állt be az értékében, egy szívet ábrázoló ábrával.

3.7 Offline megjelenítő

A BME Ergonómia és Pszichológia tanszékén, Köles Mátéval folytatott konzultációk eredményeként felszínre került egy probléma, miszerint az általunk alkalmazott online megjelenítő felület legsűrűbben is csak 4,5 másodpercenként tud képeket megjeleníteni. Ez problémát jelent a dinamikusan változó alkalmazások behatóbb vizsgálatánál. Ez különösen a szemmozgás-adatokból készített hőtérképek, illetve az egérpozíció adatok értelmezésénél okozott gondokat. Ezek megfelelő kezeléséhez optimális esetben folyamatos, de legalább 1FPS sebességű képernyőkép rögzítésre lett volna szükség. Mivel ilyen mennyiségű képadat kezelése és ideiglenes vagy végleges tárolása igen erőforrás-igényes feladat, úgy ítéltük, hogy ennek a funkciónak az Android platformon történő megvalósítása nem volna előnyös. Emellett ilyen mennyiségű képadat fogadására a szerver sem volt felkészítve, ezért úgy döntöttünk, hogy az ezt támogató megoldás csak Windows platformon és csak offline formában valósuljon meg. Ilyen formán jelentősen több erőforrás állt rendelkezésre, mintha szigorúan törekedtünk volna a mobil eszközökkel történő teljes kereszt-kompatibilitásra, és mégis lehetővé tettük a nagyobb granularitású adatokat igénylő mérést. Ezt kihasználva lehetőségünk nyílt a szenzorokból kinyerhető összes adat rögzítésére és ezek esetleges későbbi feldolgozására, illetve megjelenítésére. Ez magában foglalhatja az egyes szenzorok által szolgáltatott technikai információkat (úgy mint: akkumulátor feszültség, jelerősség, stb.), valamint

feldolgozatlan vagy félig feldolgozott adatokat (úgy mint: nyers EEG jel, spektrumbontott EEG jel, pupillaméret, tekintet-fixáció, stb.).

A fent említett új információk közül előre nem ismert, hogy egy adott célú mérésnél melyik bizonyulhat leginkább hasznosnak, ezért az offline megjelenítő tervezésénél fontos szempont volt a modularitás. Tehát végső soron szükség volt egy gyors képmentési metódusra, egy ehhez tartozó képmegjelenítőre, valamint a megjelenítőbe integrált, a mérési adatok grafikus formában történő ábrázolására alkalmas panelekre.

3.8 Osztályozó algoritmus

Szintén az Ergonómia és Pszichológia tanszék munkatársával, Köles Mátéval folytatott konzultációk egyik tanulsága volt, hogy az ilyen jellegű adatok manuális feldolgozása igen körülményes és időigényes feladat. Ezért szükségesnek találtuk egy olyan módszer keresését, amellyel a mérési adatpontok csoportokba rendezhetők, az adott pillanatban feltételezett érzelmi-figyelmi állapotok szerint, a szenzorokból származó adatok alapján, automatikusan.

4 Megvalósítás

4.1 EEG eszköz illesztése

4.1.1 Android platform

A Mindwave Mobile EEG-t a gyártó által biztosított Java SDK segítségével volt lehetséges illeszteni a platformhoz. A gyártó által definiált osztályokhoz a *ThinkGear.jar* fájl segítségével férünk hozzá. Engedélyeznünk kell az alkalmazás számára a Bluetooth-hozzáférést (ezt az alkalmazás Manifestjében tehetjük meg).

Az eszköz beállításához létre kell hoznunk az Activityben egy példányt a *TGDevice* és a *BluetoothAdapter* osztályokból. A *TGDevice* az alkalmazásunkkal egy handler függvény segítségével kommunikál, üzeneteket használva. Az alábbi táblázat mutatja az üzenetek lehetséges állapotait:

Message	Description	Data
MSG_STATE_CHANGE	The state of the TGDevice has changed	STATE messages stored in the `arg1` field of the message object
MSG_POOR_SIGNAL	Signal quality status data	The poor signal status from the headset is stored in the `arg1` field of the message object
MSG_ATTENTION	Attention level data	The attention level is stored in the `arg1` field of the message object
MSG_MEDITATION	Meditation level data	The meditation level is stored in the `arg1` field of the message object
MSG_BLINK	Strength of detected blink	The blink strength is stored in the `arg1` field of the message object
MSG_RAW_DATA	Raw EEG data	The raw EEG value is stored as an int in the `arg1` field of the message object
MSG_EEG_POWER	EEG powers data	The EEG powers are passed in as TGEegPower object in the `obj` field of the message object
MSG_RAW_MULTI	Multi-channel raw data	The multi-channel raw data is passed in as a TGRawMulti object in the `obj` field of the message object
MSG_HEART_RATE	Heart rate data	The heart rate data is passed in as an int in the `arg1` field of the message object

9. ábra A TGDevice lehetséges üzenetei [25]

Az adatok küldését a *TGDevice* osztályban definiált *connect(), start(),* és *close()* parancsok segítségével egyszerűen meg tudjuk tenni.

Tehát a feladat annyi volt, hogy ezeket az osztályokat és értékeket a számunkra megfelelő módon használjuk föl. Ehhez az Attention és Meditation üzenetekben lévő értékeket használtuk fel.

Az eszközt tehát sikeresen illesztettük a rendszerhez, és a továbbiakban a keretrendszerben már jelenlévő és azzal együttműködő megvalósítást használtuk a tesztelés során.

4.1.2 Windows platform

Ezt az eszközt elsősorban a gyártó mobil eszközökhöz tervezte, így a desktop környezetben sokkal kisebb a támogatottsága. A gyártó biztosít egy ThinkGear Connector elnevezésű programot, ami a Windows-on keresztüli párosítás után, fogadja az EEG által, Bluetooth-on küldött adatokat és ezeket elérhetővé teszi egy lokális webszerveren keresztül JSON csomagok formájában. Ennek a feldolgozására a gyártó további segítséget nem nyújt. Azonban némi kutatás után, Andreas Borg és Bruno Panerai Velloso által elérhetővé tett programkódok mentén, egy Socket alapú megoldáson keresztül, sikerült ezen adatok fogadását lehetővé tenni. Az Attention és Meditation adatok a keretrendszerben foglalt módon feltölthetők a szerverre, míg az összes fogadott adat menthető offline feldolgozáshoz.

4.2 HxM eszköz illesztése

4.2.1 Android platform

A gyártó által biztosított Java SDK segítségével a következő előre definiált osztályokat tudjuk felhasználni az eszköz illesztéséhez: a BtClient osztály felelős a bluetooth kapcsolat működésért, egy eseménykezelő regisztrálásával figyelhetjük az általa küldött eseményeket. Példányosítanunk kell a ZephyrProtocol osztályt, melyhez egy listener segítségével kapcsolódva nyerhetjük ki az adatokat a szenzorból.

A csatlakozás után a tervezéskor említett adatcsomagokat kapjuk meg, és ebből tudjuk felhasználni a számunkra fontos adatokat, mely jelen esetben a szívritmus és az elmúlt 15 szívverés időbélyege.

Mivel a keretrendszerben jelen lévő implementáció a rendszer egy régebbi verziójához volt megalkotva, az újabb verzióhoz illeszkedve meg kellett valósítanunk az adatok szerverre történő feltöltését.

Ehhez a rendszerben egy új eseményt kellett definiálnunk (HeartRateChangedGameEvent), mely átalakított formában az eredeti eseményt (HeartRateChangedEvent) a szerverre feltölthető formátumúvá teszi.

A keretrendszerbe illesztés folyamatát az adatstruktúra leírása c. alfejezetben mutatjuk be részletesen.

4.2.2 Windows platform

Mivel ezt az eszközt is elsősorban mobil környezetben történő felhasználásra tervezték a gyártó itt csak az adatok logolását támogatja Windows környezetben. Itt teljesen saját illesztőt kellet írni. A Windows-on keresztüli párosítás után az eszközfelderítést, a csatlakozást és a Bluetooth stack elérését a BlueCove által biztosított Java illesztőkönyvtáron keresztül oldottuk meg. Az eszköz, specifikáció szerint, 1 másodpercenként küld 1 darab, 60byte-os adatcsomagot. Ennek biztos fogadására fél másodpercenként vizsgáljuk a stack-et és amennyiben tartalmaz 60byte-ot, azt kimentjük és dekódoljuk (8. ábra). A dekódolt csomagból ugyanazok az adatok kerülnek feldolgozásra és nagyjából ugyanolyan módon, mint Android platformon. Annyi különbség mutatkozik, hogy a HPV számításához implementálva van az FFT-alapú, valamint a Burg-algoritmus alapú megoldás is, ebből az utóbbi van aktívan alkalmazva. A feldolgozott adatok itt is elmenthetők offline feldolgozásra.

4.3 EyeTribe illesztése

4.3.1 Android platform

Jelenleg nem elérhető a piacon a rendszerünk koncepciójába illeszkedő (tehát megfizethető és hordozható) szemmozgáskövető eszköz Android-os fejlesztőkörnyezettel, így a szemmozgáskövetést Android-on a PC-s megoldáson keresztül tudjuk megvalósítani. Ehhez több megoldást is használhatunk, melyek elérhetők a piacon: Miracast, Samsung SydeSinc, Mobizen alkalmazás.

4.3.2 Windows platform

Ehhez az eszközhöz a gyártó Windows platformra teljes támogatást biztosít. A számítógéphez USB 3.0-ás foglalaton keresztül csatlakoztatva azonnal települ és a továbbiakban az operációs rendszer webkameraként kezeli. A csatlakozást követően, az EEG-hez hasonlóan, az adatok itt is egy lokálisan futó webszerveren keresztül érhetőek el, JSON csomagok formájában. Ezek kezelését a gyártó által biztosított SDK-n keresztül oldottuk meg. Minden használat előtt az eszközt kalibrálni kell. Ez szintén megoldható a gyártó által biztosított programmal.

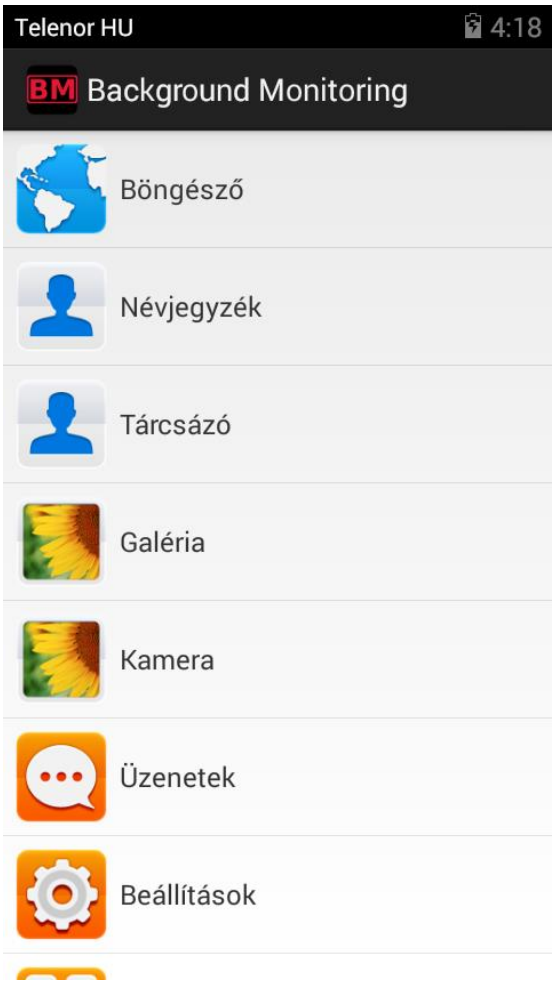
4.4 Monitorozó alkalmazás Android platformra

4.4.1 Monitorozás Activity segítségével

Az alkalmazás megvalósításakor a keretrendszer lehetőségeit kihasználva elsőként egy, a keretrendszer *GameBase* osztályából leszármazó *Activity-t* hoztunk létre. Ezen keresztül hozzáférhetünk a már definiált függvényekhez és osztályokhoz. Ez az *Activity* a

```
@RequireLogin(securitylevel=SecurityLevel.REQUIRED, role=ROLE.USER)
```

annotáció segítségével kapcsolódik a szerverhez, és a megfelelő adatokkal belépve rögzíthetjük a mérést. Az alkalmazás a bejelentkezést követően egy keresést futtat le, és a PackageManager segítségével kilistázza az összes telepített alkalmazást, megjelenítve azokat egy ListView-ban. A listázásra szűrést kell bevezetnünk, hogy ne jelenítsen meg minden telepített alkalmazást, ugyanis ebben rengeteg fölösleges lenne a számunkra. Kétféleképpen szűrhetünk: egyrészt kiszűrhetjük a rendszeralkalmazásokat, mely azzal jár, hogy a gyárilag telepített alkalmazások (pl. Gmail, Maps, Naptár) nem jelennek meg a listában. A másik lehetőség, hogy a Launcher iconnal rendelkező alkalmazásokat listázzuk. A programban utóbbi lehetőséget valósítottuk meg.



10. ábra A telepített alkalmazások listája

Ha a listából elindítunk egy alkalmazást, az egy explicit intent segítségével meghívódik, és ezzel egy időben új játékmenet indul a keretrendszeren a *startNewGamePlay()* függvényhívás segítségével. Ezzel együtt a monitorozott adatok feltöltése a szerverre megkezdődik.

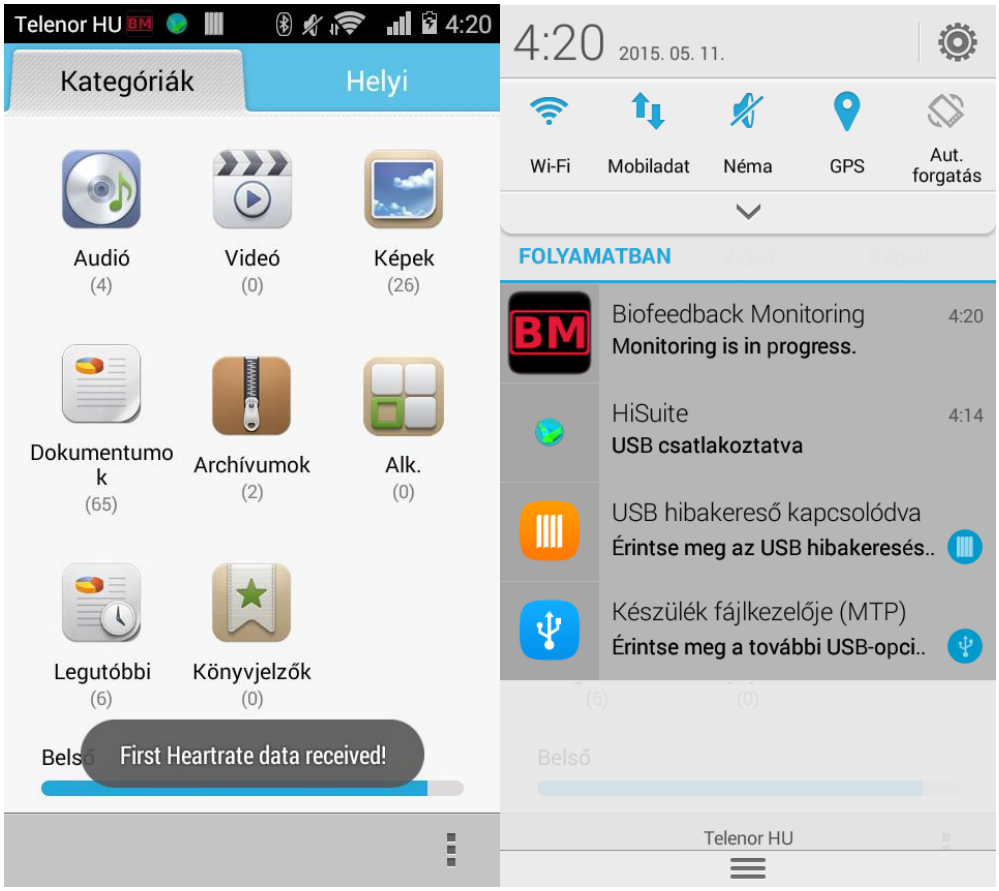
A „játékmenet” akkor fejeződik be, amikor az indító Activity-t bezárjuk.

A könnyebb használhatóság érdekében hozzáadtunk két kiegészítő funkciót az alkalmazáshoz:

1.) Ha elkezdjük a monitorozást, megjelenik az értesítősávban egy *„Ongoing Notification”* mely jelzi, hogy éppen folyamatban van az adatok rögzítése és felküldése a szerverre.

2.) Mivel a HxM szívritmusmérő csak az első 15 szívverés alapján tudja elkezdeni számolni a HPV értékét, nem érdemes addig elindítanunk a monitorozandó alkalmazást, amíg nem jelenik meg az első érték. Ezért az első

érték érkezésekor egy *Toast message* segítségével jelezzük, hogy megkezdhetjük a monitorozást.



11. ábra Toast Message és Ongoing Notification

4.5 Monitorozó alkalmazás Windows platformra

Az alkalmazás az AdaptEd keretrendszer egy módosított verziójára épül. A Framework és a megfigyelő rendszerrészek mélyebben egymásba épültek, mint az Android-on futó verzióban. Ebben a konstellációban a Supervisor alkalmazás felől lehet vezérelni a Framework-öt és ezzel mérési folyamatot is. Gyakorta a rendszer a parancsokra válaszüzenetet is küld. Ehhez módosításokat kellett végrehajtanunk mindkét programon, amellett, hogy a keretrendszer, specifikációja szerint, támogatja az egyedi üzenettípusok definiálását.

A programok portolását próbáltuk úgy megoldani, hogy az Android-specifikus osztályokat és metódusokat funkciójában hasonló megoldásokkal próbáltuk pótolni. Ahol ez nem sikerült, ott strukturális módosításokra is szükség volt. Praktikussági okokból a Framework adatbufferét az eredeti SQLite megoldást MySQL-re cseréltük.

A megfigyelő rendszer strukturális alapját az információs források szolgáltatják. Ezek jellemzően a szenzorok illesztői, de ide csatolható a képrnyőkép-készítő, vagy az egérhasználatot figyelő programrészlet is. Ezekre a forrásokra listener-ként csatlakoznak az információ feldolgozó egységek, akik később a kész adatokat továbbítják a Framework felé *GameEvent-ek* formájában.

A vizsgálat elindítása előtt a Supervisor felől összeállítható a feldolgozókból az épp használni kívánt mérési konstelláció. A rendszer csak akkor kapcsol be egy forrást, ha arra aktivált feldolgozó hivatkozik. Amennyiben a forrás külső program futását is igényli (mint pl. ThinkGear Connector), annak működését hozzáadás előtt a Framework ellenőrzi és esetleges hiba esetén üzenetet küld a Supervisor felé. Az adatok offline rögzítése a forrásokba integrált funkcióként lett megvalósítva.

A források közül kiemelendő a képernyőkép-készítő, mert ennek a működéséhez van szinkronizálva a hőtérképek feltöltése is. Az erre a modulra speciális listener-ként feliratkozó feldolgozók értesítést kapnak egy kép készítéséről egy timestamp formájában. A szerveroldalon a feltöltött hőtérképek az azonos timestamppel ellátott képernyőképre kerülnek fel overlay-ként. A képernyőképek rögzítése a java.awt csomagban található megoldással történik.

Az egér pozíciójának lekérése illetve az egérgombok lenyomásának érzékelése egy Kristian Kraljić által írt, Windows natív megoldáson keresztül történik, mivel ezek az információk nem érhetőek el a Java programablakán kívül, a monitorozó alkalmazást a háttérben futtatva.

4.6 Az adatok struktúrája

Az adatok, mint a tervezésnél említettük, események paramétereként kerülnek feltöltésre a szerverre. A HxM két értékének segítségével mutatjuk be a folyamatot:

Létre kell hoznunk a megfelelő esemény osztályokat a Framework és Supervisor által közösen használt Library projectben, jelen esetben *HeartrateChangedEvent* és *HeartRateChangedGameEvent* néven. Előbbi a szenzormodul által küldött és létrehozott esemény, utóbbi pedig az ebből létrejövő szerverre feltöltésre kerülő osztály.

A megfelelő működéshez szükség van az osztályok megfelelő eseménykezelő objektumokba való felvételére: az *AbstractEventProcessorVisitor.java,*

AbstractFalseEventMatcherVisitor.java, AbstractTrueEventMatcherVisitor.java, EventMatcherVisitor.java, EventProcessorVisitor.java fájlok módosítására van ehhez szükség.

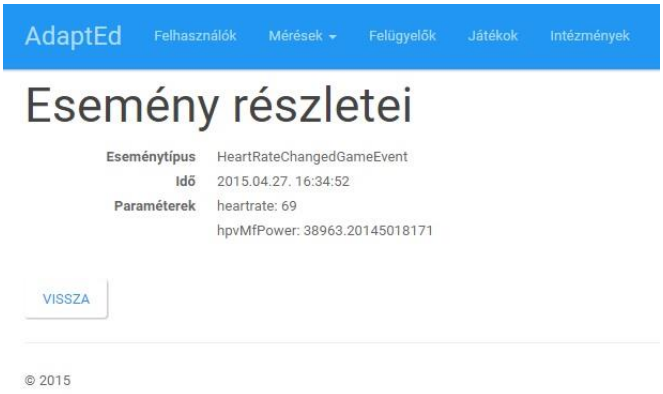
A *MessageGenerator* osztályban létre kell hoznunk a függvényt, mely az esemény hatására létrehozza a számunkra megfelelő üzenetet. Az eseményt fel kell dolgoznunk a felügyelő alkalmazás (Supervisor) számára is, mely a fenti osztályok használatán felül a *SensorEventDecoder* osztály segítségével dekódolja az esemény által küldött üzenetet.

A szenzorral való kommunikációt megvalósító InnoLearnSensors projektben van megvalósítva az adatok kinyerése és feldolgozása. Az adott eszközhöz tartozó modul feldolgozza az adatokat és elküldi a keretrendszernek, melyet a keretrendszerben levő modul (jelen esetben *HRMSensorModule*) egy broadcastreceiver-rel fogad, és létrehoz egy eseményt. A *MeasurementModule* nevű osztály dolgozza fel az összes eseményt, melyet a keretrendszer állít elő a mérésekből, és küldi el azt a *GameBase* számára. Egy új játék létrehozásakor ezt a *GameBase* osztályt származtatjuk le, tehát ez rendelkezik minden alapfunkcióval, amit a későbbi Activity-k felhasználnak. Az eseményt a *GameBase* egy broadcastreceiver segítségével elkapja, és üzenetként elküldi a háttérben folyamatosan futó *FrameworkService* számára. Ez egy proxy-n keresztül létrehozza a Game Eventeket, azaz jelen esetben a HeartRateChangedEvent-ből a HeartRateChangedGameEvent példányt, és a szerver ezt az osztályt képes fogadni és feldolgozni.

Az adatok egy JSON Object paramétereként kerülnek föl a szerverre, amit az adott esemény osztályában (jelen esetben HeartRateChangedGameEvent) definiálunk):

```
@Override
	protected JSONObject getParametersForJSON() {
		JSONObject ret = new JSONObject();
		try {
			ret.put("heartrate", String.valueOf(heartRate));
			ret.put("hpvMfPower", String.valueOf(hpvMfPower));
		} catch (JSONException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return ret;
	}
```

Ezek az értékek a szerveren is megjelennek, a táblázatos nézet eseményeire rákattintva:

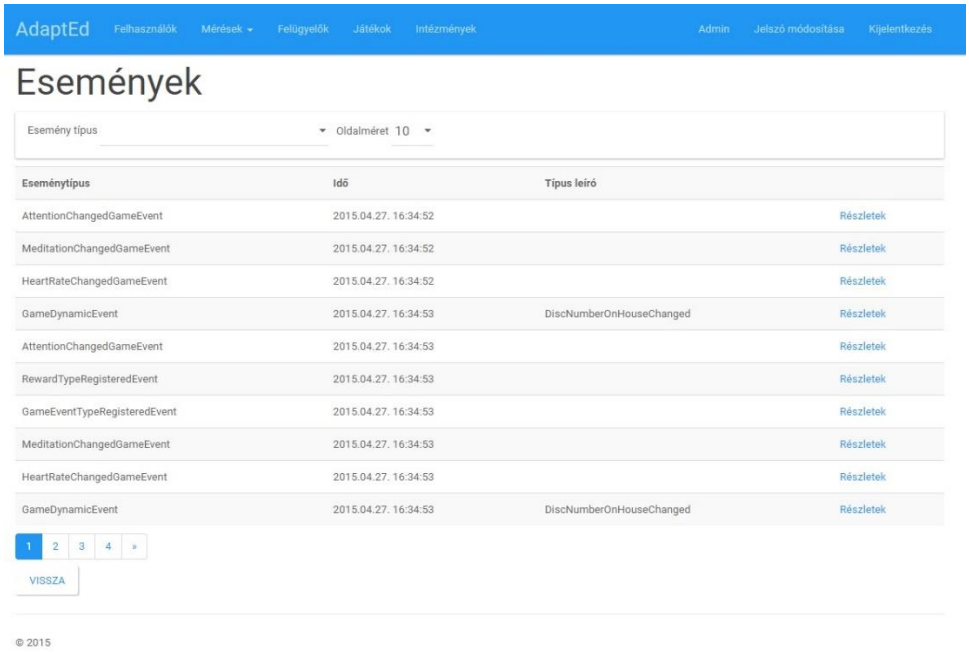


12. ábra A HeartRateChangedGameEvent esemény részletei

Az események egymástól függetlenek, így nem befolyásolja az események száma az események működését.

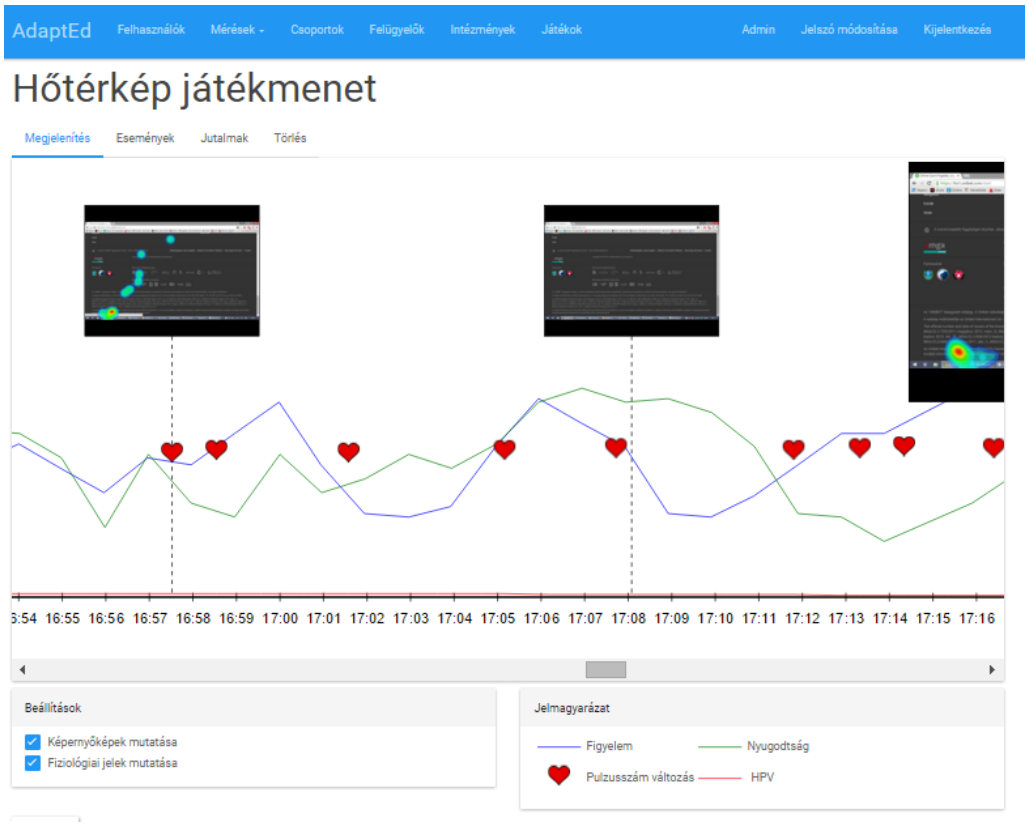
Az EEG szenzor jelei és eseményei a fentiekhez hasonlóan vannak definiálva a rendszerben. Az overlay-t, melyre a hőtérképet rajzoltuk rá, a ScreenshotEvent eseménnyel párhuzamosan juttatjuk el a szerverre.

A szerverre való feltöltés után az adatok egyrészt megjelennek a táblázatos nézetben:



13. ábra Események a szerveren

Másrészt grafikusan is meg tudjuk jeleníteni őket:



14. ábra Grafikus megjelenítés a weben

4.7 Offline megjelenítő

4.7.1 Adatok offline rögzítése

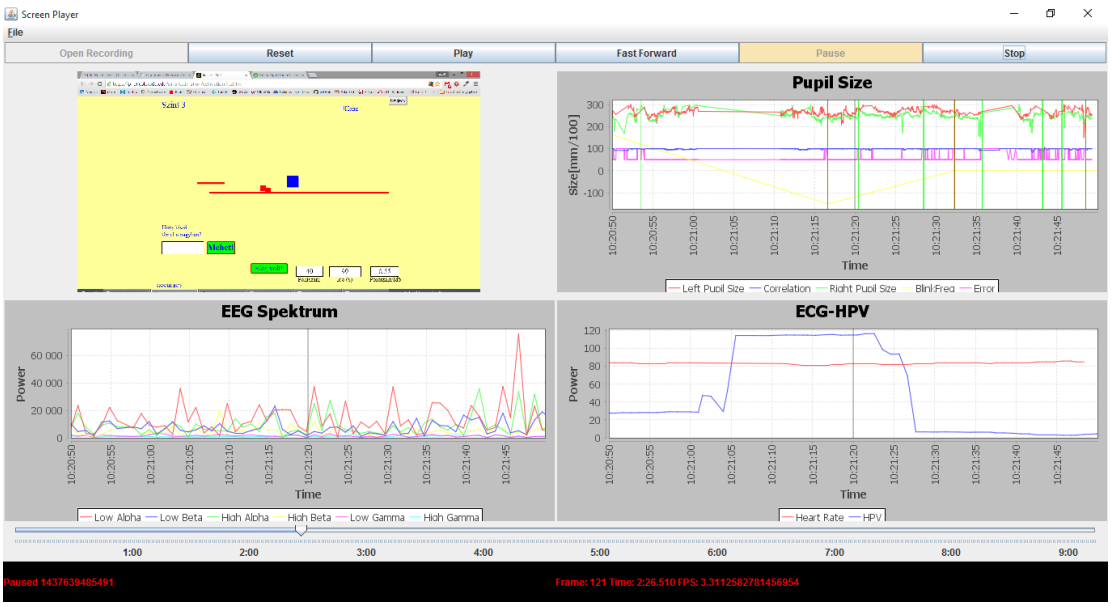
Az offline megjelenítés első problematikája a rögzített adatok tárolása. Kezdetben a szenzorok adatait, eszközönként szeparáltan .csv kiterjesztésű táblázatokba mentettük. A képernyőképeket egy nyílt forráskódú Java alkalmazás segítségével rögzítettük, annak reményében, hogy az majd módosítható lesz olyan formán, hogy az elkészült képekhez társítsa azok elkészítési időbélyegjét, adatszinkronizálási okokból. Ennek a módszernek az volt a hátránya, hogy a mérési adatok több fájlba kerültek, ami adatbiztonsági kockázatokat jelentett, valamint a különböző struktúrájú szenzoradatok kezelése is problematikusnak bizonyult. A képrögzítő alkalmazás sem váltotta be a hozzá fűzött reményeket, mert ugyan módosítható volt úgy, hogy tárolja az időbélyegeket, de sajnos az adatokat egy láncolt lista jellegű struktúrában tárolta, amiben az előre- és visszatekerés tetszőleges időponthoz nem volt megvalósítható.

A fenti problémák megoldására úgy döntöttünk, hogy az adatokat praktikusabb adatbázisban rögzíteni. Némi keresés után a választásunk a MongoDB nevű ingyenes NoSQL megoldásra esett. Ennek a rendszernek komoly előnye, hogy az adatstruktúrája viszonylag kötetlen és így jól tárolhatók benne a többféle szenzorokból származó adatok. Emellett képek rendszerezett tárolására is alkalmas és hatékonyan archiválható, valamint tömöríthető is, ami fontos szempont nagy mennyiségű mérési adat esetén. A NoSQL megoldások hátránya, hogy nem garantálják az ACID működést. Jelen esetben úgy ítéltük, hogy ez nem komoly probléma, hiszen az adatok csak egyszer kerülnek írásra, sosem módosulnak és elkészülésük után is feltehetően egyszerre csak egy forrás olvassa. Emellett mivel a mérési eredmények jellemzően statisztikai jellegűek, ezért az egyes rekordok meghibásodása nem komoly probléma a nagy egész tekintetében.

4.7.2 Adatok megjelenítése

Mivel Windows alatt nincs időgarancia a programszálak futásra kerülésében, ezért a képek rögzítésénél sem garantáltak az egyenletes időközök. Ennek okán megjelenítésnél a frame-ek közötti időintervallumokat előrebufferéléssel és a betöltött képek időbélyegei alapján a program automatikusan állítja. A többi grafikus megjelenítő panel ehhez szinkronizálva, fél perces előretartással tölti be az adatokat az adatbázisból.

A megjelenített képernyőképre overlay-ként felkerül az egér, illetve a tekintet pozíciója a képeken. Átláthatósági szempontokat figyelembe véve a képmegjelenítő mellett három grafikus panelnek jutott hely. Ebből az elsőn a bal és jobb pupillaméret, a kettő közti korreláció és a pislogás vélt frekvenciája jelenik meg. A másodikon az EEG spektrumbontott jelei láthatók. A harmadikról a pulzus, a HPV és a mérésvezető által hozzáfűzött rövid kommentek olvashatóak le. Bármely grafikon tartalmi összetételének megváltoztatása nagyon kevés programkód módosítással megoldható, valamint könnyen új adatforrások is hozzáadhatók a rendszerhez, a modularitás jegyében.



15. ábra Offline megjelenítő felület

4.8 Osztályozó algoritmus

Ez a megoldás jelenleg az offline megjelenítőbe van integrálva, megjelenítés előtt, illetve közben fut. Mivel az adatok értékkészletüket tekintve igen különböző intervallumba esnek, ezért az együttes használatukhoz a minimum és maximum értékük alapján számított százalékos értéküket használtuk. A szélsőértékek egy rendezett lekéréssel könnyen megkaphatóak. Az együttes feldolgozásnál fél másodpercenkénti mintavétellel N-koordinátájú adatpontokat hoztunk létre, ahol N a használni kívánt információforrások száma.

4.8.1 Előfeldolgozás

Az osztályozó algoritmus elindítása előtt szükség van az osztályok számának és azok középpontjának definiálására. Ezt úgy értük el, hogy felveszünk egy teljes gráfot, aminek pontjai az N-dimenziós adatpontok és éleit a pontok közti távolsággal jellemezzük. Ezután egy szintet adaptívan állítva, elhagyjuk a meghatározott szint feletti értékű éleket. Az így kialakuló szigetek közül elhagyjuk azokat, amik elhanyagolhatóan kevés pontot tartalmaznak (az aktuális beállítás szerint 10 alatti). A megmaradó szigetek pontjainak a mértani közepe szolgáltatja az osztályozó csoportjainak kiindulási adatpontjait. Az előfeldolgozó jelenleg minden megjelenítés előtt lefut, de a jövőben

használatát kiváltanánk a nagyszámú mérésből tapasztalatilag kalkulált, fix számú és koordinátájú osztálykezdőpontokkal.

4.8.2 Osztályozó

A jelenleg használt megoldás egy Dusza Andrea hallgatótársunk által implementált, K-means elvű osztályozó [30]. Ennek lényege, hogy a beérkező adatpontot ahhoz a csoporthoz rendeli, amelynek közepéhez a legközelebb esik. Egy csoporthoz hozzáadás után a rendszer újrakalkulálja annak közepét. A beérkezett adatpontokat a rendszer öregíti és egy bizonyos idő után törli azokat, garantálva ezzel, hogy a rendszerben ne legyen dinamikus kalkulálásra már alkalmatlan mennyiségű adatpont. Jelenleg ez a megoldás csak a mérés visszajátszásánál érhető el, de a jövőben, amikor már rendelkezünk a fix osztálykezdőpontokkal, a megoldást szeretnénk úgy implementálni, hogy az már a vizsgálat közben, valós időben is tudjon egy becslést szolgáltatni.

5 Eredmények

5.1 Kísérletek

5.1.1 A kísérletek célja

A kísérletek elsődleges célja a rendszer működőképességének igazolása volt. Egyfelől egy szakember segítségével milyen következtetések vonhatóak le a valamilyen formában megjelenített adatokból. Illetve annak felmérése, hogy ehhez képest milyen aspektusból és milyen pontossággal alkalmazható az osztályozó algoritmus. Ennek felmérésére egy folyamatosan nehezedő referencia-feladatsorhoz hasonlítunk egy alkalmazáson elvégzendő feladatsort. A kísérletek során az általunk megalkotott a rendszert az Ergonómia és Pszichológia tanszéken régóta működő, tesztelt és stabil megoldással hasonlítottuk össze (mely rendszer beszerzési ára azonban a miénk több, mint tízszerese volt).

5.1.2 A kísérletek menete

A kísérlet menetének kialakításakor hasonló témákban végzett vizsgálati folyamatokat [31][32], illetve Köles Máté tanácsait vettük alapul. A kísérlet egy relaxációs folyamattal indul. Ennek célja a felhasználó nyugalmi értékeinek a meghatározása. A kísérletet végző személynek az a feladata, hogy 3 percig nézzen a képernyőre és próbáljon ellazulni. A kijelzőn ekkor egy kevés mozgást tartalmazó, tájképet ábrázoló videó fut, ami alatt erdei ambiens zajok hallatszanak[1]. Megfelelő lehetőségek híján a méréseket gyakorta a folyosón kellett végezni. Ebben az esetben a felhasználót fülhallgató használatára kértük, a nem kívánt zavaró hatások csökkentése érdekében.

A kísérlet következő fázisában megpróbáltuk meghatározni a felhasználó különböző mentális megterheléséhez tartozó szenzorértékeket. Ezt egy becslő játék segítségével oldottuk meg[2]. Az első fázisban a felhasználó szabadon játszhat a legegyszerűbb fokozaton, amíg úgy nem érzi, hogy teljesen el nem sajátította a játék mechanikáját. Amint ezzel elkészült, mind a három, egyre nehezedő nehézségi fokozaton 2-2 percet kell játszania, azzal a céllal, hogy minél több pontot szerezzen az adott szinten.

[1] https://www.youtube.com/watch?v=pUdZFXsHk0o

[2] https://phet.colorado.edu/sims/estimation/estimation_hu.html

Motiváció gyanánt minden szinten a legjobban teljesítő játékosnak apró nyereményt ajánlottunk fel, a mérések lezárása után.

A kísérlet utolsó fázisában egy honlapon fellehető információkat kellett megtalálniuk[1]. Ehhez bármilyen, a számítógépen rendelkezésre álló segédeszközt használhattak. A feladatok a következők voltak, ebben a sorrendben:

1. Olvassa fel hangosan a főoldalon található utolsó mondatot! (ez keresés nélkül elérhető információ)
2. Hány lépésből álnak az októberi (adott havi) küldetések? (ez a főoldalról 1 kattintással megtalálható információ)
3. Mennyi az odds arra, hogy a labdarúgó Bajnokok Ligája, Arsenal – Bayern München (tetszőleges, viszonylag aktuális meccs) meccsen, félidőben 3-0 az állás? (ez a kérdés a sportfogadás oldalról, a feltevés pontjai mentén következetesen haladva, megválaszolható. A megoldás keresőfunkcióval gyorsítható)
4. Találja meg, hogy honnan lehet élő chat segítséget kérni! (ez közvetlen kereséssel nem megoldható, de bármely oldalról 1-2 kattintással elérhető)

A felhasználók bármikor visszaismételtethették a kérdést, fogalmi/nyelvi problémák esetén kérhettek segítséget. Ha egy feladatot 5 perc alatt nem sikerült megoldania, a felhasználó 2 percenként kérhetett tippet a feladat megoldásához. 11 perc után, ha nem sikerült megoldani a feladatot, megmutattuk a megoldást. A mérés aktív része átlagosan 25-30 percig tartott, a mérőeszközök felhelyezése, kalibrálása és a mérési folyamat elindítása nagyjából ugyanennyi ideig.

[1] https://hu1.unibet.com/start

5.1.3 Nehézségek

A szűkös időbeni erőforrások, a kísérlet hossza miatt, valamint komoly motiváló eszköz nélkül igen nehéz volt a kísérleti alanyok beszervezése. Ezen okokból nagyon alacsony számú mérést sikerült elvégeznünk, annak ellenére, hogy kísérleti eredményeknek statisztikai jellegűeknek kellene lenniük.

A mérések során többször előfordult, hogy valamelyik gyártói illesztőprogram nem tudott csatlakozni a hozzá tartozó szenzorhoz. Emellett a kísérletek során két EEG készüléket is alkalmaztunk, amelyek közül az egyikről később kiderült, hogy hiányos adatcsomagokat küld. Többször előfordult, hogy az operációs rendszer automatikusan újraosztotta a COM-portokat és így a Framework nem tudott csatlakozni az EKG-hoz.

A hibák többsége orvosolható volt a számítógép néhányszori újraindításával, de mivel a legtöbb kísérletre jelentkező erre szánható ideje véges volt, gyakorta hiányos konstellációval kellett lefolytatni a mérést. Halmozott hibák esetén előfordult, hogy a mérést le kellett mondani.

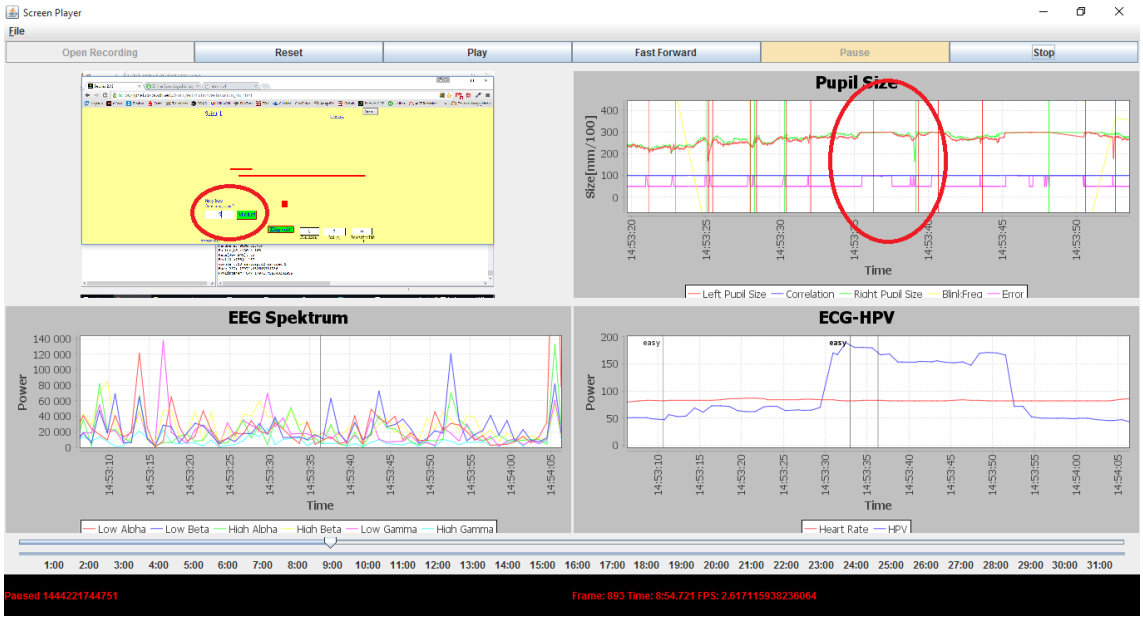
5.1.4 Eredmények

Az elvégzett mérések eredményüket tekintve jól alkalmazhatóak a rendszer képességeinek és hibáinak felmérésére és így egy nagyobb volumenű kísérletsorozat előfutára lehet.

A legfontosabb tapasztalatnak tekinthetjük, hogy a rendszernek jelen formájában vannak stabilitási problémái. A hardver oldali problémákra jelenthet megoldást új eszközök felkutatása és integrálása. Ez igen komoly extra költségekkel járhat. A szoftver oldali problémák kétféleképpen oldhatóak meg. Első megoldásként megkereshetőek és orvosolhatóak a fennálló hibák. Ennek megvalósítása azért volna körülményes, mert ezek közül több is lehet operációs rendszer-szintű probléma és/vagy igényelheti gyártói szoftverek elhagyását és azok újraírását. A második megoldás lehet a rendszer agilisabbá tétele, felkészítése az igen hiányos adatok fogadására és feldolgozására. Ebben a témakörbe tartozhatnak továbbá az egyes önjavító megoldások is.

Emellett több hibára sikerült rávilágítani magán a mérési folyamaton belül is. Annak ellenére, hogy a kísérleti alanyok túlnyomó többsége érdekesnek találta a becslős játékkal való foglalkozást, ez nem bizonyult a legjobb megoldásnak a mentális terhelés fokozatos felmérésére. Egyrészről a minél több pont megszerzésére való motiválás nem

volt a legjobb ötlet, mert voltak felhasználók, akik a mennyiség, és voltak, akik a pontosság oldaláról közelítették meg a problémát. Ez okozhat különbséget a mentális terhelés szempontjából, ami nem praktikus a referencia felvételénél. Szintén a felhasználók véleménye mentén a három nehézségi szint nem bizonyult lineárisan nehezedőnek, az elsőt sokkal könnyebbnek találták, mint a másik kettőt. Továbbá nem szerencsés olyan megoldásokat használni, ahol gépelni is kell, mert ehhez a legtöbb embernek le kellett vennie tekintetét a kijelzőről és ezzel hiányokat okozva a pupillaméret és egyéb szemmozgás-adatokban. A jövőben erre lehet használni képernyő-billentyűzetet, de az így történő gépelés sokkal időigényesebb, nem alkalmas gyorsasági feladatok megoldására.



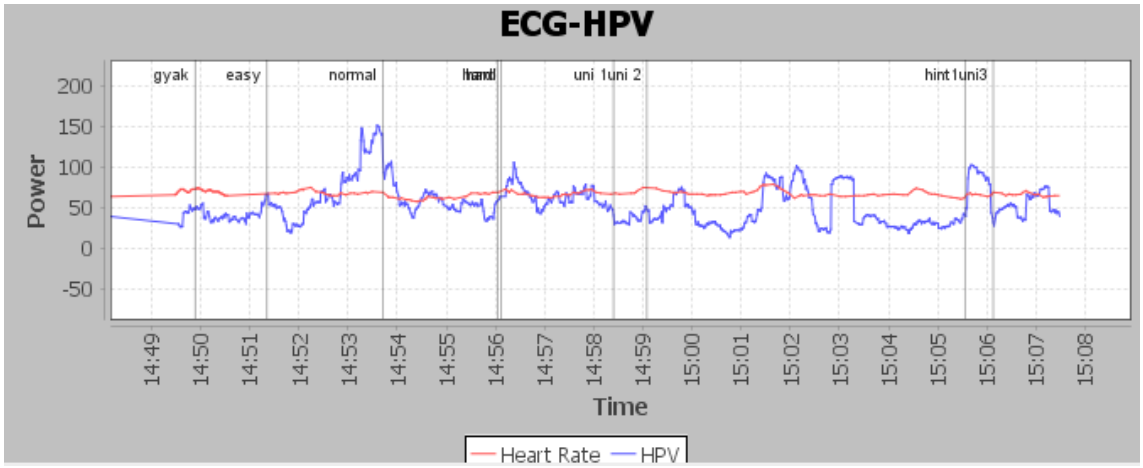
16. ábra Hiba a pupillaméret adatokban gépeléskor

Emellett olyan kísérleti helyszínre is szükség lesz, ahol kevésbé változnak a mérés közben a fényviszonyok, amik megnehezítik a pupillaméret adatok értelmezését. Hasonló okok miatt nem praktikus szemüveges kísérleti alanyok alkalmazása sem.

Mivel az ilyen jellegű kísérleteknek sok nem jól automatizálható eleme is van, ezért a jövőben praktikus volna felvételt készíteni a felhasználókról, vizsgálat közben. Erre megfelelő eszköz lehet a vizsgálati platformként szolgáló notebook beépített mikrofonja és webkamerája.

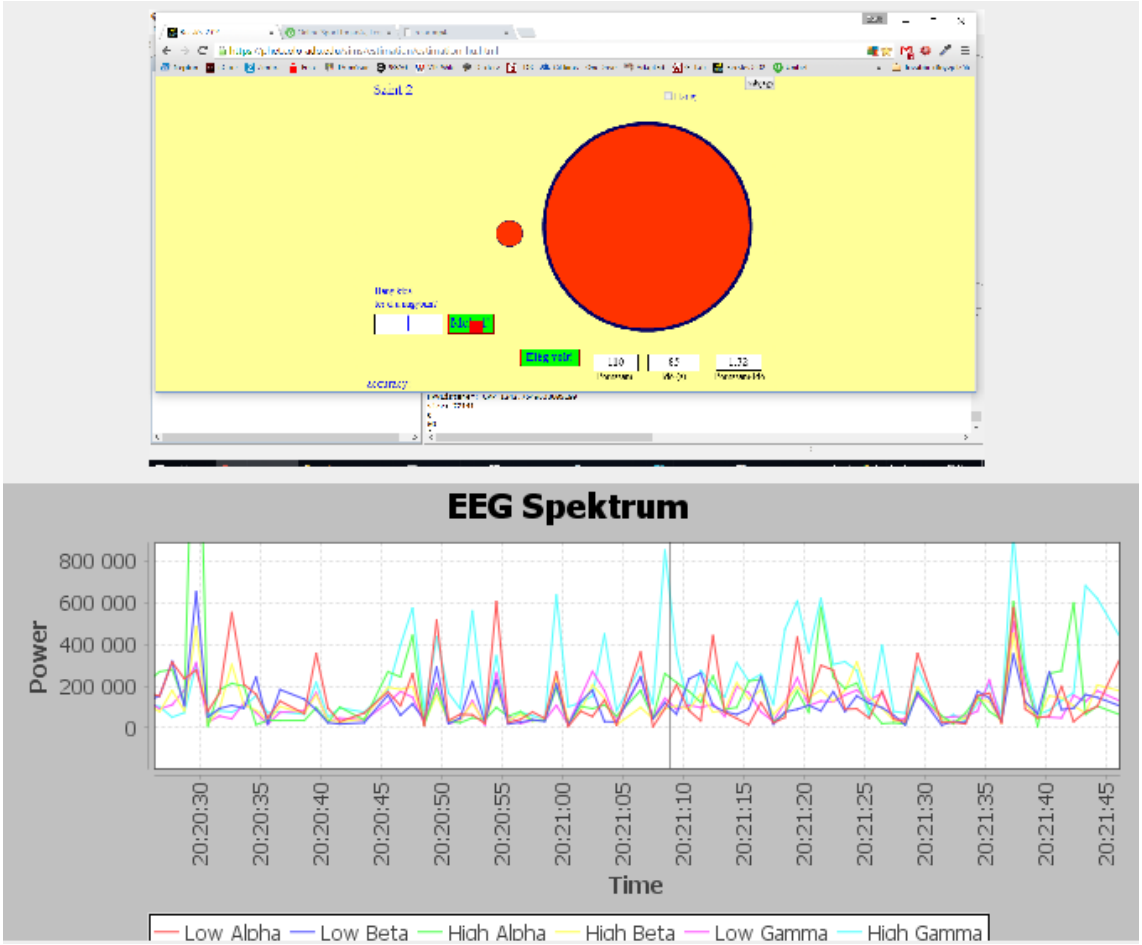
17. ábra Mérési összeállítás

Néhány jelenségre is felfigyeltünk, vagy feltevést igazoltunk, ami befolyásolhatja a kutatás jövőbeli irányát. Igazoltuk, hogy a HPV érték egy feladat megkezdésekor hirtelen letörik, a feladat megoldásakor pedig emelkedik. Ez alkalmassá teheti a rendszert az egyes feladatok megoldási idejének automatikus becslésére.



18. ábra HPV érték változása egy kísérlet során

Emellett felfigyeltünk arra, hogy a becslős játéknál főleg a nehezebb becsléseknél több felhasználónál is elég szignifikáns csúcsok láthatóak. Ez felhasználható lehet rövid, de intenzív mentális erőfeszítések kimutatására.



19. ábra Gamma csúcsok (világoskék) közepes nehézségű becsléseknél

A kísérletek eredményei tehát bizonyítják, hogy habár a rendszer nem ad olyan pontosan elemezhető eredményeket, mint a tanszéki, körülbelül tízszeres árú megoldás, az adatok együttes és kompakt megjelenítése biztosítja számunkra a hasonló következtetések levonását, és az ergonómusok és szoftverfejlesztők segítségére válhat. Így használhatóbb és nagyobb felhasználói élményt biztosító szoftverek fejlesztése lehetséges.

6 Összefoglalás

Dolgozatunkban bemutattuk a megalkotott monitorozó rendszert, a felhasznált biofeedback eszközök elméleti hátterét és működését, a rendszerbe való illesztésük folyamatát. Dokumentáltuk az alkalmazás tervezését és megvalósítását Android és Windows platformon, valamint az offline tárolás és az osztályozás lehetőségét. Ismertettük, hogy a különböző eszközökből kinyert adatok hogyan kerülnek feltöltésre, és hogyan jelennek meg strukturáltan a szerver oldalon. A rendszer működését és létjogosultságát a végrehajtott kísérletek eredményeinek elemzésével demonstráltuk.

Létrehoztunk az említett két platformon egy megfizethető árú és hordozható monitorozó rendszert, mely által képesek vagyunk bármilyen alkalmazás használata közben a felhasználó megfigyelésére, valamint fiziológiás jeleinek rögzítésére, megjelenítésére. Az adatokat lehetőség van offline és az AdaptEd keretrendszer segítségével online módon rögzíteni és megjeleníteni. A rendszer segítségével az ergonómusok és alkalmazásfejlesztők képesek lehetnek a megalkotott szoftverek jobb ergonómiai kialakítására, a használhatóság növelésére és nagyobb felhasználói élmény kialakítására. A felhasználó élettani jeleit valós időben tudjuk követni, így élőben visszacsatolást kapunk az őt érő hatásokról. A kompakt megjelenítés és az osztályozó funkció segítségével könnyen elemezhető adatokat bocsátunk a felügyelést végző személy részére. Ez által lehetővé válik a feladatok nehézségének módosítása, illetve az adott értékekhez megjegyzések fűzése.

További fejlesztésként jelenleg is folyamatban van a rendszer Android TV-re való illesztése. Adott a lehetőség a további platformok (iOs, Windows Phone) hozzáadására. Az újabb, fejlettebb technológiákat alkalmazó biofeedback eszközök elérhetővé válásával tervezzük a rendszer fejlesztését. A felhasznált algoritmusokat is szeretnénk továbbfejleszteni, valamint számunkra pontosabb adatokat kinyelni a nyers jelekből a megfelelő algoritmusok segítségével, együttműködve az Ergonómia és Pszichológia tanszékkel. Egy másik lehetőség a felügyelő alkalmazás specifikusabb összehangolása a rendszerrel. Összességében azt gondoljuk, sikerült a területen újat alkotnunk, és kutatómunkánk gyakorlati megvalósítása, valamint az előttünk álló további irányok kidolgozása által hozzájárulhatunk a szoftverergonómia fontos és aktuális területének előmozdításához.

Irodalomjegyzék

[1] B. Hadházi-Boros, ”An Overview on Software Ergonomy,” in Proceedings of the 8th International Conference on Applied Informatics Eger, Hungary, vol. 2, pp. 323–329, January 2010.

[2] Kieras, D. E. (1999). A guide to GOMS model usability evaluation using GOMSL and GLEAN3. *University of Michigan*, (313).

[3] C. Bouchard, et al., „Building a design ontology based on the conjoint trends analysis.” in 3rd Virtual International Conference on Innovative Production Machines and Systems, Cardiff. 2007.

[4] MacLean, A., Young, R. M., Bellotti, V. M., & Moran, T. P. (1991). Questions, options, and criteria: Elements of design space analysis. *Human–computer interaction*, *6*(3-4), 201-250.

[5] L. Izsó, M. Antalovits , Bevezetés az információ-ergonómiába, Budapest Műszaki Egyetem, 2000

[6] Melody, W. H. (1999). Telecom reform: progress and prospects.Telecommunications Policy, 23(1), 7-34.

[7] Baranyi, P., & Csapo, A. (2012). Definition and synergies of cognitive infocommunications. *Acta Polytechnica Hungarica*, *9*(1), 67-83..

[8] Sallai, G. (2012). The cradle of cognitive infocommunications. *Acta Polytechnica Hungarica*, *9*(1), 171-181.

[9] Vidal, J. J. (1973). Toward direct brain-computer communication. *Annual review of Biophysics and Bioengineering*, *2*(1), 157-180.

[10] Grimes, D., Tan, D. S., Hudson, S. E., Shenoy, P., & Rao, R. P. (2008, April). Feasibility and pragmatics of classifying working memory load with an electroencephalograph. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 835-844). ACM.

[11] Moon, B. S., Lee, H. C., Lee, Y. H., Park, J. C., Oh, I. S., & Lee, J. W. (2002). Fuzzy systems to process ECG and EEG signals for quantification of the mental workload. *Information Sciences*, *142*(1), 23-35.

[12] Mantri, S., Dukare, V., Yeole, S., Patil, D., & Wadhai, V. M. (2013). A Survey: Fundamental of EEG. *International Journal*, *1*(4).

[13] Hogervorst, M. A., Brouwer, A. M., & van Erp, J. B. (2014). Combining and comparing EEG, peripheral physiology and eye-related measures for the assessment of mental workload. *Frontiers in neuroscience*, *8*.

[14] Fink, A., Grabner, R. H., Neuper, C., & Neubauer, A. C. (2005). EEG alpha band dissociation with increasing task demands. *Cognitive Brain Research*,*24*(2), 252-259.

[15] E. Wascher, et al. (2014). Frontal theta activity reflects distinct aspects of mental fatigue. *Biological psychology*, *96*, 57-65.

[16] M. Stikic, et al. (2014). Modeling temporal sequences of cognitive state changes based on a combination of EEG-engagement, EEG-workload, and heart rate metrics. *Frontiers in neuroscience*, *8*.

[17] Oliveira, I., & Guimarães, N. (2013, March). A tool for mental workload evaluation and adaptation. In *Proceedings of the 4th Augmented Human International Conference* (pp. 138-141). ACM.

[18] A szívfrekvenciavariabilitás (HRV) alapelvei http://r-medical.hu/szivvizsgalat

[19] Wikipedia: *The Eye Tribe*, http://en.wikipedia.org/wiki/The_Eye_Tribe (revision 16:11 20 January 2015)

[20] Marshall, S. P. (2002). The index of cognitive activity: Measuring cognitive workload. In *Human factors and power plants, 2002. proceedings of the 2002 IEEE 7th conference on* (pp. 7-5). IEEE.

[21] Forstner, B., Szegletes, L., Angeli, R., & Fekete, A. (2013, December). A general framework for innovative mobile biofeedback based educational games. In *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on* (pp. 775-778). IEEE.

[22] L. Szegletes, B. Forstner, ”Reusable framework for the development of adaptive games.” in: *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on. IEEE, 2013.*, pp. 601-606.

[23] BME AUT: *InnoLearn játékfejlesztés gyorstalpaló* (2015 ápr.)

[24] *Zephyr Technology Hxm Bluetooth API Guide*, Zephyr Technology Corporation (Version 1.7 – Date: 2010.07.22.)

[25] *Android Development Guide for ThinkGear*, Neurosky (2011. October 16.)

[26] Levendovszky, J., Jereb, L., Elek, Z., & Vesztergombi, G. (2002). Adaptive statistical algorithms in network reliability analysis. *Performance Evaluation*,*48*(1), 225-236.

[27] National Istruments: *LabVIEW grafikus fejlesztői környezet leírása*, http://www.ni.com/ (2010. nov.)

[28] Fowler, M.: *UML Distilled*, 3rd edition, ISBN 0-321-19368-7, Addison-Wesley, 2004

[29] Wikipedia: *Evaluation strategy*, http://en.wikipedia.org/wiki/Evaluation_strategy (revision 18:11, 31 July 2012)

[30] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Applied statistics*, 100-108.

[31] Skinner, W. (2006). User-Centered Design Evaluation by Application of Biofeedback Technology (Doctoral dissertation).

[32] Palinko, O., Kun, A. L., Shyrokov, A., & Heeman, P. (2010, March). Estimating cognitive load using remote eye tracking in a driving simulator. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications* (pp. 141-144). ACM.

Ábrajegyzék