Mobil eszközökön futó illesztőprogram és arra épülő alkalmazás fejlesztése non-invazív agyi szenzorrendszerhez

TDK dolgozat

Készítette: Bársony Georgina Eszter
Egészségügyi mérnök Msc (2. évfolyam)

Konzulensek: Dr. Márton Gergely
tudományos munkatárs (ELKH-TTH)

Dr. Szlávecz Ákos
egyetemi docens (BME-IIT)

2020 BUDAPEST

Összefoglaló

Az Eötvös Loránd Kutatóhálózat Természettudományi Kutatóközpontja (ELKH TTK) és a MindRove Kft. együttműködésével fejlesztés alatt álló neuro-fejpánt non-invazív módon, a fejbőrre helyezett vezető szövet elektródok segítségével méri a felhasználók agyának elektromos aktivitását. Az elektroencefalográfiai (EEG) szenzorrendszer elsősorban az agykéreg centrális és parietális területeiről származó potenciálváltozásokat méri. Lehetséges felhasználási területeihez tartoznak az elektronikus eszközök gondolattal, például elképzelt mozgásokkal történő irányítása, a neurológiai visszacsatolás (NeuroFeedback). A neurális interfész jelenleg Windows és macOS operációs rendszereket futtató számítógépekkel kompatibilis, előnyös lenne azonban, ha mobil eszközökre – mobiltelefonokra, táblagépekre – is lehetne a neuro-fejpántra épülő applikációkat fejleszteni, lehetővé téve ezzel, hogy a felhasználók szélesebb körben alkalmazhassák azt.

Elsődleges célom egy olyan szoftveres infrastruktúra kialakítása Android környezetben, amely a MindRove neuro-fejpánt által mért agyi jelek WiFi kapcsolaton keresztül történő fogadására, tárolására, feldolgozására és megjelenítésére alkalmas, mely felhasználható a felhasználó agytevékenységének visszajelzésére, azaz neurológiai visszacsatolásra. Ezen kívül alapként szolgálhat összetettebb mobil alkalmazások (pl. elektromos tolószékek, robotkezek vezérlése) és offline adatelemzést igénylő kutatások (pl. alvásmonitorozás, stressz faktorok detektálása) fejlesztéséhez.

A szoftverrendszer az Android Studio fejlesztői környezetben készült, Java nyelven. Az adatok fogadása WiFi csatornán, UDP protokoll szerint történik. A jelfeldolgozó szoftvermodul többek között az EEG jelek esetében releváns, különböző frekvenciasávokban domináns agyhullámok (alfa, béta, gamma, delta, théta) valós idejű mérését teszi lehetővé.

Abstract

In cooperation with the Natural Science Research Center of the Eötvös Loránd Research Network (ELKH TTK) and MindRove Ltd. a neuro-headband is under development. Non-invasively, it measures the electrical activity of users' brain using conductive tissue electrodes placed on the scalp.The electroencephalographic (EEG) sensor system primarily measures potential changes from the central and parietal areas of the cerebral cortex, and its potential applications include controlling electronic devices with thought, such as imaginary movements, and neurological feedback (NeuroFeedback). The neural interface is currently compatible with computers running Windows and macOS operating systems, but it would be beneficial to be able to develop applications based on the neuro-headband for mobile devices — mobile phones, tablets — allowing users to use it more widely.

My primary goal is to create a software infrastructure in an Android environment that is capable of receiving, storing, processing and displaying brain signals measured by the MindRove neuro headband via WiFi connection, which can be used to provide feedback on a user's brain activity, called neurological feedback. It can be the basis for the development of more complex mobile applications (e.g. electric wheelchairs, robot hand control) and to develop research that requires offline data analysis (e.g., sleep monitoring, stress factor detection).

The software system is built in the Android Studio development environment in Java. The data is received via Wifi channel, according to UDP protocol. The signal processing software module is relevant for EEG signals, allows real-time measurement of dominant brain waves (alpha, beta, gamma, delta, theta) in different frequency bands.

Tartalomjegyzék

1. Bevezetés

A TDK dolgozatom témájának mobil eszközökön futó illesztőprogram és arra épülő alkalmazások fejlesztését választottam non-invazív agyi szenzorrendszerhez. A szenzorrendszer a MindRove neuro-fejpánt, melyet egy magyar fiatalokból álló csapat fejlesztett ki az Eötvös Lóránd Kutatási Hálózat Természettudományi Kutatóközpont (ELKH-TTK) és a MindRove Kft. közreműködésével. Előnyös lenne, ha ez az interfész asztali, vagy hordozható számítógép nélkül, mobil eszközökön tudna futni, lehetővé téve ezzel, hogy a felhasználók sokkal nagyobb időtartamban és helyszíntől, azok adottságaitól függetlenül alkalmazhassák azt. Ezen kívül amennyiben a fejpánt mobiltelefonokkal, táblagépekkel is használható, úgy tolószékekhez is könnyebben integrálható lesz.

Célom az volt, hogy egy olyan applikációt fejlesszek Android alapú okostelefonokra, amely a fejpánttal kétirányú adatkommunikációt valósít meg Wifi kapcsolaton keresztül. Ezen kívül a szenzorrendszer által mért jeleket átalakítja és több módon, valós idejű diagram formájában jeleníti meg azokat, illetve lehetővé teszi azok mentését a telefon külső tárhelyébe.

Feladatomat a szakirodalmak elemzésével kezdtem, mely során megismerkedtem az elektroenkefalográfia, valamint az agy-gép interfész jelentésével, működésével és alkalmazási területeivel. Ezt követően megterveztem a mobilapplikációt és annak tesztelését. Ezt a fejpánt és az applikáció közötti kommunikáció megvalósítása, majd a szoftver fejlesztése követte. Az elkészült szoftvert több szempont alapján és több eszközön is teszteltem. Legvégül pedig a továbbfejlesztési és alkalmazási lehetőségekről gyűjtöttem információt.

2. Az elektroenkefalográfia

Rövidítve EEG-nek nevezzük az elektroenkefalográfiás vizsgálatot, amely az agyi tevékenységet kísérő elektromágneses változásokat leképező vizsgáló módszer. Az EEG hullámok az invazív vagy non-invazív módon elhelyezett elektródák által közvetített potenciálkülönbség változásokból speciális átalakítással nyert vizualizált jelek. Ezek a jelek az agy szürkeállományában lévő sejtcsoportok funkcióváltozásait kísérő ingadozó és gyorsan változó elektromágneses-térből adódnak [1].

Az EEG keletkezésének pontos mechanizmusa a mai napig nem ismert. Valószínűsíthető, hogy több ezer neuron összesített elektromos aktivitását tükrözi, melyek az agykéreg felszíni részében

helyezkednek el. A mélyebb struktúrák aktivitásai a szignál gyengülése miatt nem idézhetnek elő a fejbőrről elvezethető mikrovoltos nagyságrendű jelet. Tekintettel az EEG fokozatos jellegű változásaira, inkább a piramissejtek dendritnyúlványain lejátszódó küszöb alatti depolarizációra gondolhatunk, nem pedig a gyors akciós potenciálokra. Az agyban keletkező bioelektromos jelek, a neuronok elektromos aktivitása, az agykérgi modulok kisülései, melyek kicsiny elektromágneses mezőket gerjesztenek. Ez azt jelenti, hogy az agykérgi aktivitás során feszültségváltakozások történnek. A sejtben protonok, elektronok, ionok áramlása elektromos áramot jelent. Az áramot a sejt aktív enzimköreinek ezrei hozzák létre, többek között a sejtmembránon tapasztalható feszültségkülönbséggel [2] [3].

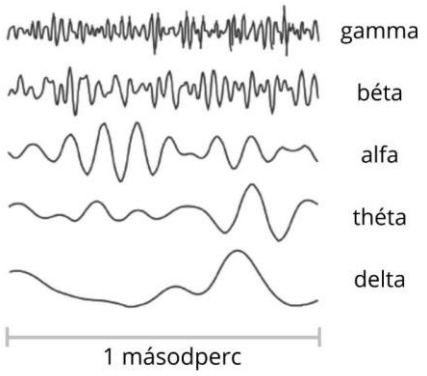
A testünk egyik legnagyobb szerkezete az idegrendszer, amely neuronhálózatok csoportja. Ez a kapcsolatrendszer az egész testünket átszövi. Az idegrendszert alkotó idegsejtek, azaz a neuronok az inger felvételére, tovavezetésére és az ingerület másik sejtnek való átadására specializálódtak. Képesek egymással kommunikálni, és ebből következően az agyból érkező ingerek közvetítésére. Magát az ingerület átvitelt és a sejtek közötti kontakthelyeket, amelyeken át az ingerület az egyik sejtről a másik sejtre terjed tovább, szinapszisnak nevezzük. A szinapszis idegsejtek közötti információáramlást tesz lehetővé. Ezt nagyobb részben az egyik neuron sejtteste (axon) hozza létre a másik neuron idegvégződésével (dendrit). A szinapszis lehet kémiai vagy elektromos, ám az emberben leggyakoribb a kémiai szinapszis. Ekkor az ingerület átadásához azt átvivő anyagokra, azaz neurotranszmitterek felszabadulására van szükség. Ilyenkor az ingerület egy irányba terjedhet, azonban a neurotranszmitterek mindkét irányban haladhatnak. A visszafelé irányuló transzmissziónak általában szabályozó szerepe van. A neuronok rendelkeznek egy alapvető potenciállal, amely a külső, illetve belső sejtmembrán közötti feszültségkülönbséggel magyarázható. Ezt nevezzük nyugalmi membrán- potenciálnak. Az idegsejteknek ezek a potenciálváltozásai önmagukban nem észlelhetők, azonban több együttesen már mérhető. A neuronok akciós és gátló tevékenységei egymás hatását erősítve létrehozhatnak egy EEG-görbét. Sok funkcionális változásra érzékeny az EEG. Ilyenek lehetnek a szemmozgás vagy izommozgás. Ezek az úgynevezett műtermékei az EEG-nek, amiket célszerű kiszűrni. A fejbőrön elhelyezett elektródák segítségével lehet mérni a feszültséget és abból alakul ki az EEG jel [4] [5].

Az EEG jel csak az agytevékenység töredékéről nyújt információt, mégis egyedülálló abban, hogy non-invazív módon is jellemezhető vele az agy globális és regionális funkcionális állapota. Ezáltal a módszer betekintést enged fiziológiás folyamatok történéseibe, továbbá bizonyos kórállapotokban mással nem helyettesíthető, vagy célszerű kiegészítő diagnosztikai eljárásként alkalmazható [6].

Az EEG jel egy komplex, több komponensből álló periodikus görbe. A jeleket 3 tulajdonsággal jellemezhetjük:

- Frekvencia - az oszcilláció sebességét mutatja, mértékegysége a Hertz (Hz), amely a másodpercenkénti oszcillációk számára utal.
- Teljesítmény - a frekvenciasávban jellemző energiamennyiség négyzetes amplitúdóban kifejezve.
- Fázis - több generátor (neuronok) szinkronizálásának összege [7].

Az EEG jelek némileg függenek az egyedi tényezőktől, az inger tulajdonságaitól és a belső állapottól. A szakirodalom a jeleket meghatározott frekvenciatartományok vagy frekvenciasávok szerint osztályozza:



1. ábra Az EEG jelek frekvenciasáv szerinti osztályozása [7]

Delta (0,5-4 Hz): A leglassabb és a legmagasabb amplitúdóval rendelkező agyhullám, amely általában alváskor jelenik meg, így az alvás mélységéről, álmatlan alvásról, vagy eszméletlen állapotról kaphatunk információt általa. Ezenkívül a Parkinson kór, a skizofrénia, a demencia és az alkoholizmus fennállásának lehetőségét is segíthet feltárni. A delta frekvenciák általában a jobb agyféltekén erősebbek és forrásuk a talamuszban található.

Théta (4-8 Hz): Fókuszálás, odafigyelés, információ felvétel és feldolgozás, visszaemlékezés, azaz valamilyen mentális művelet során jelentkezik a théta aktivitás. Minél nagyobb a feladat nehézsége, annál nagyobb szerepet kapnak a jelek. Ez az oka annak, hogy a théta általában azon agyi folyamatokhoz kapcsolódik, amelyek a mentális terhelés vagy memóriahasználat alapjául szolgálnak. Théta hullámok az egész agykéregből érkeznek. Tipikus tanulmányok a thétával: labirintusban való tájékozódás, betűk, számok, ikonok listájára való visszaemlékezés.

Alfa (8-13 Hz): Ez a hullám a mélyagyi thalamikus, valamint a parietális és a occipitális agykérgi régiókban képződik. Több funkcionális korrelátorral rendelkezik, amelyek érzékszervi, motor- és memóriafunkciókat tükröznek. A szellemi és fizikai pihenés során az alfa-sáv teljesítménye megnő (csukott szem). Ezzel szemben az alfa-erő csökken, vagy elnyomódik a mentális, vagy testi tevékenység során (nyitott szem). Azt is mondhatjuk, hogy az alfa hullám azt jelzi, hogy az agy felveszi az információt különböző érzékekből a figyelemfelkeltő erőforrások összehangolásával

azokra a tényekre összpontosítva, amelyek igazán fontosak abban a pillanatban. Gyakran használják arra, hogy a relaxáció, meditáció szintjét mérjék vele.

Béta (13-30 Hz): Mind az agy hátsó, mind az első régióiban keletkeznek béta hullámok. A központi kéreg felett a béta hullám teljesítménye megváltozik, ha mozgásokat tervezünk, vagy hajtunk végre. Érdekes, hogy az ún. béta deszinkronizáció akkor is kiemelkedő, amikor mások testmozgásait megfigyeljük, mert olyankor az agyunk látszólag utánozza azokat a mozgásokat. Az agyhullám jellemzően a vizsgálati alanyok elképzelt és megvalósított mozgása során tanulmányozgató.

Gamma (30 Hz felett): Egyelőre még nem tisztázott, hogy hol keletkeznek pontosan és mit mutatnak meg ezek az oszcillációk. Egyes kutatások szerint a thétához hasonló hordozófrekvenciaként szolgál, a tárgyak különböző érzékszervi benyomásaihoz köthető. Mások azt állítják, hogy a gamma-frekvencia más neurális folyamatok melléktermék, mint például a szemmozgások és a mikro-saccádok, és ezért nem mutatják a kognitív feldolgozást [7].

Egy fizikai jelenség frekvenciája az adott időintervallumba eső jelváltások száma. Fourier tétele szerint a jelenséget leíró komplex periodikus jel (időfüggvény) diszkrét frekvenciájú szinuszos jelek összegére bontható. A frekvencia alapú jelfeldolgozás során tehát az időben megadott amplitúdó jelet áttranszformáljuk frekvencia alapúra. Az így kapott vonalas spektrumképben különféle frekvenciájú komponensek különíthetők el, adott amplitúdó értékekkel. A frekvenciaspektrum az egyenkomponens mellett alapharmonikusból és felharmonikusokból áll. Többféle algoritmus létezik a frekvenciatartományra történő adattranszformációra, mint például a diszkrét Fourier transzformáció (DFT), a gyors Fourier transzformáció (FFT), az ablakozott Fourier transzformáció, vagy a Wavelet elemzés. Ezek a spektrumanalízis módszerei, mely tehát annak kimutatására szolgál, hogy milyen frekvenciájú összetevők dominálnak az adott jelben. A transzformáció eredménye megadja a jel amplitúdóját a frekvencia függvényében. Egyéb jellemzően vizsgált paraméterek az átlagfrekvencia, illetve a frekvenciaterjedelem [8].

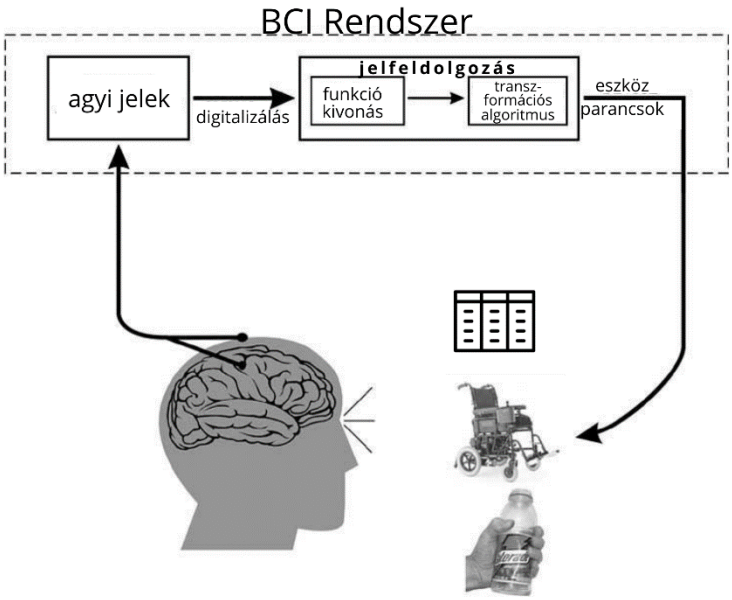
2.1 Agy-gép interfészek

Az agy-gép interfész (angolul Brain-Computer Interface, később BCI) kifejezés olyan eszközöket foglal magába, amelyek lehetővé teszik a felhasználó számára, hogy agyi aktivitás révén lépjenek kapcsolatba a számítógéppel. Ennek alapja leggyakrabban az elektronenkefalográfia (később EEG) [9].

Manapság egyre nagyobb népszerűségnek örvendenek az ilyen eszközök és ennek több oka is van. Ezek a felhasználói parancsot az EEG jelből ismerik fel, és ideális esetben annak megfelelően reagálnak. Egy egyszerű példa, hogy a számítógép képernyőjén lévő nyíl pozícióját a felhasználó

úgy tudja mozgatni, hogy csak elképzeli, hogy a jobb vagy a bal kezét mozgatja. A képzeletbeli folyamat során az agyhullámokat a felhasználó parancsainak felismerésére használja. Komplexebb feladatokat is elláthatnak, mint például olyan betegek esetében, akik elvesztették valamely végtagjukat vagy annak funkcionalitását. Nekik az EEG által vezérelt mesterséges végtagok hatalmas segítséget nyújthatnak a mindennapokban [10]. Sajnos a technológia még nem érte el azt az információátviteli sebességet és megbízhatóságot, amellyel akár csak megközelítő lenne egy természetes végtaghoz hasonló vezérlés.

A BCI rendszerek mérik az agyi tevékenység sajátosságait és átalakítják azokat eszközvezérlő jelekké. Tulajdonképpen egy kommunikációs csatornát hoznak létre az agy számára, amely úgy működik, mint az izmok, vagy a perifériás idegek vezérlése. A BCI művelet azonban egy olyan új készség, amely nem a megfelelő izomszabályozásból áll, hanem az EEG aktivitás szabályozásából. A BCI működése attól is függ, hogy a felhasználó hogyan kódolja a parancsait az EEG jellemzőiben [11] [12].



2. ábra Agy-gép interfész rendszer általános séma [11]

A BCI működése két adaptív vezérlőtől függ: az egyik a felhasználó agya, amely a BCI rendszer által mért aktivitást hozza létre, a másik pedig maga a rendszer, amely ezt a tevékenységet átalakítja konkrét paranccsá. Mint minden kommunikációs vagy vezérlő rendszernek, a BCI-nak is van bemenete, kimenete, valamint fordító algoritmusa, amely a bemenetet kimenetté alakítja. A bemenet az agyi tevékenység sajátossága, amely magában foglalja a lassú kortikális potenciált, a P300 hullám által kiváltott potenciált, a fejbőrön érzékelt szenzormotoros ritmusokat, vagy a kéregben a neurális akciós potenciálokat. Ezek a sajátosságok jellemezhetők a frekvencia, vagy az idő függvényében.

Minden BCI rendszer egy meghatározott algoritmust használ a bemenet lefordítására. Ez tartalmazhat lineáris vagy nemlineáris egyenleteket, neurális hálózatot vagy más módszereket. Magában foglalja azok paramétereinek folyamatos hozzáigazítását a felhasználó által biztosított inputhoz. A BCI kimenetek pedig lehetnek például a kurzor mozgatása, betű vagy ikon kiválasztása, vagy egy eszköz vezérlésének egyéb formája. A visszajelzést a felhasználó és a BCI rendszer is alkalmazhatja a kommunikáció optimalizálására.

A bemeneten, kimeneten és fordítási algoritmuson kívül minden BCI rendelkezik még más megkülönböztető tulajdonságokkal is. Ezek lehetnek a kimenet és bemenet mechanizmusa, a válaszidő, a sebesség, a pontosság (vagy azok kombinációja, az információátviteli sebesség). Ezen kívül a felhasználó szükséges képzettségi szintje, a felhasználó populáció, az alkalmazások, a szenzorokra előírt korlátozások vagy akár annak a követelménye, hogy a felhasználó mozdulatlan maradjon a használata közben [11] [12].

A jelenleg kapható non-invazív BCI eszközöket két csoportra osztottam kinézetük és funkciójuk szerint. Az egyik csoportba sorolhatók a könnyen kezelhető, bárki számára elérhető fejpántok. Ezek az eszközök a mért adatokat nem az egészségi állapot, vagy betegségek felderítésére használatosak, inkább gondolattal vezérelhető játékok, alkalmazások megvalósítására, valamint a meditációs szint mérésére, összességében szabadidős tevékenységre alkalmazhatók. Általában kisméretűek és egy fejhallgatóhoz hasonlítanak. Egyszerű és bárki számára könnyen elsajátítható a használatuk. Ezekben az elektródák száma is 2 és 5 között mozog. A mért és feldolgozott jeleket könnyű értelmezni, nem kell hozzá orvosi, mérnöki szakképzettség. Ilyenek például az Emotiv – Insight, a Muse – Original Muse, vagy a NeuroSky - MindWave mobile 2 eszközök.

A másik csoportba pedig a nagyméretű, sapkaszerű fejpántok tartoznak. melyek jellemzően magasabb árkategóriát képviselnek. A motoros funkciókhoz tartozó potenciálok mérésében azonban sokkal megbízhatóbbak. Ilyenek például az Open BCI - Ultracortex Mark IV, vagy a g.tec – g.Nautilus.

Az általam is használt MindRove kifejlesztésének célja az volt, hogy a fentiekben bemutatott két csoport pozitív tulajdonságait ötvözze a motoros agykérgi vizsgálatokhoz. Tehát a motoros funkciókhoz tartozó potenciálok mérésében kimagasló teljesítményt nyújtson, emellett könnyű használatot biztosítson, kényelmes legyen a viselése és megfizethető is legyen.

2.2 EEG mérésen alapuló szoftverek alkalmazásának lehetőségei

Az EEG mérésen alapuló szoftverek alkalmazási lehetőségei közé tartozik a neurofeedback terápia. Az 1960-as és 1970-es években felfedezték, hogy lehetséges az agyhullámok képzésének tanulása,

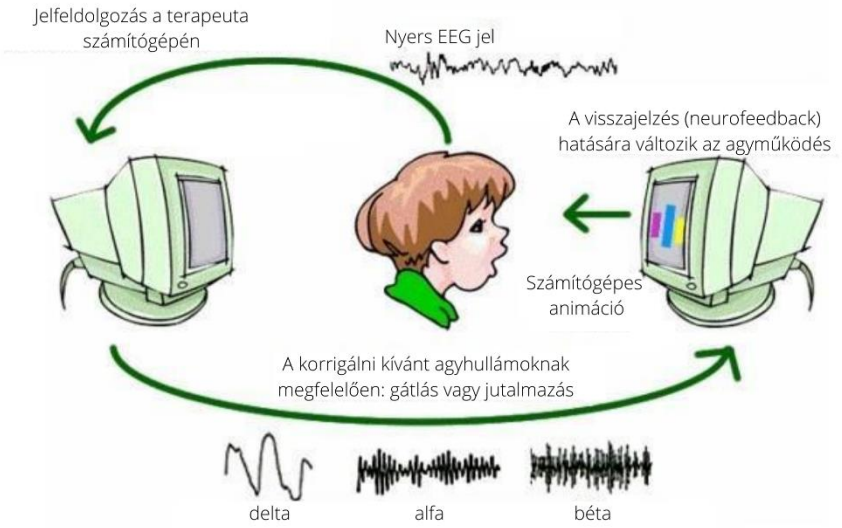
azok megváltoztatása. Kezdetben ezt az epilepsziás betegeknek való segítésre, vagy az alfa aktivitás növelésére használták, amely az optimális relaxációs szint elérését segítette. Ezt az agyhullám tréninget nevezik neurofeedbacknek (vagy EEG biofeedbacknek).

Az agyunk működése nagyon komplex folyamat. A 2. fejezetben bemutatott hullámok valamilyen szinten mindig jelen vannak az agyunk különböző részein. A delta hullámok akkor jelenek meg, amikor az agy éppen táplálékot vesz fel, de összefüggésbe hozható a tanulási nehézségekkel is. Amikor valaki szorong vagy feszült, akkor magas frekvenciás béta hullámok jelennek meg az agy különböző részein, de egyes esetekben ez összefügghet nem megfelelő alfa aktivitással az elülső területeken. A figyelemhiányos, hiperaktivitási rendellenességgel rendelkező, fejsérülést, stroke-ot átélt személyeknél, valamint az epilepsziában vagy fejlődési fogyatékosságban szenvedőknél is általában a lassú (többnyire théta, de alfa is előfordulhat) hullámok túlzottan jelen vannak. Azonban, ha az agy elülső részeiben túl sok lassú hullám van jelen, akkor nehéz lesz irányítani a viselkedést, az érzelmeket és a figyelmet. Az ilyen embereknek általában problémájuk van a koncentrációval, a memóriával, a hangulatuk irányításával, vagy a hiperaktivitással. Gondjuk lehet továbbá az összepontosítással és csökkent intellektuális hatékonyságot mutatnak.

Kutatások igazolták, hogy heterogenitás fedezhető fel az EEG mintázatban a különböző diagnosztikai feltételek mellett, például figyelemhiányos-, hiperaktív-, szorongásos-, vagy kényszerbetegségekben szenvedők esetében. A diagnosztikát azonban bonyolultabbá teheti, hogy néha ezek a betegségek nem csak önmagukban vannak jelen egy adott páciensnél. Ezért a neurofeedback terápia megkezdése előtt személyre szabottan kell meghatározni, hogy melyek azok az EEG frekvenciák, amelyek túlzottan, vagy kevéssé vannak jelen, esetleg probléma van a feldolgozási sebességben vagy koherenciában, és ha igen, akkor az agy melyik részein. Ez lehetővé teszi a páciens egyénre szabott kezelését [13].

A neurofeedback terápiát mindig megelőzi egy előzetes kiértékelés, mivel a terápiának igazodnia kell az egyén egyes agyhullám-mintáinak és tüneteinek sajátos jellemzőihez. Ennek keretein belül feltesznek a páciens kórtörténetére vonatkozó kérdéseket, bizonyos esetekben pszichológia teszteket is elvégeznek. Ezt követően alaposan és pontosan mérik, majd kiértékelik az agyhullámokat. Az eredményeket összevetik egy adatbázissal is, melynek célja, hogy összehasonlítsák az eredményt a hasonló paraméterekkel rendelkező más emberek eredményeivel. Ezáltal meghatározzák a normálistól való eltéréseket. Az agyhullámok mérését nemcsak nyugalomban, hanem különböző tevékenység végzése közben is mérik [13].

Ezek után jön a terápia. Egy átlagos neurofeedback terápia folyamán egy, vagy több elektródot helyeznek a páciens fejbőrére, továbbá egyet, vagy kettőt a fülcimpájára. Utána elektronikus berendezések biztosítják a valós idejű és azonnali visszajelzést az agyi aktivitásról.



3. ábra A neurofeedback terápia sémája [14]

A páciensek általában azért nem tudják befolyásolni az agyhullámaikat, mert nincs tudatosságuk azokkal kapcsolatban. Azonban, ha valós idejűen láthatják azokat vizuálisan, akkor lehetőségük van befolyásolni, és fokozatosan megváltoztatni azokat. Ez a megváltozás kezdetben rövid időtartamú, majd fokozatosan, a folyamatos tréning hatására válik tartóssá. A folyamatos visszajelzés és gyakorlás segítségével a kívánatos agyhullámokat megerősíthetik, míg a nem kívánatosakat gyengíthetik [13].

A tréning során többféle feladat is adható. A leggyakrabban a személy egy filmet néz, vagy zenét hallgat. Amennyiben "jól állítja be" az agyhullámait, jó minőségben élvezheti a filmet, vagy hallgathatja a zenét. Amennyiben azonban nem "dolgozik eléggé", az adás rossz minőségűvé, halkká, zavaros hangúvá változik, tehát élvezhetetlen lesz. Az egyén célja az lesz, hogy jó minőségűvé alakítsa át az ingert. A tréning során a számítógépből semmi parancs, vagy egyéb jel az agyhoz nem érkezik. A gép csak mér és visszajelez, az agyba nem nyúl bele semmilyen módon. A páciens önmaga találja meg agyának azt az állapotát, amikor a feladatot végre tudja hajtani. Néhány tréning során az agy megtanulja ezt az állapotot, és már a hétköznapi tevékenységek során is jól fog működni [14].

A neurofeedback terápiát leggyakrabban az alábbi területeken szokták alkalmazni:

- figyelemhiány
- hiperaktivitás
- autizmusspektrum-zavar (ASD)
- depresszió
- szorongásos zavarok
- kényszerbetegségek

- étkezési zavarok
- poszttraumás stressz szindróma (PTSD)
- függőségek
- epilepszia
- migrén
- krónikus fájdalmak
- stresszterhelés
- csúcsteljesítmény elérése (például sportolóknál) [15]

Ezeken kívül szabadidős tevékenységekre is alkalmazható, EEG mérésen alapuló szoftverek is vannak. Ilyenek a meditációs vagy koncentrációs szint mérésére készült programok, valamint a játékok. Erre példa a NeuroSky MindWave Mobile 2 eszköz és az ahhoz készült mobilapplikációk:

- játékapplikáció: például a „FlappyMind” nevű játék, melyben a felhasználó egy madarat irányít miközben figyelmét arra összpontosítja, hogy ne ütközzön össze a különböző akadályokkal.

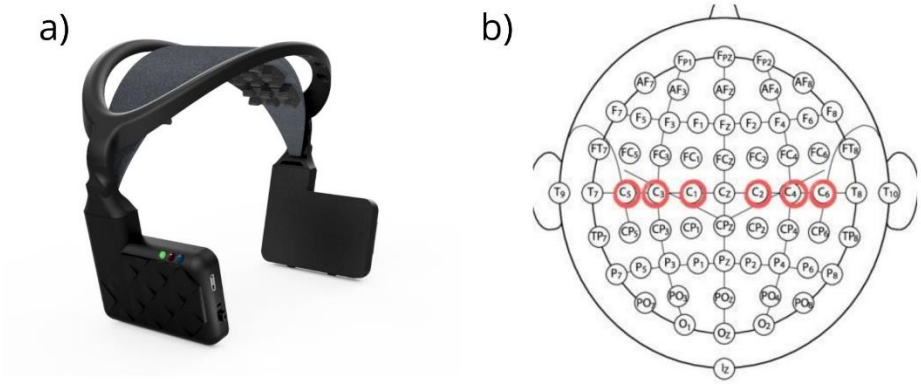
4. ábra FlappyMind játék (forrás: https://play.google.com/store/apps/details?id=com.maf.flappymind)

- koncentráció, meditáció, valamint hangulatkövetést megvalósító applikáció: például az „EEG Meditacion” nevű applikáció, amelyet meditáció közben lehet használni. Az alkalmazás által a felhasználó ellenőrizheti a meditációját, méghozzá úgy, hogy beállít egy küszöbértéket, és ha a meditáció szintje meghaladja azt, hangjelzés formájában értesülhet róla.
- tanulást segítő applikáció: például az „Effective Learner”, amely az agy fókuszszintjének mérésével megmutatja, hogy melyik az az időpont, amikor a felhasználó a leghatékonyabban tanulhat.

Az agy-gép interfészek és szoftvereik alkalmazhatók továbbá fogyatékos sérültek vagy a mozgásukban korlátozott személyek mindennapi tevékenységeinek megkönnyítésére is.

3. A MindRove neuro-fejpánt

A MindRove neuro-fejpánt – amely eszközhöz az applikációt fejlesztettem - egy fülhallgatóhoz hasonló módon helyezhető a fejre, azonban nem a füleken támaszkodva, hanem azok felett, a halánték tájékán. Felső részén a dupla áthidalás biztosítja a tartást és a kényelmet, mely alatt található az elektródákat tároló rész.



5. ábra a) MindRove neuro-fejpánt (forrás: www.mindrove.com) b) az elektródahelyek

Az elektródák a C1, C2, C3, C4, C5 és C6 jelzésű pontokon foglalnak helyet. Ezen kívül rendelkezik még referenciaelektróddal, amely a jobb fül mögé, valamint egy úgynevezett DRL (Driven Right Leg Circuit, magyarul jobbláb áramkör) elektróddal is, ami pedig a bal fül mögé helyezendő. Ezek a fenti ábrán (Lásd:5. ábra) nem láthatóak.

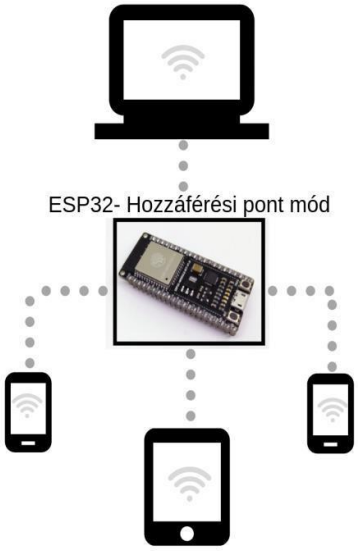
Az eszköz mintavételi frekvenciája 500Hz. Készült hozzá egy szoftver, amely asztali és hordozható számítógépeken, Windows és Linux operációs rendszereken fut. Az eszközben az elektródákon kívül található még egy analóg-digitális átalakító, az ADS1299, valamint egy mikrokontroller is, az ESP-WROOM-32 (később: ESP32). Az ADS1299 egy EEG és más biopotenciálok méréséhez használandó analóg-digitális konverter. Az EEG jel feldolgozásának alapfeltétele, hogy az eredeti analóg jelet digitálissá alakítsuk.

Az ESP32 egy IOT („A dolgok internete”, Internet Of Things) mikrokontroller. Ez egy erőteljes, általános wifi + BT + BLE MCU modul, azaz egy ujjbegynyi chipbe integrálva tartalmazza a processzor mellett a Wifi és a Bluetooth modult is, valamint számos alkalmazást céloz meg az alacsony energiaigényű szenzorhálózatoktól a legigényesebb feladatokig, mint például a hangkódolás, a zenei streaming és MP3 dekódolás [16].

A Bluetooth technológia egy elterjedt megoldás, viszont hátránya, hogy csak jóval kisebb távolságon használható, mint a Wifi, valamint a hálózatok sebessége is lassabb. A kapcsolatot Wifin keresztül valósítottam meg, így a fejpánt és az okostelefon között lévő nagyobb távolság esetén is

kialakítható a kommunikáció. Nagyobb sávszélesség és gyorsabb átvitel valósítható meg általa, így az érkező jelek pontosabb képet adhatnak az agyhullámok változásairól. Emellett pedig egyedüli ezen a téren, hiszen a többi hasonló eszköz WiFi-n keresztül csak számítógépek számára képes adatokat küldeni, mobil eszközöknek nem.

Az ESP32 egyik legkedvezőbb tulajdonsága, hogy WiFi moduljának köszönhetően egyrészt meglévő WiFi hálózathoz kapcsolódhat és webszerverként működhet, másrészt pedig saját hálózatot is felállíthat, lehetővé téve más eszközök számára, hogy közvetlenül csatlakozzanak. Ezáltal az ESP32 három különféle módban működhet: állomás módban (angolul: station mode), hozzáférési pont módban (angolul: acess point mode), valamint mindkettőben egyszerre. Jelen esetben a működés hozzáférési pont módban történik. Hozzáférési pont módban az ESP32 egy saját WiFi hálózatot hoz létre, és egy vagy több állomás osztópontjaként működik, épp úgy, mint egy vezeték nélküli útválasztó (router). Nincs kapcsolata semmilyen vezetékes hálózattal. Ebben a módban a hozzá csatlakoztatható állomások maximális száma öt lehet. A hozzáférési pont mód használatának előnye, hogy az adattovábbítás az okos eszköz és a fejpánt között akkor is létrejöhet, amikor nincs a közelben elérhető WiFi hálózat [17].



6. ábra ESP32 hozzáférési pont mód

A MindRove neuro-fejpánt firmware-e az Arduino fejlesztői környezetben készült. Ez a szoftver vezérli az ESP32 modult, lehetővé téve az elektródák által kapott jelek továbbítását más eszközök számára. Ezt a kódot készen kaptam meg. A firmware magában foglal egy DC szűrőt és egy 50Hz-es lyukszűrést. Ebben a környezetben lehetőség van a kiírt adatok soros monitoron és soros plotteren való megtekintésére. Ezt a funkciót használtam ahhoz, hogy megnézzem, milyen adatok kerülnek majd továbbításra. Amennyiben egy eszköz WiFi-n keresztül rácsatlakozik a fejpántra

(azaz a benne lévő WiFi modulra), az automatikusan megkezdi az adatok küldését. A küldött adat mérete 216 byte. Az applikáció fejlesztése szempontjából is hasznos tartalmak a következők:

Byte-ok	Tartalom	Mérési pont
0. - 3.	EEG 1. csatorna	1.
4. - 7.	EEG 2. csatorna	
8. - 11.	EEG 3. csatorna	
12. - 15.	EEG 4. csatorna	
16. - 19.	EEG 5. csatorna	
20. -23.	EEG 6. csatorna	
...		
76. - 79.	Trigger	
...		
104. - 107.	Sorszám	
108. - 111.	EEG 1. csatorna	2.
112. - 115.	EEG 2. csatorna	
116. - 119.	EEG 3. csatorna	
120. - 123.	EEG 4. csatorna	
124. - 127.	EEG 5. csatorna	
128. - 131.	EEG 6. csatorna	
...		
184. - 187.	Trigger	
...		
212. - 215.	Sorszám	

1. táblázat A küldött UDP csomag tartalma (részlet)

Az adatküldés az UDP (User Datagram Protocol, magyarul felhasználói datagram protokoll) protokoll szerint történik, amely összeköttetés kiépítése nélkül IP-datagramokba ágyazott formában továbbítja az adatokat. Olyan szegmenseket használ az átvitelhez, amelyek egy fejrészből, valamint a felhasználói adatokból állnak. A fejrészben található a forrás- és célpont, amely két port a végpontok forrás- és a célgépen belüli azonosítására szolgál. Előnye, hogy egyszerű és gyors, azonban hátránya, hogy nem garantálja a csomag megérkezését, azonban a valós idejű alkalmazásoknál a gyorsaság fontosabb, mint a csomagvesztés kiküszöbölése [18].

4. Tervezés

A feladat megvalósítását tervezéssel kezdtem, amely magában foglalja a fejlesztési körülmények definiálását, az applikáció szoftvertervének összeállítását, valamint az applikáció tesztelésének tervezetét is. Az alábbiakban ezeket mutatom be részletesen.

4.1 Az applikáció tervezése

Az applikáció Android operációs rendszeren fut. Az applikációt az Android Studio fejlesztői környezetben fejlesztem mely egy hivatalos, integrált fejlesztési környezet Android alkalmazások fejlesztéséhez. A kódot Java nyelven írtam. A felhasználó számára is látható felületekhez az XML-t használtam, amely egy leíró programozási nyelv.

4.1.1 Szoftverterv

Első lépésben a követelményeket határoztam meg, azokat is két részre bontottam. Az elsődleges követelmények az alábbiak:

Azonosító	Leírás	Használati eset
R01	Egyirányú adatkommunikáció: neuro-fejpánt → applikáció	Adatok küldése
R02	Menüelrendezés	Menüpont kiválasztása
R02	Beérkező jelek feldolgozása Fast Fourier transzformációval → Frekvenciaosztályok meghatározása	Jelfeldolgozás
R03	Nyers adatok megjelenítése az idő függvényében	Megjelenítés
R04	Frekvenciaosztályok időalapú vonaldiagramos megjelenítése	Megjelenítés
R05	Frekvenciaosztályok és hozzá tartozó amplitúdó értékek hisztogramos megjelenítése	Megjelenítés
R06	Adatok mentése külső állományba	Adatok mentése

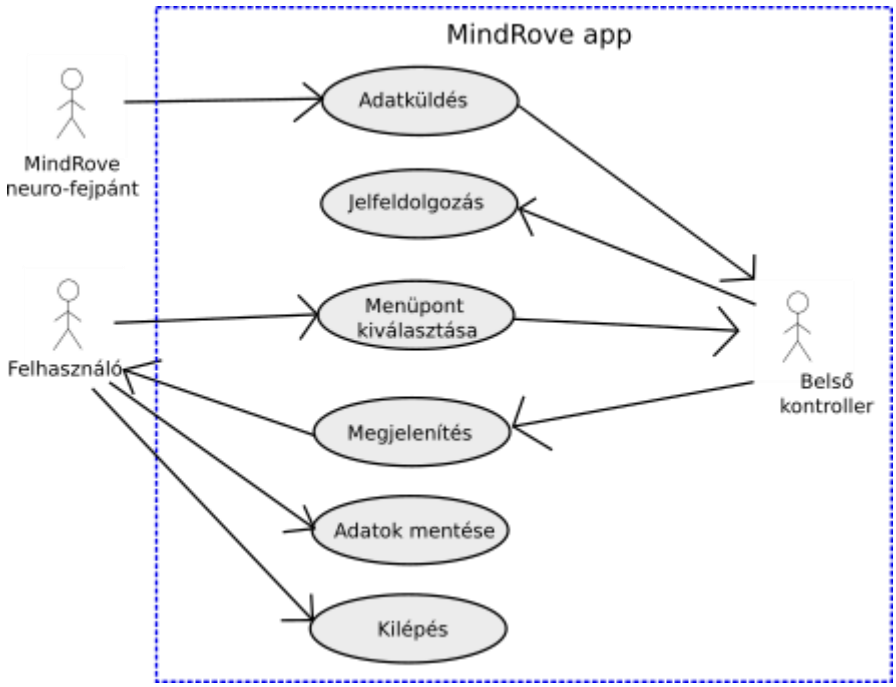
2. táblázat MindRove App elsődleges követelmények

További követelmények, amelyek nem létfontosságúak, hanem a tesztelésnél szükségesek, vagy a használat megkönnyítésére alkalmasak:

Azonosító	Leírás	Használati eset
R07	Kétirányú adatkommunikáció neuro-fejpánt ↔ applikáció	Adatok küldése
R09	Kilépés	Kilépés

3. táblázat MindRove App további követelmények

Ezt követően elkészítettem a használati eset diagramot és a hozzá tartozó leírásokat is.



7. ábra MindRove App use-case diagram

Cím	Adatok küldése
Leírás	A MindRove neuro-fejpánt adatokat továbbít az applikációnak, illetve fogad attól
Aktorok	MindRove neuro-fejpánt
Forgató-könyv	A MindRove neuro-fejpánt 500Hz-el UDP adatcsomagokat továbbít WiFi csatornán keresztül, amelyeket az applikáció fogad. Az applikáció képes UDP adatcsomagot küldeni a neuro-fejpántnak, az pedig, ha sikeresen fogadta, a következő küldött csomagban jelzi.

4. táblázat Adatok küldése használati eset leírása

Cím	Jelfeldolgozás
Leírás	A beérkező nyers jelek mozgóablakos Fast Fourier transzformációja és a frekvenciatartományok, valamint azok átlagainak kiszámítása.
Aktorok	Belső kontroller
Forgató-könyv	A beérkező adatokon 2 másodperces időablakokban Fast Fourier Transzformáció végzése, majd az ablak folyamatos csúsztatása 100 milliszekundumonként. A transzformált, komplex számokból álló adatsor abszolút értékének számítása, majd az egyes frekvenciaosztályokhoz tartozó amplitúdó értékek kiválasztása és átlagolása, végül pedig ezen átlagok csatornás átlagolása.

5. táblázat Jelfeldolgozás használati eset leírása

Cím	Menüpont kiválasztása
Leírás	A felhasználó kiválaszthatja, hogy milyen módon szeretné megtekinteni a MindRove neurofejpánt által továbbított jeleket.
Aktorok	Felhasználó
Forgató-könyv	1. Nyers adatok idő alapú, valós idejű megjelenítése 2. Transzformált adatok idő alapú, valós idejű megjelenítése 3. Transzformált adatok hisztogramos megjelenítése 4. Alfa teszt, adatmentés

6. táblázat Menüpont kiválasztása használati eset leírása

Cím	Megjelenítés
Leírás	Az adatok valós idejű megjelenítése.
Aktorok	Belső kontroller
Forgató-könyv	A felhasználó által kiválasztott menüponthoz tartozó valós idejű diagram és a hozzá tartozó funkciók megjelenítése a felhasználói felületen.

7. táblázat Megjelenítés használati eset leírása

Cím	Adatok mentése
Leírás	Az adatok mentése a mobil eszköz külső állományába.
Aktorok	Felhasználó
Forgató-könyv	A felhasználó az Alfa teszt menüpontban található gomb megnyomásával megkezdi a nyers és a transzformált adatok txt fájlba való mentését. A gomb ismételten történő megnyomásával befejezhető a mentés folyamata.

8. táblázat Adatok mentése használati eset leírása

Cím	Kilépés
Leírás	Az applikáció bezárása.
Aktorok	Felhasználó
Forgató-könyv	A kilépés gombot megnyomva felugrik egy párbeszédablak, melyben meg kell erősíteni a kilépési szándékot. Ha ez megtörtént, az alkalmazás bezáródik.

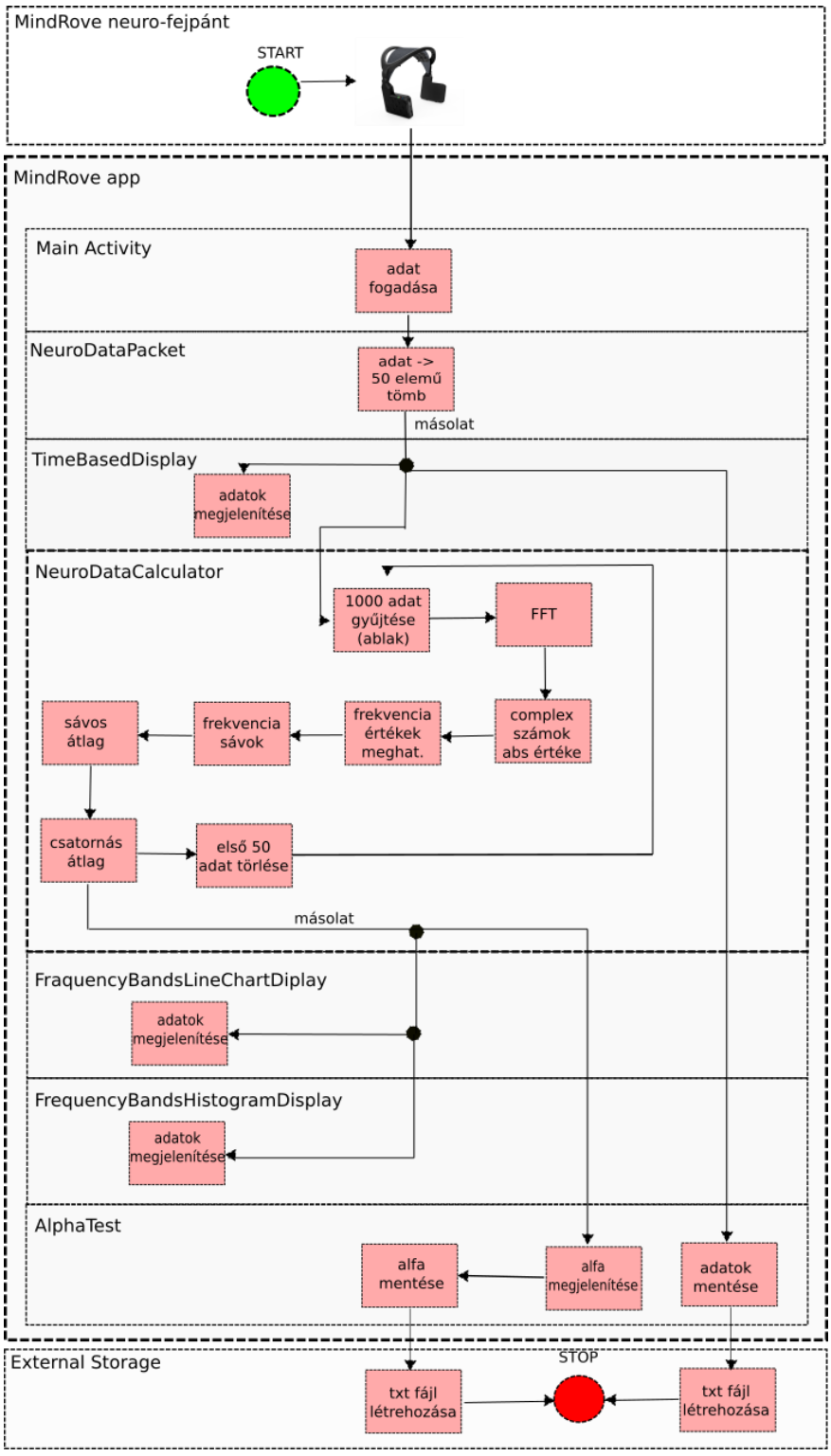
9. táblázat Kilépés használati eset leírása

Az osztályok leírása pedig a következő:

- MainActivity osztály: Itt történik az adatkommunikáció és a menürendszer vezérlése. Az UDP adatcsomagok fogadását illetve küldését a runUDP() függvény valósítja meg, amely külön szálon fut.
- NeuroDataCalculator osztály: Ebben az osztályban történik a jelfeldolgozás, azaz a mozgóablakos Fast Fourier transzformáció, illetve a frekvenciatartományok és azokhoz tartozó átlagok számítása. Szintén külön szálon fut.

- TimeBasedDipslay, FrequencyBandsLineChart, FrequnecyBandssHistogram osztályok: Ezekben az osztályokban az agyi jelek valós idejű diagramos megjelenítése valósul meg.
- AlphaTest osztály: Az adatok külső állományba való mentését végrehajtó osztály.

A program várt működésének folyamatábrája a következő:



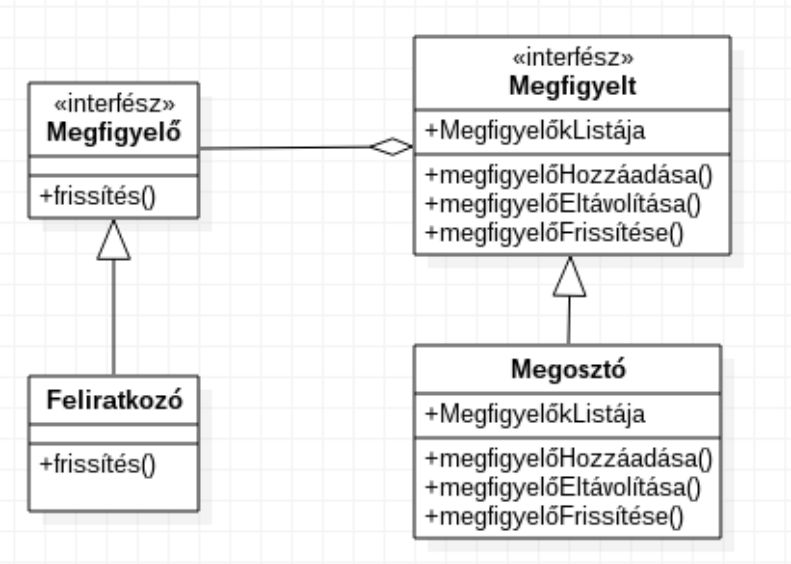
8. ábra MindRove App működési terv folyamatábra

4.1.2 Megfigyelő programtervezési minta

Ahhoz, hogy többféle módon is megjelenítésre kerüljenek a jelek, arra volt szükségem, hogy az egyes megjelenítő mechanizmusok megkapják azokat. Nem lehetséges azonban, hogy egy applikáción belül az adatok több helyen is fogadásra kerüljenek ugyanazon a porton. Így elkerülhetetlen volt egy olyan technika használata, amely lehetővé teszi, hogy az adatok egy helyen kerüljenek fogadásra, majd a beérkezett adatokat több osztály is megkapja/elérje, hogy fel tudja dolgozni, meg tudja jeleníteni azokat. Erre alkalmas technika lehet a „ViewModel” vagy az „EventBus” is, azonban a legcélravezetőbb és legegyszerűbb megoldásnak az interfészek használatát találtam, azaz a megfigyelő minta alkalmazását. Az interfész egy absztrakt osztály, amelyet a kapcsolódó metódusok csoportosítására használnak, de objektumok létrehozására nem használhatók. Nincs body-juk, a body-t az interfészt implementáló osztály fogja megadni [19].

Az úgynevezett megfigyelő programtervezési minta (angolul Observer Pattern vagy Publisher-Subscriber Model) egy olyan szoftvertervezési minta, amelyben egy objektum listát vezet a megfigyelőiről és automatikusan értesíti őket bármilyen állapotváltozásról, többnyire valamely metódusok meghívásán keresztül [20] [21].

A minta két alaposztályból áll, az egyik az úgynevezett Megfigyelő (angolul: Observer), a másik pedig az úgynevezett Megfigyelt osztály (angolul: Subject, azaz lefordítva Tárgy, ami a megfigyelés tárgyára utaló elnevezés.)



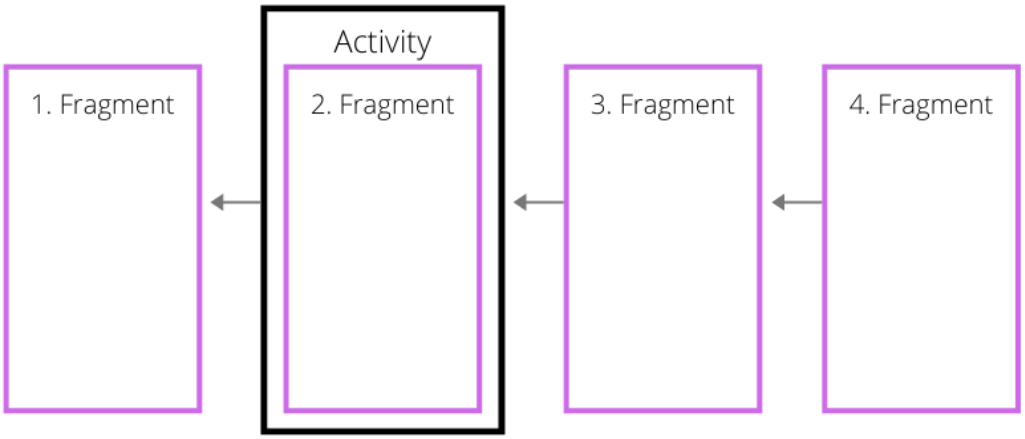
9. ábra Megfigyelő programtervezési minta osztálydiagram séma

A Megfigyelt osztály információt, változókat tárol, amelyeket megoszt az úgynevezett megfigyelőivel, akik feliratkoztak az információk fogadására. Amennyiben a megfigyelők abba szeretnék hagyni az üzenet fogadását, a feliratkozási listáról való leiratkozással ez megtehető.

4.1.3 Menühierarchia tervezés

Nemcsak a beérkezett jelek több osztálynak való hozzáférését kellett megvalósítani, hanem szükség volt egy olyan felhasználói felület megtervezésére és megvalósítására is, ahol a felhasználó könnyen kiválaszthatja, hogy melyik megjelenítési módot szeretné éppen nézni. A választásom az úgynevezett „Tabbed” megoldásra esett, amely az applikáció kezdőlapján belül füleket tartalmaz. Ez egy internetböngésző alkalmazáshoz hasonlítható, ami lehetővé teszi, hogy egy ablakban több lap legyen egyszerre megnyitva, és azok között váltogathatunk anélkül, hogy a böngészőalkalmazást bezárnánk. Mobilapplikáció esetében ezeket fülnek nevezzük. A MindRove App 4 fület tartalmaz. A megjelenítési módok mindegyikéhez egyet-egyet illetve egy negyediket az adatok mentésére. Ezen menürendszer előnye, hogy nemcsak az adott fülre kattintva hozhatjuk be az adott menüpont tartalmát, hanem jobbra vagy balra húzással is váltogathatunk azok között. Ezen kívül az egyes menüpontok így folyamatosan láthatók, nem kell egy plusz gombnyomás ahhoz, hogy megjelenjenek.

Egy Android Applikáció több alkotóelemből épül fel. A felhasználók az Android operációs rendszerrel és a felhasználói alkalmazásokkal is az úgynevezett Activityken (magyarul: tevékenység) keresztül léphetnek interakcióba. Egy Activity nem más, mint egy felhasználói felület egység, az alkalmazások egyik alapvető alkotóeleme. Központi szerepet játszik abban, hogy a felhasználó miként navigáljon egy alkalmazáson belül, vagy az alkalmazások között. A Fragment (magyarul: töredék) a felhasználói felület egy részét képviseli a hozzá tartozó Activityben. A Fragment az Activity egy moduláris szakasza, amelyet el lehet távolítani, vagy hozzá lehet adni a folyamat futása közben. Saját életciklusa van és megkapja a saját bemeneti eseményeit. Egy Activity több Fragmentet is tartalmazhat többrétegű felhasználói felület létrehozásának céljából. Minden felhasználói felület rendelkezik különféle vezérlőelemekkel, például nyomógombokkal, beviteli mezőkkel stb. Ezek neve Androidos szóhasználatban a view (magyarul: nézet). Ezen kívül minden Androidos alkalmazás rendelkezik egy XML alapú leírással, ami az alkalmazásról, az alkalmazás Activityjeiről, az alkalmazás engedélyeiről, stb. tárol általános információkat. Minden alkalmazásnak rendelkeznie kell ilyen leírással, aminek a neve kötelezően: AndroidManifest.xml [22] [23] [24].



10. ábra Activity és Fragmentek

A MindRove App egy Activityn alapul, melyhez további négy Fragment tartozik. Amikor tehát a felhasználó valamelyik fülre rákattint, ugyanabban az Activityben marad, csak abba egy másik Fragment töltődik be. Az applikáció Activityje a MainActivity (magyarul: fő tevékenység) osztály, a hozzá tartozó fragmentek pedig a TimeBasedDisplay (magyarul: idő alapú megjelenítés), a FraquencyBandsLineChart (magyarul: frekvenciaosztályok vonaldiagram), a FrequencyBandsHistogram (magyarul: frekvenciaosztályok hisztogram) és az AlphaTest (magyarul: alfa teszt) osztályok.

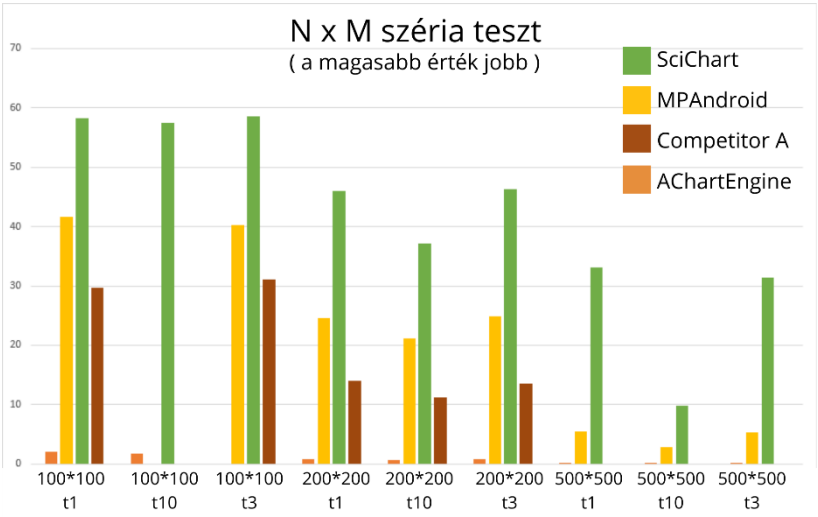
4.1.4 Külső könyvtárak

Az applikáció fejlesztéséhez külső könyvtárakra is szükségem volt, ezek az MP Android Chart és az OpenCV könyvtárak. Előbbit a diagramos megjelenítésnél, utóbbit pedig a jelfeldolgozásnál alkalmaztam.

A diagramos megjelenítéshez a külső könyvtár kiválasztásakor az alábbi szempontokat vettem figyelembe:

- ingyenesen elérhető
- adatok valós idejű megjelenítésére alkalmas
- 500 minta/másodperces beérkező adat megjelenítése során se lassuljon
- több diagramtípus támogatása
- több adatsor megjeleníthető egy diagramon belül

Az MP Android Chart nevű könyvtárra esett a választásom, amely egy nyílt forráskódú külső könyvtár Android számára és támogatja a vonal-, oszlop-, torta-, buborék-, sugár-, és gyertyadiagram típusokat is. Ezek mellett az alábbi, diagramkönyvtárak összehasonlítására szolgáló N*M sorozatteszten a következő ábrán látható eredmény született. Ebben a tesztben Ndb M sorozatot csatolnak a sorokhoz, majd a diagram a lehető leggyorsabban újrarajzol (ugyanazok az adatok), az Y tengely átméretezésével 10 másodpercenként, tesztenként.



11. ábra Az NxM sorozatteszt eredményét bemutató diagram [26]

Látható, hogy a legjobb teljesítményt minden esetben a SciChart hozza, azonban az nem ingyenesen hozzáférhető könyvtár, így azzal nem tudtam dolgozni. Tőle nem sokkal lemaradva a sárga színű oszlop az MP Android, amely ugyan gyengébben teljesít, de a többi nyílt forráskódú könyvtárhoz képest a legjobb eredményt érte el a teszten [25] [26].

A jelfeldolgozás megvalósításához FFT-t végeztem, amelyet különféle jelek, vagy képek frekvenciajellemzőinek elemzéséhez használnak. Ez egy olyan (gyors) algoritmus, amellyel kiszámíthatjuk a DFT-t. A transzformáció elvégzéshez egy külső könyvtárat, az OpenCV-t használtam. Az OpenCV egy nyílt forráskódú képfeldolgozó és gépi tanulás szoftverkönyvtár. Az OpenCV-t úgy építették fel, hogy közös infrastruktúrát biztosítson a képfeldolgozó alkalmazásokhoz, és felgyorsítsa a gépi tanulás kereskedelmi termékekben történő használatát. Ez a könyvtár több, mint 2500 optimalizált algoritmust tartalmaz, amely magában foglalja a klasszikus és a legkorszerűbb képfeldolgozó és gépi tanulási algoritmusok átfogó készletét.

Azért választottam az OpenCV-t, mert nagy felhasználói közösséggel rendelkezik, így folyamatos karbantartás és fejlesztés alatt áll. Ezen kívül rendelkezik Java interfésszel és támogatja az Android operációs rendszert is. Előnye továbbá, hogy nem nekem kellett megírnom a transzformációt elvégző algoritmust, hanem csak meghívtam azokat a függvényeket, amelyet már előre

elkészítettek, és amelyekhez dokumentáció is elérhető. Az OpenCV algoritmusai elsősorban a képek feldolgozásához készültek, azonban azok jelfeldolgozásra is jól alkalmazhatók [27].

4.2 Teszt tervezés

A kész applikációt több szempontból is szükséges tesztelni. A következő teszteket szeretném elvégezni az applikáció programkódjának kiegészítésével, illetve MATLAB segítségével:

1. Adatkommunikáció teszt: Ezen teszt esetében azt kell megvizsgálni, hogy minden UDP adatcsomag beérkezik-e, illetve van-e késleltetés vagy torlódás az adatátvitelben. A teszt megvalósításához a beérkező UDP adatcsomagban található minta sorszám, illetve az oda-vissza kommunikáció is felhasználható.

2. EKG teszt: Nemcsak az agyi aktivitást, hanem a szívműködést is kísérik elektromos változások, amelyek változó erősségű és irányú elektromos erőteret hoznak létre a szív körül. Ez felületi elektródok segítségével regisztrálható és elvezethető. Ezt az elvezetett elektromos aktivitást nevezzük EKG-nek [28]. A MindRove neurofejpánt EKG mérésére is alkalmas. Ez úgy valósítható meg, hogy a felhasználó egyik kezével a 2 referencia elektródot fogja úgy, hogy 2-2 ujja a 2 elektródán van, a másik kezének egyik ujját pedig az egyik elektródhoz érinti. A teszt célja a beérkezett adatok validálása.

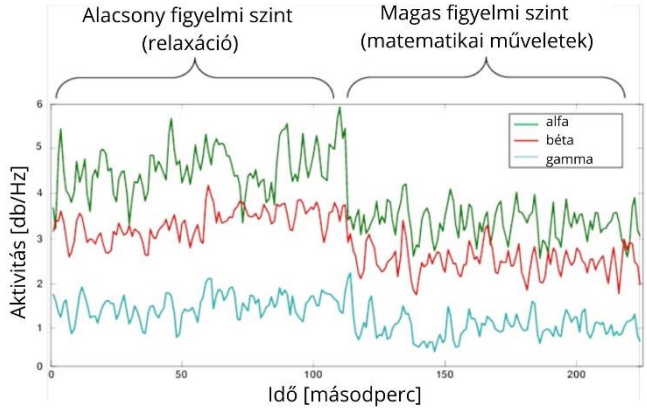
3. FFT algoritmus tesztelése: Teszteltem az OpenCV könyvtár FFT-t megvalósító algoritmusát, hogy megbizonyosodjak a jelfeldolgozás hitelességéről.

4. Adatkésleltetések tesztelése: Ezen tesztek esetében azt vizsgálom meg, hogy mennyi idő telik el az adatok beérkezésétől a jelek feldolgozásáig, illetve milyen paraméterek mellett nem jár elhanyagolhatatlan késleltetéssel az FFT algoritmus. Ehhez a 10. táblázat eszközein szeretnék tesztet végezni.

Név	Típus	Modell	Android verzió	CPU	RAM
1. eszköz	Tablet	Lenovo TB-7104F	8.1.0	4core 1,3GHz	1 GB
2. eszköz	Telefon	Xiaomi Redmi 3S	6.0.1	8core 1,4GHz	3 GB
3. eszköz	Telefon	Google Pixel 3a	11.0.0	2core 2,0GHz & 6core 1,7GHz	4 GB

10. táblázat Az adatkésleltetés teszteléséhez használt eszközök listája

5. Alfa teszt: Az alfa teszt az alfa hullám mérésén alapszik. Lényege, hogy amikor a páciens becsukja a szemét, az alfa aktivitás megnő, míg nyitott szemnél csökken.



12. ábra Agyhullámok relaxált és koncentrációs állapotban [34]

Az alfa teszt lényege, hogy általa ellenőrizhetővé válik a neuro-fejpánttól érkező adat, valamint az applikáció jelfeldolgozása is.

Terv: A felhasználó megnyitja az alkalmazást és megkezdi a mérést egy gomb megnyomásával. Ezt követően egy mérés 2 percig tart, mely során az első percben nyitva, majd a második percben csukva tartja a szemét. A mért adatok mentésre kerülnek egy külső állományba.

5. Az applikáció fejlesztése

Ebben a fejezetben az applikáció fejlesztésének lépéseit, menetét mutatom be az adatkommunikációtól a jelfeldolgozáson és a diagramos megjelenítésen keresztül az adatok külső állományban való mentésének megvalósításáig.

5.1 Az adatkommunikáció megvalósítása

Az adatkommunikáció megvalósítását tesztadat generálással kezdtem, amelyre azért volt szükség, mert eleinte kiszámítható adatokkal szerettem volna dolgozni, hogy könnyedén ellenőrizhessem, hogy az adatátvitel megvalósul-e, elvárás szerint működik-e a szoftver. Ehhez módosítottam a hardver firmware-jét, így a küldött adatcsomag tartalma megváltozott, annak első 4 byte-ja négyszögjelet továbbított. A fejlesztés első lépéseként az applikációnak engedélyeket osztottam ki, hogy hozzáférjen a következőkhöz: internet, wifi státusz lekérdezése, wifi státusz megváltoztatása. Ezeket a „felhatalmazásokat” az AndroidManifest.xml fájljában adtam meg, amely tulajdonképpen az applikáció gyökere, ami leírja az alkalmazással kapcsolatos lényeges információkat.

Elkészítettem az applikáció kezdőképernyőjét, amely egy üres szövegdobozt tartalmazott, majd egy egyszerű, UDP adatcsomagok fogadására alkalmas programkódot írtam, melynek leglényegesebb része a runUdp() függvény.

A függvény meghívásával az applikáció a definiált porton keresztül fogadja az eszköz által küldött adatokat, ami jelen esetben a generált tesztadat, ami byte-okban érkezik. Ezekből a byte-okból először sztringet csinál, majd azt a korábban létrehozott szövegdobozban jeleníti meg. A szövegdoboz tartalma azonnal frissül, amint új adatcsomag érkezik az eszköztől. A zavartalan futás érdekében kivételkezelést, azaz a try-catch metódust alkalmaztam.

13. ábra A fogadott adatok karakterláncként való megjelenítése a felhasználó felületen (képernyőfotó)

Ekkor a képernyőn még nem a várt számadatok jelentek meg, hanem összefüggéstelen és értelmezhetetlen karakterláncok, azonban látható, hogy az adatok fogadása sikeres, az adatkapcsolat megvalósult. Ezt követően a beérkező adatokat az alábbi beépített függvénnyel 4 byte-onként számmá alakítottam.

```
Jel=java.nio.ByteBuffer.wrap(buffer).order(java.nio.ByteOrder.BIG_ENDIAN).getInt(0);
```

1. kódrészlet MindRove App: a fogadott jel átalakítása byte-ból számformátummá

A neuro-fejpánt programkódja úgy működik, hogy amennyiben az applikáció egy adott méretű és adott jelet tartalmazó UDP csomagot küld az eszköznek, az fogadja és a fogadás utáni legelső küldött csomagot megjelöli. A kétirányú kommunikáció megvalósítása előnyös lehet az olyan méréseknél, mikor fontos az a pillanat, amikor a felhasználó éppen becsukta a szemét, vagy éppen érzékelt egy hang vagy fényhatást, de emellett a tesztelésnél is fontos funkciója lehet.

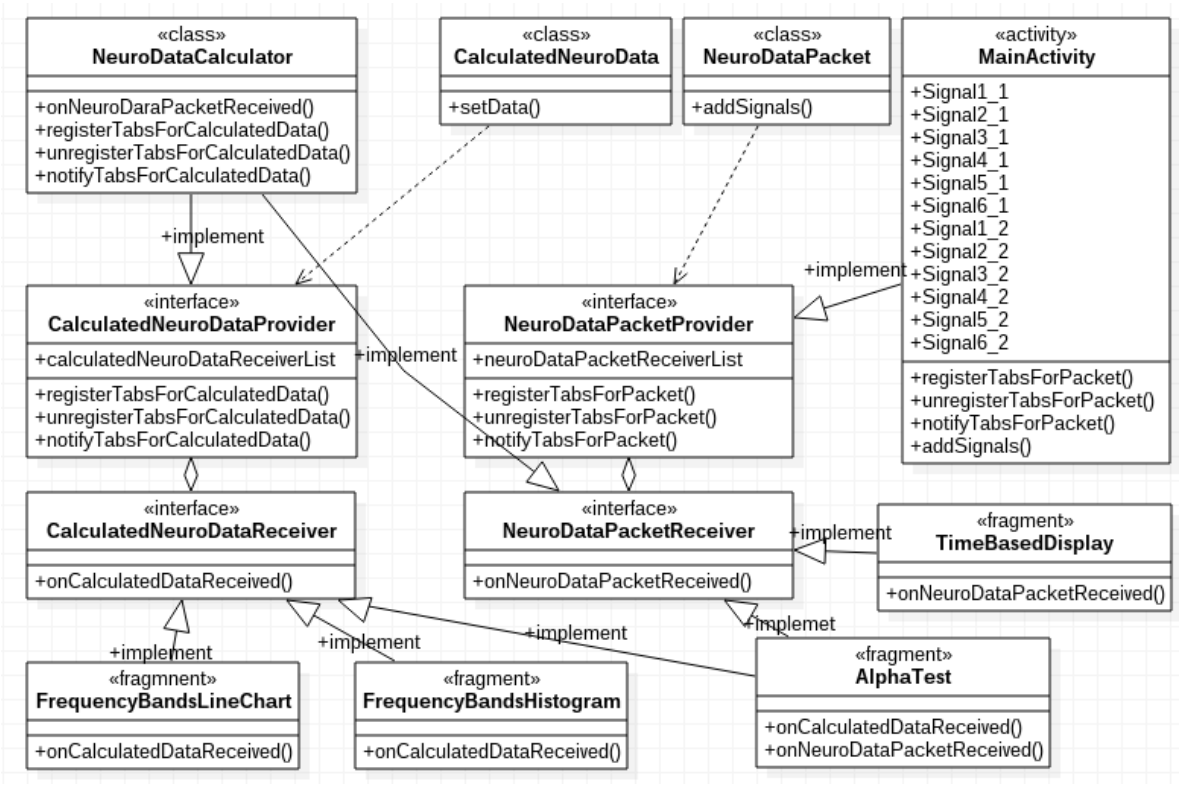
Az applikáció → fejpánt irányú kommunikációt úgy valósítottam meg, hogy összeállítottam egy UDP adatcsomagot, melynek mérete 5 byte és a 0. byte az alábbi értéket tartalmazza: 01010101. Amennyiben a 0. byte-on ez szerepel, úgy a csomag többi részét az eszköz már nem vizsgálja, így ott bármi állhat.

Ekkor a beérkező adatcsomag 76.-79. vagy 184.-187. byte-jain 1-es szám jön vissza. Amennyiben nincs küldött csomag, úgy ezeken a helyeken mindig a 0 szám áll.

Az adatkommunikáció megvalósítása után elvégeztem az 4.2-es pontban tervezett és 6.1 illetve 6.2 fejezetekben ismertetett teszteket.

5.2 A megfigyelő programtervezési minta megvalósítása

A MindRove applikáció fogadja a neurofejpánt által küldött adatokat, és átalakítja azokat számmá. Ez a Main Activity osztályban történik, amely ebben az esetben a Megosztó osztály szerepét tölti be, és a Megfigyelt interfészt implementálja. Három osztály pedig fogadni szeretné ezeket a számmá alakított jeleket. Az osztályoknak nincs szükségük arra, hogy ilyen gyorsan hozzájussanak az adatokhoz, azok tömbösítve kapják meg a jeleket, hat darab 50 elemű tömböt. A három osztály egyike a NeuroDataCalculator osztály, amely a jelfeldolgozást végzi. A feldolgozott jeleket is aztán több osztály kapja meg, így ez Megosztó osztály is lesz. A transzformált adatokra szintén három osztály iratkozik fel.



14. ábra MindRove App osztálydiagram részlet

Ennek megvalósításához nem csak a 4.1.2 fejezetben említett két interfészre volt szükség, hanem négyre. Az egyik Megfigyelt interfész a NeuroDataPacketProvider (magyarul: neurális adat szolgáltatója), a másik pedig a CalculatedNeuroDataProvider (magyarul: számított neurális adat szolgáltatója) nevet kapta. Utóbbi minden egyes kiszámolt jel keletkezésekor értesíti feliratkozóit, a második pedig csak akkor, amikor már megteltek a tömbök. Mindkét adattovábbítási interfészhez tartozik Megfigyelő interfész is, a NeuroDataPacketReceiver (magyarul: neurális adatcsomag

fogadója), valamint a CalculatedNeuroDataReceiver (magyarul: számított neurális adat fogadója). A NeuroDataPacket és CalculatedNeuroData osztályok pedig lehetővé teszik, hogy az adatfogadás metódusának csak egy objektum paramétere legyen hat tömb helyett.

A megfigyelő és megfigyelt interfészek az alábbiak a MindRove App-ban:

```
public interface NeuroDataPacketProvider {
 public void registerTabsforPackets
 (NeuroDataPacketReceiver neuroDatapacketReceiver);
 public void unregisterTabsforPackets
 (NeuroDataPacketReceiver neuroDataReceiver);
 public voPacketid notifyTabsForPackets();}
```

2. kódrészlet MindroveApp megfigyelt interfész

Az interfésznek három metódusa van. Az első a megfigyelő listához való adásra, a második megfigyelő listából való eltávolításra, a harmadik pedig a megfigyelők értesítésére.

```
public interface NeuroDataReceiver {
 public void onNeuroDataPacketReceived(NeuroDataPacket neuroDataPacket);}
```

3. kódrészlet MindRove App megfigyelő interfész

Amennyiben a megfigyelt interfészhez tartozó osztály notifyTabs() metódusa meghívódik, az értesíti a megfigyelőket az esemény bekövetkeztéről (ami jelen esetben 50 jel beérkezése) azáltal, hogy meghívja ezt az onNeuroDataPacketReceived() függvényt. A MindRove App-ban a MainActivity osztály az, ahova implementáltam a Megfigyelt interfészt, így az interfész metódusai itt kapják meg a body-t. Definiáltam egy listát, amely listában lesznek vezetve az események értesítésére feliratkozott osztályok.

Amikor új osztály kerül a megfigyelők listájára, fontos feltételt szabni, hogy csak akkor kerülhessen fel, ha eddig még nem volt rajta, ugyanis, ha már kétszer szerepel rajta, akkor kétszer annyi adatot fog kapni, ha pedig tízszer, akkor a kapott adatmennyiség is tízszer annyi lesz. Az pedig nagyban lassíthatja az applikáció futását. Nem a beérkezett adatok kerülnek majd átadásra, hanem azok másolata. Az adatok lemásolása a NeuroDataPacket osztályban történik. Minden művelet szinkronizáltan történik. Amikor a jelek beérkezése után megtörtént azok számmá alakítása, meghívódik az addSignals() függvény, amelyet a NeuroDataPacket osztály tartalmaz.

A CalculatedNeuroData osztály, azaz a transzformált jelekről való értesítés is ugyanezen az elven valósul meg.

5.3 Jelfeldolgozás

A jelfeldolgozás a NeuroDataCalculator osztályban történik. Ez az osztály tömbösítve kapja meg a beérkező nyers jeleket, amelyeken elvégzi a transzformációt, majd a transzformált jeleket továbbítja. A nyers jelek fogadására azonban nem az osztály maga iratkozik fel, hanem az a Fragment írja fel az osztályt az adatfogadásra, amelynek a transzformált adatra van szüksége. Így az osztály csak akkor fogadja a nyers adatokat, ha van olyan Fragment, amely felírta.

Az 4.1.1 fejezetben leírt paramétereknek megfelelően a transzformációt mindig egy 1000 elemű tömbön végeztem el, amelynek az új adatcsomag beérkezésekor kitöröltem az első 50 elemét és a végéhez hozzáadtam az újonnan beérkezett adattömb tartalmát.

Az optimális, nagyobb késleltetés nélküli működés érdekében amint az új 1000 elemű tömb elkészül, az azt követő transzformációs és számolási műveletek már egy másik szálon futnak. Így az adatok fogadása, továbbítása történhet párhuzamosan a transzformációval.

Ahhoz, hogy az OpenCV transzformációt végző dft() függvényét meghívjam, MatOfFloat objektumra volt szükségem. A Mat alapvetően egy osztály, amely két adatrészből áll: a mátrix fejlécéből (amely olyan információkat tartalmaz, mint például a mátrix mérete, a tároláshoz használt módszer, milyen címen tárolja a mátrixot stb.), és egy pointerből, ahol a mátrix a pixelértékeket tartalmazza (bármilyen dimenzió esetén, a tároláshoz választott módszertől függően). A mátrix fejlécének mérete állandó, azonban a mátrix mérete képektől függően változhat, és általában nagyságrenddel nagyobb. Mivel jelen esetben a függvényt nem képre, hanem jelre alkalmazom, úgy a mátrix 1x1000-es méretű. A MatOfFloat a Mat egy részosztálya, amely csak lebegőpontos számokat tárol [29].

A dft() függvénynek két paramétere van, mindkettő MatOfFloat objektum. Egyik a transzformálni kívánt adatsort tartalmazza, a másik pedig egy üres, ahova pedig a transzformálás eredménye kerül. Ahhoz, hogy tovább tudjak dolgozni a transzformált értékekkel, egy tömbbe tettem azokat.

Amennyiben egy adatsoron Fast Fourier transzformációt végzünk, az eredmény egy komplex számokból álló adatsor lesz. Az OpenCV dft() függvényének visszatérési értéke egy olyan tömb, amely a komplex számokat az alábbi formában tartalmazza:

Tömb indexe	0.	1.	2.	3.	4.	...	N-2.	N-1.	N.
Tömb eleme	ReY_0	ReY_1	ImY_1	ReY_2	ImY_2	...	$ReY_{N/2-1}$	$ImY_{N/2-1}$	$ReY_{N/2}$

11. táblázat A transzformált adatsort tartalmazó tömb

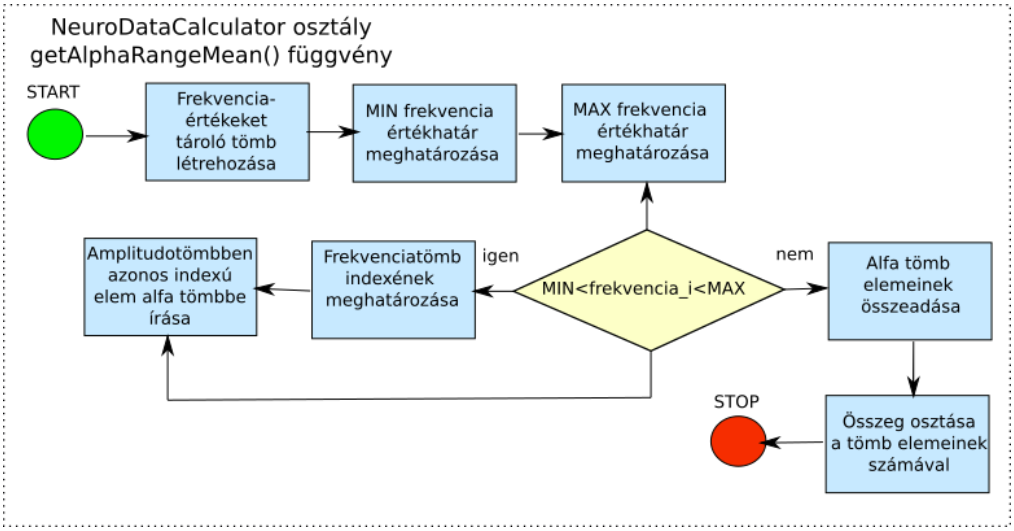
A tömb elemeinek száma megegyezik az input adatsor elemszámával. Azonban látható, hogy az első és az utolsó számnak nincs imaginárius része, azt követően pedig amennyiben az indexelés 0-tól kezdődő, minden páratlan indexű elem valós és minden páros indexű elem az előtte lévő valóshoz tartozó képzetes rész. A komplex számok egy vektort határoznak meg. Ahhoz, hogy a vektor hosszát kiszámítsam, a vektor abszolút értékét kell vennem az alábbi képlet segítségével:

$$\sqrt{x^2 + y^2}$$

1. képlet A komplex számok abszolút értékének számítása

A tömb első és utolsó elemének kivételével párokat képeztem, minden pár első tagja x, második tagja pedig y és az 1. képlet segítségével kiszámoltam az abszolút értéket.

Ezt követően meg kellett keresnem, hogy melyek azok az értékek, amelyek az egyes osztályokba tartoznak. Az Alfa esetén, ezek a 8Hz és 13Hz közötti értékek, melyeket az idő függvényében úgy jelenítettem meg, hogy az egyes csatornák alfa tartományának átlagát vettem, majd a 6 átlagból is átlagot számoltam, így 100 milliszekundumonként lett egy alfa értékem. Ugyanígy jártam el a béta, gamma, delta és théta esetében is. Az adott határfrekvenciák közé eső értékek meghatározását az alábbi módon hajtottam végre:



15. ábra A getAlphaRangeMean() függvény működésének folyamatábrája

Szükségem volt egy olyan tömbre, amely a frekvenciaértékeket tartalmazza, amelyeket az alábbi képlet segítségével határoztam meg:

$$frekvencia_i = \frac{i * mintavételiFrekvencia}{ablakhossz},\ 0 < i < \frac{ablakhossz}{2}$$

2. képlet Frekvenciaértékek meghatározása

Kihasználtam azt, hogy az első csatorna i. eleme és a frekvenciaértékeket tároló osztály i. eleme tulajdonképpen a jel és a hozzá tartozó frekvenciaérték. A függvény a frekvenciaértékeket tároló tömbben megkeresi például az alfa esetében a 8-as értékhez tartozó indexet, majd a jelértékeket tároló tömb ugyanezen indexű elemét egy listába teszi. Végül a listában a 8Hz és 13Hz közötti tartományhoz tartozó jelértékek lesznek. A függvény ezt a listát adja vissza, majd e lista elemeinek átlagát kell meghatározni. Ekkor lesz 6 érték az alfára, így még a 6 érték átlagát is ki kell számolni.

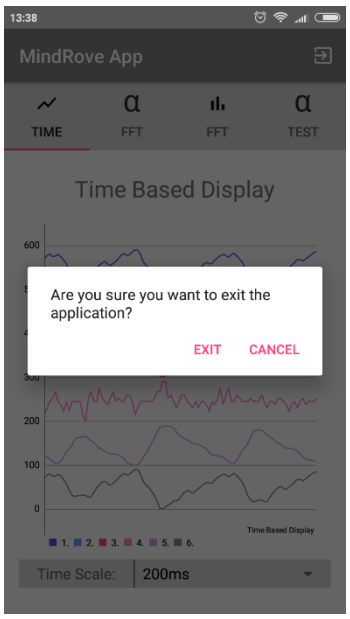
A jelfeldolgozás megvalósítása után elvégeztem az 4.2 fejezetben tervezett és 6.3 fejezetben ismertetett tesztet.

5.4 Az adatok valós idejű, diagramos megjelenítése

Az adatkommunikáció és a jelfeldolgozás megvalósítása után a felhasználói felület programozása következett. Az alkalmazás Activityje egy eszköztárat, valamint négy fület tartalmaz. A hozzá tartozó felhasználói felület az alábbi:

16. ábra MindRove App MainActivity: eszköztár és fülek

Az úgynevezett eszköztárban található egy címsor, azaz az applikáció neve és a kilépés gomb is. A kilépés gombra kattintva a felhasználó kiléphet az alkalmazásból. Rákattintva egy párbeszédablak jelenik meg, amely megkérdezi a felhasználót, hogy biztosan ki akar-e lépni.



17. ábra MindRove App felhasználói felület a kilépés gombra kattintás utána (képernyőfotó)

Az Exit (magyarul: kilépés) gombra kattintva a felhasználó kiléphet a programból, a Cancel (magyarul: érvénytelenítés) gombra kattintva pedig bezáródik a párbeszédablak.

A fülekre kattintva, vagy azok között jobbra-balra húzással navigálva pedig a kiválasztott menüponthoz tartozó Fragment töltődik be.

Az adatok valós idejű megjelenítése több módon is megvalósul. Ezek a következők:

- nyers adatok idő alapú, vonaldiagramos megjelenítése
- frekvenciaosztályok időalapú, vonaldiagramos megjelenítése
- frekvenciaosztályok hisztogramos megjelenítése

Ahhoz, hogy az egyes Fragmentek értesüljenek a beérkező adatokról, fel kell iratkozniuk a listára. A Fragmenteknek saját életciklusa van, így ezt a feliratkozást akkor kell megtenni, amikor az adott Fragment megjeleníti a saját felhasználói felületét. Jelen esetben ez akkor következik be, amikor a felhasználó kiválasztja a hozzá tartozó fület.

Amennyiben pedig a frekvenciaosztályokhoz tartozó kiszámított átlagokról szeretne értesülni, a Fragmentnek először fel kell írnia a NeuroDataCalculator osztályt a tömbösített nyers adatok fogadására, illetve fel kell iratkoznia a számított frekvenciaosztályok átlagértékéről értesítendő adatok listájára.

Fontos továbbá, hogy amikor a felhasználó egy fület kiválaszt, a másik háromnak nem kell frissülnie az új adatokkal, hiszen azok megjelenítését a felhasználó nem látja. Ilyenkor Fragmentek leiratkozhatnak a listáról. Ezt abban az életciklusban kell megtenniük, amikor nem az adott Fragment felhasználói felülete van az Activityn.

A diagramos megjelenítésekhez telepítettem az MP Android Chart könyvtárat (Lásd: 4.1.4), majd a felhasználói felületre beillesztettem a diagramdobozt. Az adatok megadásához egy 1x2-es méretű entries típusú változót definiáltam. Az első elem az X tengely értékét tartalmazza, ami jelen esetben a számláló nevű változóban található aktuális érték. A számláló értéke minden egyes új adatcsomag beérkezésekor eggyel megnő. A második pedig az Y tengely értékét tartalmazza, ami jelen esetben a jel nevű változó aktuális értéke, tehát a beérkezett és dekódolt adat. Ezen kívül a megjelenítésre vonatkozó beépített függvényeket is meghívtam, például, hogy a diagramdobozban mindig automatikusan az éppen legfrissebb értékek legyenek láthatók, azaz a képernyő együtt halad előre a diagrammal.

Amikor először megjelenítettem az adatokat, észrevettem, hogy adatvesztés történik, ugyanis nem került minden beérkező adatpont megjelenítésre a diagramon. Ennek az volt az oka, hogy többszálú programok esetében gyakran előfordul, hogy több szál próbálja meg elérni ugyanazokat az erőforrásokat. Itt minden új adat beérkezésekor lefut a ciklus, amelyben az adatok kirajzoltatása történik, azonban többször a ciklus változói még a kirajzolás előtt felülíródnak az új adat értékével. A probléma megoldásához szinkronizálni kellett a folyamatokat, melynek segítségével meggyőződhettem arról, hogy adott időpontban csak egy szál fér hozzá az erőforrásokhoz. Ehhez a Java programnyelv beépített synchronized() függvényét használtam, amely a szinkronizált blokkban egyszerre csak egy szál futását engedélyezi. Minden más szálat, amely be szeretne lépni a szinkronizált blokkba, addig blokkolja, amíg a szinkronizált blokk belsejében lévő szál ki nem lép a blokkból. Ezt követően már minden adat megjelent a vonaldiagramos ábrázolásban [30].

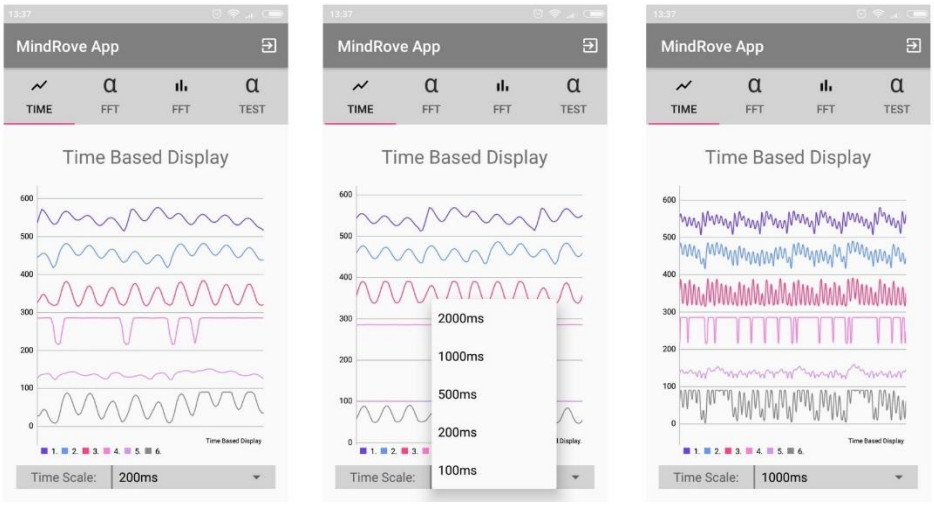
A beérkező UDP adatcsomag első 24 byte-ja és a 108.-tól a 132. byte-ig a 6 csatornán mért jeleket tartalmazza. Az applikáció kódjában pedig ezt a 48 byte-ot dekódoltam, és mind a 6 csatorna adatait megjelenítettem a diagramon. Ezután az egyes csatornákhoz tartozó adatsorok vonaldiagramját skáláztam, hogy azok ne takarják egymást, hanem egymás alatt, sorban, eltolva helyezkedjenek el az átláthatóság érdekében. Ehhez az alábbi képletet használtam:

$$\frac{maximumérték_{új}}{maximumérték_{régi}} - \frac{minimumérték_{új}}{minimumérték_{régi}} * \left(jel - minimumérték_{régi}\right) + minimumérték_{új}$$

3. képlet A jelek y tengely szerinti skálázása

5.4.1 A nyers adatok időalapú megjelenítése

Az időalapú megjelenítéshez tartozó felhasználói felület az alábbi:



18. ábra MindRove App felhasználói felület, időalapú megjelenítés több felbontásban (képernyőfotók)

A képernyő alján látható beállítási opció, a „Time Scale” (magyarul: időskála), amellyel változtatni lehet az X tengely felbontását, azaz, hogy mekkora az az időablak, amelyet meg szeretnék jeleníteni. Rákattintva egy listából választhatunk. A neurofejpánt az adatokat 500Hz-en továbbítja, így egy adatra 2 milliszekundum jut. Az alábbi táblázat azt szemlélteti, hogy a lista egyes elemére kattintva mennyi adatpont kerül megjelenítésre a képernyőn.

Listaelem sorszáma	Időskála [ms]	Adatpont [db]
1.	2000	1000
2.	1000	500
3.	500	250
4.	200	100
5.	100	50

12. táblázat Időalapú megjelenítés választható felbontásai és az azokhoz tartozó adatpontok száma

A fenti ábrán (Lásd: 18. ábra) látható, hogy az 1000 milliszekundumos opció kiválasztásakor hogyan változik a felbontás.

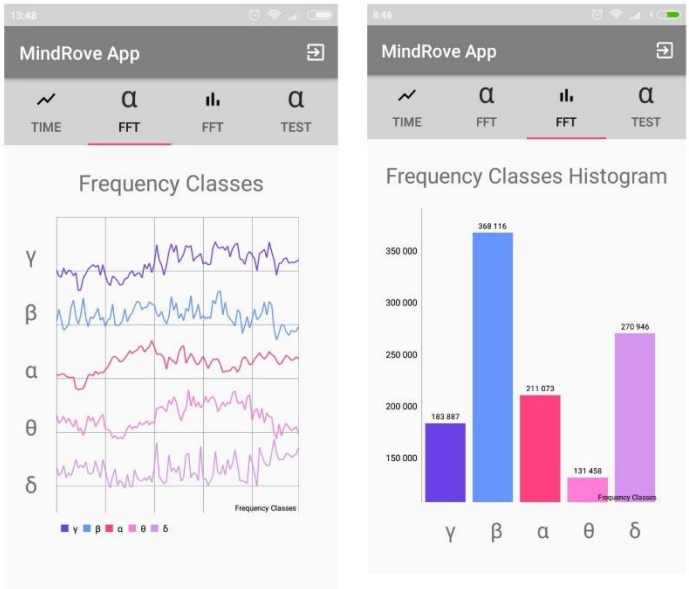
Az időalapú vonaldiagramos megjelenítést megvalósító Fragment a fejpánt által továbbított jeleket tömbösítve kapja meg. Amennyiben a legördülő listából új elem kerül kiválasztásra, úgy a hozzá tartozó adatpontok számát eltárolom egy változóban. A kapott tömb tartalmát hozzáadom egy másik tömbhöz és abban gyűjtöm az éppen megjeleníteni kívánt adatsort. Az applikáció indulásakor az alapértelmezett beállítás a 200 milliszekundumos felbontás, ekkor a változó értéke 100. Ilyenkor

addig gyűjtöm a nagyobb tömbbe az 50 elemű beérkező tömböket, míg annak a mérete el nem éri a 100-as elemszámot. Amikor ez megtörtént, az adatpontok megjelenítésre kerülnek a diagramon, majd amikor beérkezik az újabb 50 elemű tömb, akkor annak elemeit hozzáadom a nagy tömb végéhez, majd a nagy tömb első 50 elemét kitörlöm, így egy újabb 100 elemű tömb kerül megjelenítésre a diagramon. Így az időalapú megjelenítés 100 milliszekundumos csúsztatásokkal valósul meg.

Amennyiben a felhasználó a felbontást megváltoztatja például 1000 milliszekundumra, úgy az adatok szintén 100 milliszekundumos csúszásokkal 500 elemenként kerülnek megjelenítésre a diagramon. Ekkor az a tömb, amelynek elemei majd megjelennek a vonaldiagramon, fix méretű, mindig megegyezik az éppen egy ablakban megjeleníteni kívánt pontok méretével.

5.4.2 Frekvenciaosztályok vonaldiagramos és hisztogramos megjelenítése

Ebben a két menüpontban az 2. fejezetben bemutatott frekvenciaosztályokat ábrázolom valós idejű vonaldiagram és hisztogram formájában. Ezen megjelenítéshez tartozó felhasználói felületek az alábbi ábrán láthatóak:



19. ábra MindRove App felhasználói felület a frekvenciaosztályok vonaldiagramos és hisztogramos megjelenítése (képernyőfotók)

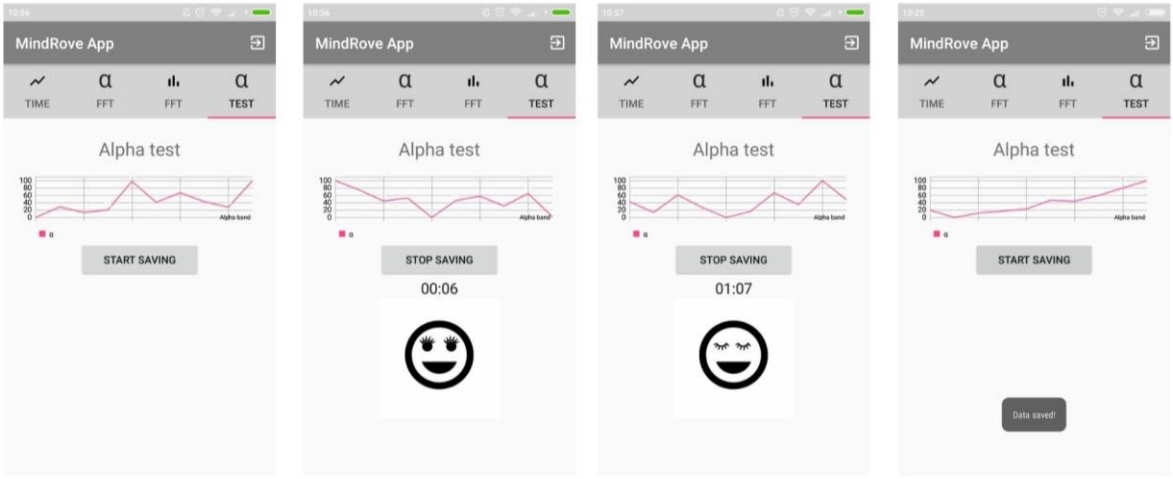
A megjelenítést a FrequencyBandsLineChart és FrequencyBandsHistogram osztályok valósítják meg, amelyek a NeuroDataCalculator osztály által kiszámolt frekvenciaosztályokhoz tartozó átlagokra iratkoznak fel.

A vonaldiagram esetében a számított átlagértékek kerülnek megjelenítésre az idő függvényében a 3. képlet szerint skálázva. A hisztogramnál pedig az egyes átlagértékekhez tartozó amplitúdóértékek láthatóak.

5.5 Az alfa teszt és az adatok mentése külső állományba

A jelfeldolgozás tesztelésére és az adatok validálására elengedhetetlen a beérkező nyers, illetve a transzformált jelek külső fájlba való mentése, ugyanis ez lehetőséget ad az adatok további, offline elemzésére. A negyedik menüpont jelenleg az alfa-teszt végzésére alkalmas menüpontot tartalmazza, ahol kimenthetőek az alfa értékek és a beérkező nyers jelek is.

A teszthez először a felhasználói felületet terveztem meg és fejlesztettem le. A felhasználó gombnyomásra kezdheti el a mérést. Amint a felhasználó megnyomja a mérés elkezdésére vonatkozó gombot, megjelenik egy számláló, amely mutatja, hogy hány perce tart a mérés. Rövid hangjelzés, valamint egy ábra jelzi a felhasználónak, hogy ki kell nyitnia vagy be kell csuknia a szemét. A hangjelzésre azért van szükség, hogy mikor a felhasználó csukott szemmel végzi a mérést, tudja, hogy mikor kell kinyitnia a szemét, az ábra pedig akkor fontos, ha a felhasználó bizonytalan azt illetően, hogy most a nyitott vagy a csukott szemű ciklus jön, ellenőrizni tudja azt. A felületen a valós idejű alfa hullám is látható. Ezen kívül pedig az adatok sikeres mentése esetén azt egy felugró üzenet jelzi a felhasználónak.



20. ábra MindRove App felhasználó felület: Az alfa teszt (képernyőfotók)

Ahhoz, hogy a mentett adatsorokkal később további műveleteket és elemzéseket lehessen végezni, fontos volt eldönteni, hogy pontosan mely adatok, milyen elrendezésbe és milyen formátumban kerüljenek mentésre.

Az alfa teszt alapja maga az alfa hullám, azonban fontos és pontosabb következtetések vonathatók le magából a nyers jelből is, így az alfa értékek és a nyers jelértékek egyaránt mentésre kerülnek.

Egy mérés eredménye így kettő szöveges fájl, azaz az adatok .txt állományba kerülnek. A szöveges fájl előnye, hogy nagyon egyszerű, kevés memóriát foglal, valamint sok programmal kompatibilis,

így a mentett adatfájlt több program, köztük a MATLAB is könnyen be tudja olvasni és a beolvasott jelértékeket fel tudja dolgozni.

Az adatokat mentésére mátrixos alakot alkalmazok. Egy 2 percig tartó mérés esetén tehát egy alfa értékeket tartalmazó fájl 1200 soros és két oszlopos, egy nyers értékeket tartalmazó fájl pedig 60000 soros és hét oszlopos mátrixot tartalmaz. A mátrix utolsó oszlopa mindkét esetben azt jelzi, hogy a jelek nyitott, vagy csukott szem állapotban keletkeztek. Ebben az oszlopban az érték 1 vagy 0 lehet, előbbi nyitott, utóbbi pedig csukott szemnél.

Ahhoz, hogy a nyers adatok és az alfa értékek is mentésre kerüljenek, az alfa teszt osztálynak fel kell iratkoznia a nyers adatcsomagok fogadására, valamint a kalkulált adatok fogadására is.

Amikor a felhasználó megnyomja a „START SAVING” gombot, akkor megjelenik a számláló és az ábra, valamint elindul a timer. A gomb felirata átváltozik „STOP SAVING”-re, tehát ha a felhasználó megnyomja, akkor ugyanazt a gombot, akkor azzal leállíthatja a tesztet és az adatok mentését is.

Amikor a timer eléri az egy percet, egy magasabb sípoló hangot hallhat a felhasználó, ez jelzi, hogy a szemét be kell csuknia. Ez a hang a timer értéke alapján két percenként megismétlődik. Amikor a timer eléri a két percet, akkor pedig egy mélyebb sípoló hang következik, ami a szem kinyitására ad jelet. Ez is 2 percenként ismétlődik, így percenként váltja egymást a magas és a mély sípolás.

A sípoló hang tehát azt jelenti, hogy a felhasználó vagy becsukta, vagy pedig kinyitotta a szemét, azaz egyik állapotból a másikba váltott. Az állapotváltás pillanatának idejét egy időbélyeg változóba mentem el, amelynek az adatok fájlba történő kiírásánál van szerepe.

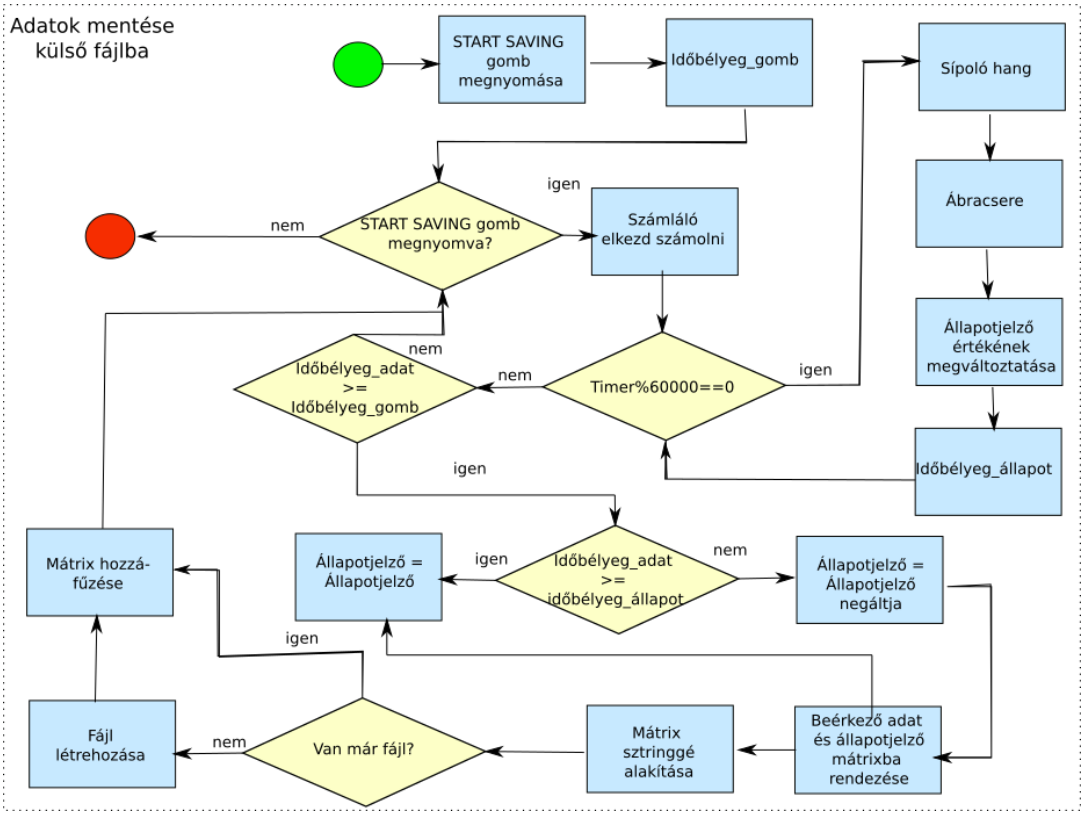
Amikor a felhasználó újra megnyomja a gombot, amin ekkor a „STOP SAVING” felirat olvasható, eltűnik a számláló és ábra, a gomb feliratán pedig újra az eredeti „START SAVING” látható, amit ha ismételten megnyom a felhasználó, egy újabb mérést indít el vele.

Az adatok mentésénél első lépésként mindig azt vizsgálom, hogy a felhasználó elindította-e a tesztet a gombnyomással és a többi művelet csak akkor fut le, ha igen. Először a mérés alatt folyamatosan hozzáadtam a mátrixot tároló 2 dimenziós tömbhöz az új adatokat, majd a mérés befejezésével az egész mátrixot egyszerre írtam ki a fájlba. Ez a megoldás azonban nagyon lassította az applikáció futását, így más opciót választottam. Minden új teszt indításakor létrehozok két új fájlt, egyet az alfa értékeknek egyet pedig a nyers adatoknak. A nyers adatcsomag 6db egydimenziós tömbben érkezik, így azokból létrehozok 1db kétdimenziós mátrixot. Az alfa értékek egyesével érkeznek, így azokat egy 1 soros, 2 oszlopos mátrixba teszem.

Az esetleges csúszások okozta mérési pontatlanság elkerülése végett a mentett adatok nem a mentés gomb megnyomásától a mentés gomb ismételt megnyomásáig tartó időintervallumban kapott adatok, hanem az adatok fájlba írása kezdésének és befejezésének szinkronban kell lennie az adatkésleltetéssel. Ennek megvalósításához minden adatcsomag kap egy időbélyeget, amely időbélyeg értéke az adatokkal együtt megérkezik az AlphaTest osztályba. A teszt indítása után a nyitott szem-csukott szem állapotváltozásokkor is lesz egy időbélyeg. Ezt a két időbélyeget használom fel annak megállapítására, hogy az AlphaTest osztály által kapott adott csukott, vagy nyitott szemű állapotkor keletkezett.

Egy alfa adat esetében tehát a mátrix első sorának első eleme maga az alfa átlagérték, az első sor második eleme pedig az időbélyegek függvényében 1 vagy 0. Amennyiben az adatcsomag egy korábbi időpontban jött létre, mint az állapotváltozás, akkor a jelenlegi állapotnak megfelelő értéket veszi fel, ha pedig később, akkor a jelenlegi állapotnak megfelelő érték negáltját.

Ez azonban így még nem elég, a mentés elindításának folyamatát is igazítani kell az időbélyeg értékekhez. Az adatok mentése akkor kezdődhet meg, ha a mentés gomb be van nyomva, illetve az adatcsomagok később jöttek létre, mint az állapotváltozás ideje, vagy ha a mentés már le lett állítva, de az adatcsomagok még korábban létrejöttek, mint az utolsó állapotváltozás, vagy a tesztet leállító gomb megnyomásának időpontja. Az adatmentés főbb lépéseit a következő folyamatábra mutatja be:



21. ábra Az adatok külső állományba való mentésének folyamata

Az adatok mentése a fájl létrehozásával kezdődik, majd meghívódik az adatok fájlhoz való hozzáfűzését megvalósító függvény. A lebegőpontos számokból álló mátrixokat a kiírás előtt mindig karakterlánccá kell alakítani. Az adatok mentése az alfa értékek és a nyers adatértékek esetén is ugyanazon logika alapján történik. A kapott fájlokban az alábbi módon kerülnek kiírásra a jelek:

Fájl neve	alphaMatrix_1Tue Sep 15 12_02_43 GMT+02_00 2020.txt	rawMatrix_1Tue Sep 15 12_02_43 GMT+02_00 2020.txt
Fájl tartalma (részlet)	2769.1633,1.0 2455.0388,1.0 5261.338,1.0 5428.6113,1.0 6819.7114,1.0 8212.763,1.0 9913.758,1.0 11978.348,1.0 18651.44,1.0 19841.266,1.0	-14.0,-15.0,15.0,-16.0,-81.0,41.0,1.0 -21.0,-13.0,17.0,-24.0,-80.0,27.0,1.0 -18.0,-14.0,7.0,-26.0,-89.0,24.0,1.0 -35.0,-13.0,15.0,-25.0,-93.0,18.0,1.0 -44.0,-16.0,13.0,-29.0,-93.0,10.0,1.0 -52.0,-15.0,13.0,-26.0,-93.0,14.0,1.0 -72.0,-22.0,1.0,-42.0,-102.0,-5.0,1.0 -69.0,-7.0,16.0,-22.0,-88.0,-8.0,1.0 -53.0,-1.0,17.0,-16.0,-80.0,-18.0,1.0 -76.0,-15.0,3.0,-36.0,-103.0,-45.0,1.0

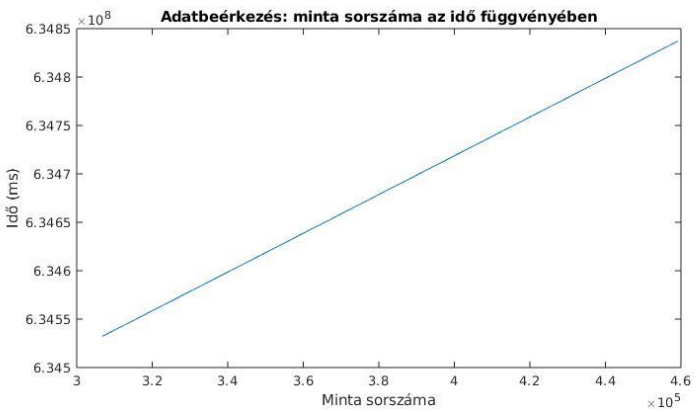
13. táblázat A jelek tárolására létrejött külső fájl tartalma (részlet)

6. Tesztek megvalósítása és eredményei

Ebben a fejezetben az 4.2-es pontban tervezett tesztek megvalósításának módját és az eredményeket ismertetem.

6.1 Az adatkommunikáció tesztelésének eredményei

A beérkező adatcsomagokhoz a beérkezésük időpillanatában egy időbélyeget rendeltem és a MainActivity osztály kódját kiegészítettem azzal, hogy a beérkező minta sorszáma valamint az időbélyeg együtt kerüljön mentésre egy fájlba. Majd a fájl tartalmát MATLAB segítségével egy diagramon ábrázoltam azokat (Lásd: 22. ábra). A mérés időtartama 5 perc volt.



22. ábra Adatbeérkezés: minta sorszáma az idő függvényében (MATLAB ábra)

Ez alapján az adatcsomagok beérkezése ez alapján a rendszertől elvárt ütemben történt. Az esetleges adattorlódás további vizsgálatához kiegészítettem az applikáció forráskódját egy olyan résszel is, amely 15000 beérkező adatcsomagonként (közelítőleg percenként) küld egy 5 byte-os csomagot az eszköznek. Az EEG eszköz a csomag beérkezésekor megjelöli az épp aktuális adatpontot (erre a firmware lehetőséget biztosít). Az Android alkalmazásom pedig figyeli, hogy a kiküldéstől számított hányadik adatcsomag tartalmazza a jelet. Ideális esetben mindig a kiküldés utáni legelső bejövő adatcsomagban lenne a jel.

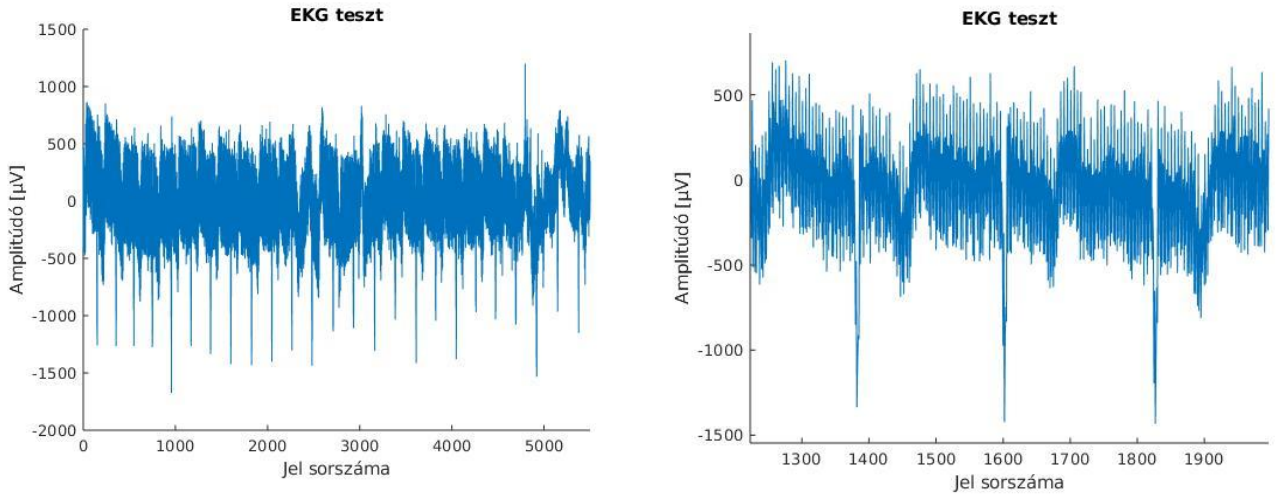
Erre vonatkozóan 5-19 perces méréseket végeztem. Legtöbbször valóban a kiküldés utáni első adatcsomagban érkezik vissza a jel, előfordult, hogy csak a második, harmadik, vagy hatodik csomagban jött. Az ebből adódó, legfeljebb 48 msec késleltetés az alkalmazások szempontjából nem jelent gondot.

Átlag [csomag sorszáma]	1,79 ≈ 2
Módusz	1

14. táblázat Az adatátviteli torlódás vizsgálatának eredménye

6.2 EKG teszt

A teszt megvalósításához a MainActivity osztály kódját kiegészítettem azzal, hogy minden beérkező jel egyesével mentésre kerüljön. Tehát egyből a fogadás után egy fájlba íródjanak ki a jelértékek. A kapott fájl tartalmát pedig egy MATLAB szkript-be olvastam be és megjelenítettem egy diagramon.



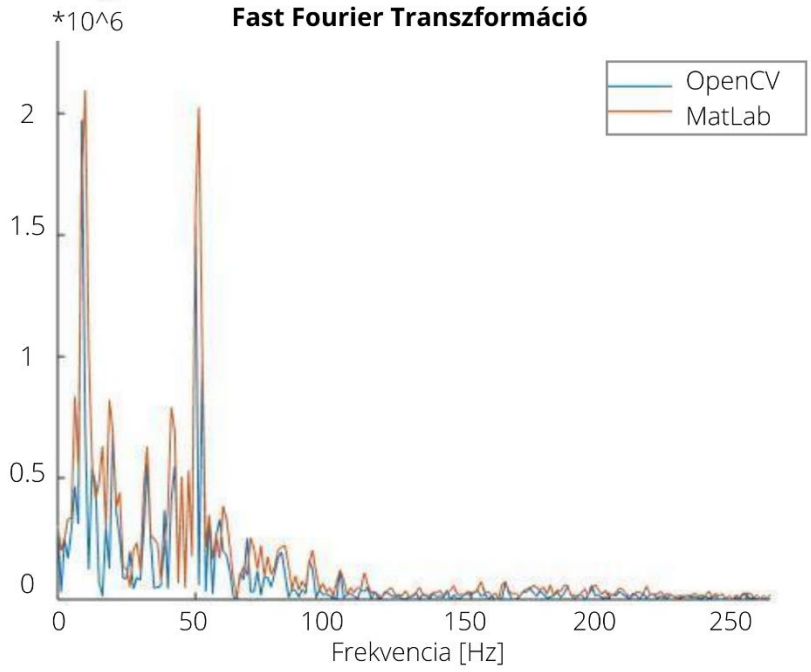
23. ábra Az EKG teszt eredményei (MATLAB ábrák)

Az EKG komplexek egyértelműen, szabad szemmel is láthatók.

6.3 Az FFT algoritmus tesztelése

Az FFT-t az OpenCV dft() függvényével végzem, azonban az eredményét összehasonlítottam a MATLAB FFT függvényével. Ennek célja, hogy megbizonyosodjak, hogy a dft() függvény helyesen számol.

Tehát ellenőrzésképpen készítettem egy MATLAB szkriptet, amelyben a jelnek egy ablakán elvégeztem a transzformációt a MATLAB fft() függvényét használva, és egy diagramon ábrázoltam a MatlLab és az OpenCV által kapott eredményt (Lásd: 24. ábra).



24. ábra A MATLAB és az OpenCV FFT algoritmusának összehasonlítása a beérkezett jelek egy részén (MATLAB ábra)

6.4 Adatkésleltetések tesztelése

Az applikáció működésének egyik legfontosabb része a jelfeldolgozás, azaz a Fast Fourier transzformáció elvégzése, majd a transzformált adatsorokból az alfa, béta, gamma, delta és théta átlagok számítása. Terv szerint a mozgóablakos átlagolás 2 másodperces ablakkal és 100 milliszekundumos lépéssel működik, ami azt jelenti, hogy a program 100 milliszekundumonként elvégzi ezt a számítási folyamatot. A különböző mobil eszközöknek nem egyforma a CPU kapacitásuk, és a memóriaméretük. Itt azt teszteltem különböző telefonokon és tableteken, hogy az

egyes eszközök esetében hol van az a lépéskorlát, ahol az adatok nagyon feltorlódnak és a késleltetés lineárisan növekedni kezd.

Ennek mérésére szintén felhasználtam az oda vissza kommunikációt. 2500 beérkező adatcsomagonként küldtem vissza egy 5 byte-os csomagot. Definiáltam egy időbélyeget minden UDP csomag kiküldésnél, amikor a megjelölt csomag visszaérkezik és az FFT előtt. Szükségem volt a beérkező adatok sorszámára is, mert azt néztem, hogy a megjelölt adatcsomagban beérkező minta mikor kerül be legelőször egy FFT ablakba és ezt a sorszáma alapján tudtam azonosítani. Az időbélyegeket egy szöveges fájlba mentettem. Kimenő adatcsomagonként 3 időbélyegek kaptam, amelyek segítségével meghatároztam a kimenő adatcsomag és megjelölt bejövő adatcsomag között eltelt időt, a megjelölt bejövő adatcsomag és az FFT elé kerülés között eltelt időt, valamint a kimenő adatcsomag és az FFT elé kerülés között eltelt időt. A tesztet az 10. táblázatban leírt eszközökkel végeztem el.

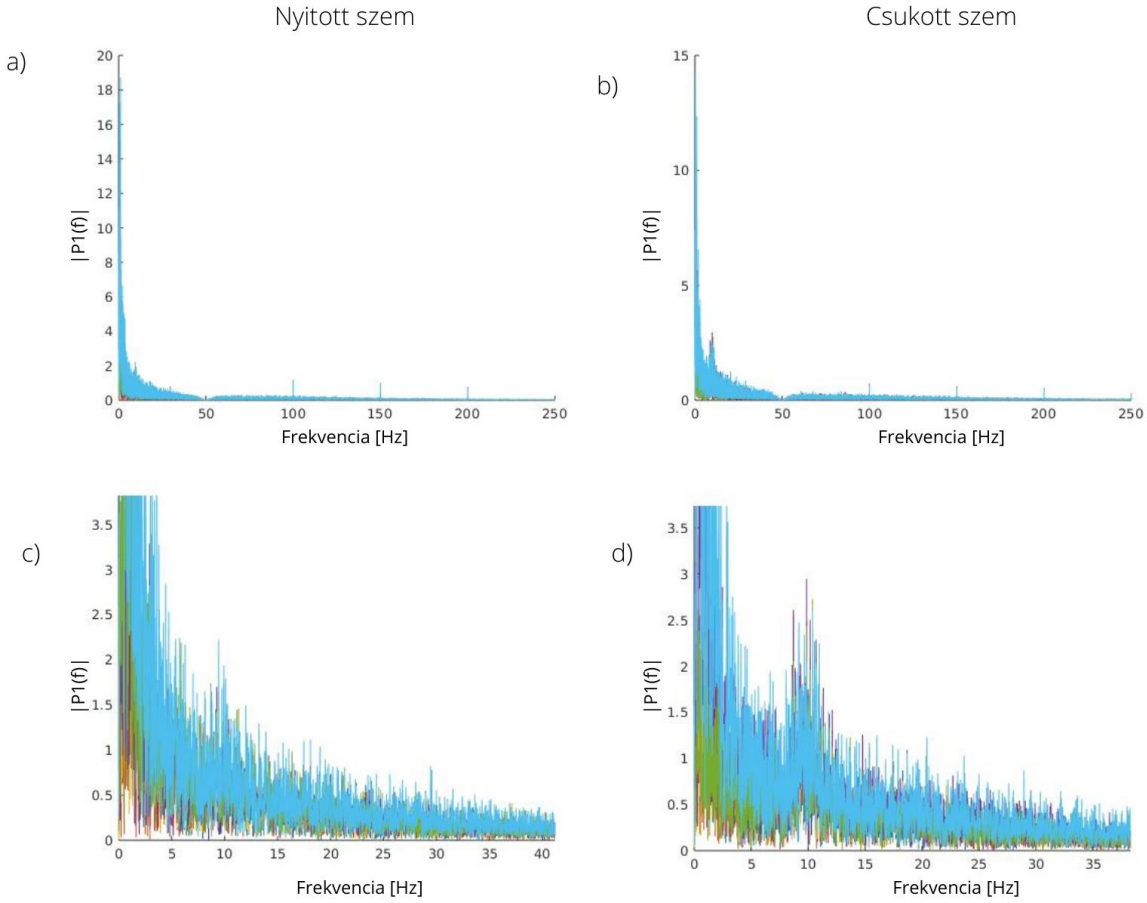
Azt tapasztaltam, hogy a várt lineáris növekedés elmaradt, helyette az applikáció futása egyes paraméterek esetén magától leállt, illetve a valós idejű diagramos megjelenítés is szaggatottá vált, akadozott és végül teljesen lefagyott. Futás közben pedig az adatkésleltetés állandó korlátok között mozgott. Ezért átalakítottam a tesztet és azt a legkisebb lépésközt kerestem, ahol még az applikáció futása nem áll megától 10 percen belül.

Eszköz neve	Legkisebb működő lépésköz [ms]
1. eszköz	56
2. eszköz	40
3. eszköz	40

15. táblázat Az adatkésleltetés tesztelésének eredményei

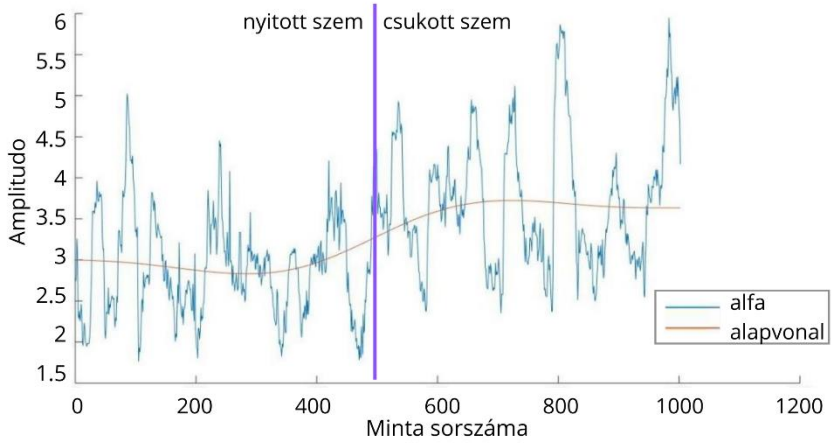
6.5 Alfa teszt

A teszteléshez az 5.5 fejezetben ismertetett Alfa Teszt menüpontot használtam. Mivel az említett menüpontban nemcsak alfa értékek, hanem a beérkező nyers jelek is mentésre kerülnek, így először a nyers jeleken egy mozgóablakos Fourier transzformációt végeztem a MATLAB fft() függvényének segítségével, majd ábrázoltam az eredményt (Lásd: 25. ábra).

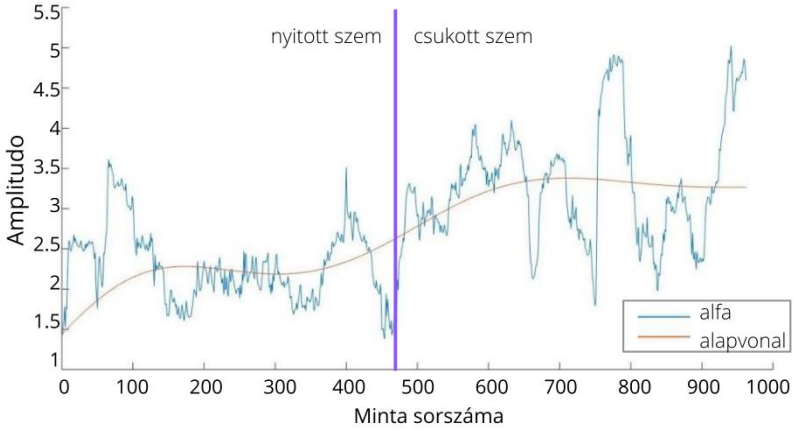


25. ábra Az alfa teszt során, a nyers adaton végzett Fast Fourier transzformáció eredménye a) nyitott szem, b) csukott szem, c) nyitott szem közelítve, d) csukott szem közelítve (MATLAB ábrák)

Ezt követően egy MATLAB szkript segítségével Alfát számoltam, először 2 másodperces ablakkal a 8Hz és 13Hz közötti értékekkel, majd 4 másodperces ablakkal, 9 és 11 Hz közötti értékekkel. Az alábbi eredményeket kaptam:

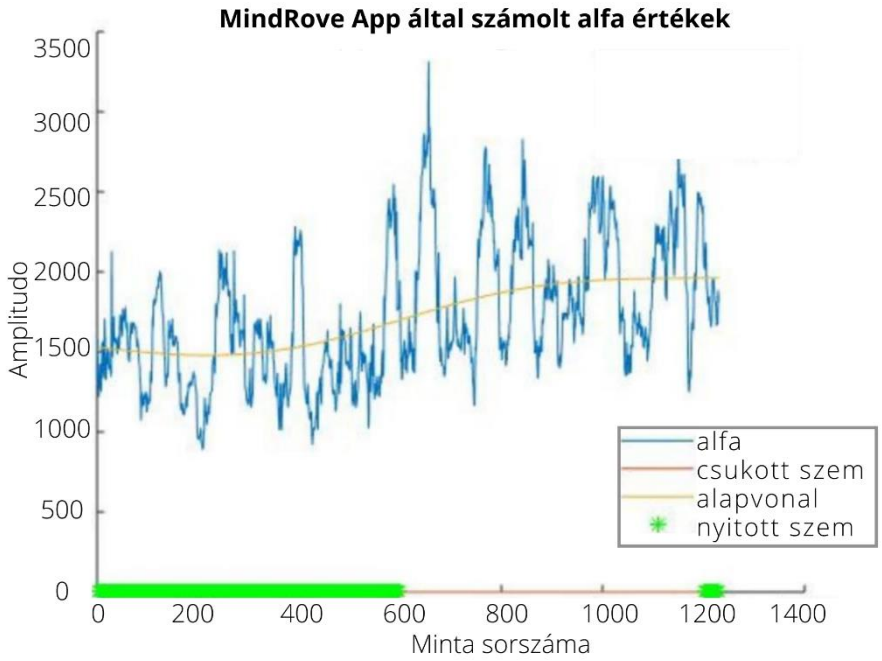


26. ábra MATLAB segítségével számolt és megjelenített alfa hullám. Paraméterek: 2 másodperces ablak, 8-13Hz (MATLAB ábra)



27. ábra MATLAB segítségével számolt és megjelenített alfa hullám. Paraméterek: 4 másodperces ablak, 9-11Hz (MATLAB Ábra)

Utána pedig az applikáció jelfeldolgozó algoritmusa által számolt és mentett alfa értékeket olvastam be a MATLAB szkriptbe és diagramon ábrázoltam az eredményt:



28. ábra A MindRove App által számolt és mentett alfa értékek diagramon ábrázolva (MATLAB ábra)

7. Következtetések

Munkám során sikerült megvalósítani a kétirányú adatkommunikációt az applikáció és a szenzorrendszer között, amely alapját képezheti többféle alkalmazásnak is, melyekre a 7.2 fejezetben hozok néhány példát. Egy olyan applikációt fejlesztettem, amellyel a felhasználó többféle jelfeldolgozási lépést végeztethet a mért jeleken, és vizuálisan megjelenítheti az

agyhullámokat. Az applikációt több szempontból is teszteltem, az eredményekből levonható következtetéseket pedig a következőkben ismertetem.

7.1 A teszteredményekből levonható következtetések

Az 6.1 fejezetben bemutatott adatkommunikáció tesztből következik, hogy az adatok küldése, illetve a mintavételezés megfelelően működik, az adatátvitel sem torlódást, sem adatvesztést nem eredményezett. Az EKG tesztből (Lásd: 6.2) pedig arról is megbizonyosodhattam, hogy azok az adatok érkeznek be, amelyeket várok.

Az 6.3 fejezetben ismertetett FFT algoritmus teszt eredményeiből látható, hogy az OpenCV dft() függvénye valamint a MATLAB fft() nagyon hasonló értéket adott. A különbség elhanyagolható, és valószínűleg abból adódik, hogy a két rendszer némileg más algoritmust használ a Fast Fourier Transzformáció számításhoz.

A transzformációs algoritmus tesztelésénél a várttól különböző eredményeket kaptam (Lásd: 6.4). A felhasználó felületen a valós idejű megjelenítés szaggatottá válása, lefagyása abból adódik, hogy az MP Android Chart, - ahogy az a 11. ábrán is látható - nem valós idejű és nem ekkora mennyiségű adathoz lett fejlesztve. Ez a hiba egy másik, külső, de nem ingyenes könyvtár alkalmazásával kiküszöbölhető lenne. Ezen kívül az FFT lépésközének szűkítésénél a várt eredmény az lett volna, hogy egy bizonyos idő után lineárisan nőni fog a késleltetés, ehelyett az Android letiltotta az applikációt, mert már nem bírta olyan gyorsan feldolgozni az adatokat. Viszont 100 milliszekundumos lépésnél egyik eszköz esetében sem történt leállás, vagy lassulás a megjelenítésben.

Az alfa teszt során egyrészt validáltam a beérkező nyers adatokat és további megerősítést kaptam arra vonatkozóan, hogy az érkezik be, amit várok, ugyanis a nyers adatokat Fast Fourier transzformáció után a frekvencia függvényében ábrázolva (Lásd: 25. ábra) csukott szemnél látható a 8Hz és 13Hz közötti növekedett intenzitás, amely nyitott szem esetében nem figyelhető meg. A 26. és 27. ábrán pedig az látszik, hogy a nyers jelből MATLAB segítségével számított és ábrázolt alfa hullám esetében csukott szemnél erősebb az alfa aktivitás, az alapvonal megemelkedik. Még látványosabb ez, amikor az FFT ablakot növelem 4 másodpercre, az alfa tartományt pedig szűkítem. Az applikáció által számolt alfa hullám (Lásd: 28. ábra) esetében is megfigyelhető az alapvonal emelkedés. Ezzel az applikáció jelfeldolgozásának helyességéről, megbízhatóságáról is megbizonyosodtam.

A tesztelés nagyon lényeges, ugyanis az applikáció ebben a formában is visszajelzést ad az agy funkcionális állapotáról, azonban még rengeteg lehetőség rejlik benne. Az eddig elért eredmények

felhasználásával többféle mobilapplikáció is elkészíthető. Az EEG mérésre épülő applikációk esetében nagyon fontos, hogy veszteségmentes legyen az adatátvitel és megbízható legyen a beérkező adat és a jelfeldolgozás.

7.2 Továbbfejlesztési lehetőségek

A következőkben az elkészült applikáció továbbfejlesztési lehetőségeire hozok néhány példát.

További elemzések: A Fast Fourier Transzformáción kívül a jeleken végezhető Wavelet elemzés vagy Berg transzformáció is. Ezek eredményeit akár összehasonlító elemzésnek is alá lehet venni.

P300 hullám megjelenítése: A pszichológiai kutatásokban talán legtöbbet vizsgált P300 nevű jel akkor jelenik meg, amikor egy inger valamilyen feladat szempontjából fontos, vagy az inger váratlan, újszerű. Az inger megjelenését követően legalább 300ms telik el megjelenéséig, ezért nevezik P300-nak. Ez az összetevő a sorban harmadik pozitív hullám, ezért P3-nak is szokták hívni. Minél kisebb egy inger szubjektív valószínűsége, ez a hullám annál nagyobb. Megjelenik akkor is, ha egy inger váratlanul elmarad. Minél bonyolultabb feldolgozást igényel egy külső hatás, annál később jelenik meg. Ezáltal megmutatja azt is, mikor tart hosszabb és mikor rövidebb ideig egy inger kiértékelése. Ez az adat független azoktól a mozgás-szervezési tényezőktől, melyek a hagyományos reakcióidő-mérések eredményeit torzítják. Éppen ezért a P300 komponens mérését egyre gyakrabban használják a megismerési teljesítmények vizsgálatában akár időskori változásokat, akár kognitív terhelést, vagy egyes farmakológiai hatásokat mérnek [31].

Objektum gondolattal való mozgatása a képernyőn: Ilyen applikáció fejlesztéséhez úgynevezett neurális parancsok létrehozásához van szükség. Ennek első lépése, hogy fel kell mérni a felhasználó neurális állapotát. Ha például a „feltol” parancsot szeretnénk létrehozni, akkor ehhez meg kell kérni a felhasználót, hogy képzelje el, hogy az adott objektum a levegőbe emelkedik. Ekkor a rendszer rögzíti, eltárolja, majd az adott parancshoz társítja a mintákat. Ezáltal, ha a felhasználó mért agyhullámaiban megtalálhatók az adott paranccsal társított minták, megkezdődik az objektum animációja [32].

„Power nap” alkalmazás: Napjainkban egyre nagyobb népszerűségnek örvend külföldön és hazánkban is az úgynevezett „power nap”, ami magyarul rövid alvást vagy rövid töltekező szieszt́át jelent. Célja a normál alvás kiegészítése és az éberség, energia fenntartása. Az alvás nem egy egységes állapot, hanem különböző szakaszokra bontható:

- REM (rapid eye movement, magyarul: gyors szemmozgások) alvás
- a könnyű alvás
- mély alvás

A REM alvásra egy aktivált agyi állapot és álmodás jellemző. Ezen három szakasz ciklikusan változik alvás közben. Az applikáció lényege az lenne, hogy a felhasználók 10-30 perces könnyű alvási fázisban szieszẗázzanak, és még azelőtt felébredjenek, hogy az alvási szintjük átmegy mélyebb, vagy REM fázisba [33].

Adatok offline elemzése: A mért adatok offline elemzése jelenleg is megvalósul, azonban előnyös lenne, ha számítógép és más szoftverek használata nélkül lenne kivitelezhető. Ehhez a mentett adatok fájlból való beolvasására, illetve a beolvasott adatsorokon végrehajtható különböző metódusok fejlesztésére lenne szükség.

8. Összegzés

Eddigi munkám során megismerkedtem a releváns szakirodalommal, azaz az EEG mérés jellemzőivel, a jelek csoportosításával, a neurofeedback terápiával, valamint többféle BCI eszközzel is. Sikerült megvalósítanom a kétirányú adatkapcsolatot a specifikus hardver és az Androidon futó applikáció között, valamint a fogadott adatokat feldolgoztam, transzformáltam és több módon is megjelenítettem az alkalmazás felhasználói felületén. Ezen kívül lehetővé tettem azok külső állományba való mentését és többféle tesztet is elvégeztem, melynek eredményeiből levontam a következtetéseket. Az applikáció továbbfejlesztési és alkalmazási lehetőségeit is ismertettem.

Összességében elmondható, hogy célomat sikerült elérnem, ugyanis a MindRove App mobilalkalmazás jelenleg egyrészt a MindRove neuro-fejpánt illesztőprogramjaként funkcionál, ugyanis két irányú adatkommunikációt valósit meg az eszközzel. Fogadja a mért agyi jeleket és alkalmas adatcsomag küldésére a szenzorrendszernek. Ezen kívül a fogadott adatokat dekódolja, jelfeldolgozást végez Fast Fourier transzformációs algoritmus segítségével és a nyers, illetve feldolgozott jeleket több módon jeleníti meg valós idejűen a felhasználó felületen. Alkalmas továbbá mind a nyers adat, mind az alfa értékek külső állományba való mentésére, ezáltal további, offline elemzések végezhetők az adatokon. A teszteredmények alapján kijelenthető, hogy az adatkommunikáció veszteségmentesen működik, a jelfeldolgozás és a beérkező adatok is megbízhatók.

9. Köszönetnyilvánítás

A TDK dolgozat zárásaként köszönetet szeretnék mondani témavezetőmnek Dr. Márton Gergelynek, a sok segítségért, hasznos tanácsaiért, amelyekkel hozzájárult a dolgozatom létrejöttéhez.

Külön köszönet illeti tanszéki konzulensem Dr. Szlávecz Ákos munkáját, akitől sokat tanultam és akinek segítségével a programozási ismereteimet bővíthettem, fejleszthettem.

10. Irodalomjegyzék

[1] V. Csernus, J. Kállai, S. Komoly és H. Ábrahám, Emberi életfolyamatok idegi szabályozása – a neurontól a viselkedésig, Pécsi Tudományegyetem: Dialóg Campus Kiadó, 2016.

[2] J. Katona, Mobil eszközök agyhullám érzékelésen alapuló irányítása kvantitatív EEG alkalmazásával, Dunaújvárosi Főiskola, 2011.

[3] B. Gulyás és S. Kéri, „Elektrofiziológiai módszerek a kognitív idegtudományban," in *Kognityv Idegtudomány*, Osiris Kiadó, 2003, pp. 81-98.

[4] G. Keszthelyi, „Az agyi elektromos tevékenység (EEG) matematikai elemzése," Eötvös Lóránd Tudományegyetem, 2009.

[5] T. Wenger, „12. fejezet," in *A makroszkópos és mikroszkópos anatómia alapjai*, Semmelweis Kiadó, 2008.

[6] D. A. Kaiser, „What is Quantitative EEG," *Journal of Neurotherapy: Investigations in Neuromodulation, Neurofeedback and Applied Neuroscience,*, pp. 37-52, 2007.

[7] B. Fransworth, „EEG (Electroencephalography): The Complete Pocket Guide," 2019. [Online]. Available: https://imotions.com/blog/eeg/. [Hozzáférés dátuma: 03 10 2020].

[8] A. László, „Jelfeldolgozási módszertanok áttekintése," Szegedi Tudományegyetem, Általános Orvostudományi Kar, Interdiszciplináris Orvostudományok Doktori Iskola, 2013.

[9] A. Gonfanolieri, „A Beginner's Guide to Brain-Computer Interface and Convolutional Neural Networks," 2018. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-brain-computer-interface-and-convolutional-neural-networks-9f35bd4af948. [Hozzáférés dátuma: 02 10 2020].

[10] K. Várszegi, „Detecting Hand Motions from EEG Recordings," Budapest University of Technology and Economics, 2015.

[11] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer és N. J. Wolpaw, „BCI:2000: A General-Purpose Brain Computer Interface (BCI) System," *IEEE Transactions on biomedical engineering,* pp. 1034-1043, 2004.

[12] J. R. Wolpaw, N. Birbaumer, W. J. Heetderks, D. J. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. A. Quatrano, C. J. Robinson és T. M. Vaughan, „Brain-Computer Interface Technology: A Rewiew of the First International Meeting," *IEEE Transactions on rehabilitation engineering,* %1. szám164-173, pp. 164-173, 2000.

[13] D. C. Hammond, „What is Neurofeedback?," *Journal of Neurotherapy: Investigations in Neuromodulation, Neurofeedback and Applied Neuroscience,* pp. 25-36, 2008.

[14] J. Sólyom, „Neurofeedback (EEG Biofeedback)," [Online]. Available: http://tanulas-fejlesztes.hu/neurofeedback-eeg-biofeedback/. [Hozzáférés dátuma: 04 10 2020].

[15] K. Segler és M. Widermann, Neurofeedback, Budapest: Bioenergetic, 2018.

[16] Espressif Systems, „ESP32-WROOM-32 Datasheet," 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. [Hozzáférés dátuma: 26 09 2020].

[17] Last Minute Engineers, „Create A Simple ESP32 Web Server In Arduino IDE," [Online]. Available: https://lastminuteengineers.com/creating-esp32-web-server-arduino-ide/. [Hozzáférés dátuma: 28 09 2020].

[18] A. S. Tanenbaum és D. J. Wetherall, „6. fejezet: Az internet szállítási protokolljai: az UDP, Panem Könyvek," in *Számítógép hálózatok,* Taramix Kft., 2013.

[19] K. Pham, „8 ways to communicate between Fragment and Activity in Android apps," 2018. [Online]. Available: https://hackernoon.com/8-ways-to-communicate-between-fragment-and-activity-in-android-apps-235b60005d04. [Hozzáférés dátuma: 29 09 2020].

[20] Pedrag, „The Observer Pattern in java," 2020. [Online]. Available: https://www.baeldung.com/java-observer-pattern. [Hozzáférés dátuma: 30 09 2020].

[21] Wikipédia, „Megfigyelő programtervezési minta," [Online]. Available: https://hu.wikipedia.org/wiki/Megfigyel%C5%91_programtervez%C3%A9si_minta. [Hozzáférés dátuma: 01 10 2020].

[22] K. Fehér, Alkalmazásfejlesztés Android Studio rendszerben, BBS-INFO KÖNYVK. ÉS INFORM. KFT., 2018.

[23] Android Developers, „Activities," [Online]. Available: https://developer.android.com/guide/components/activities. [Hozzáférés dátuma: 02 10 2020].

[24] Android Developers, „Fragments," [Online]. Available: https://developer.android.com/guide/components/fragments.html. [Hozzáférés dátuma: 02 10 2020].

[25] Android Examples 365, „A powerful & easy to use chart library for Android,” 2017. [Online]. Available: https://androidexample365.com/a-powerful-easy-to-use-chart-library-for-android/. [Hozzáférés dátuma: 02 10 2020].

[26] Z. Zedereckiy, „Android Chart Performance Comparison,” [Online]. Available: https://www.scichart.com/android-chart-performance-comparison/. [Hozzáférés dátuma: 20 09 2020].

[27] OpenCV, „About,” [Online]. Available: https://opencv.org/about/. [Hozzáférés dátuma: 17 09 2020].

[28] A. Fonyó, „11. fejezet: Az elektrokardiogram - Kollai Márk,” in *Az orvosi élettan tankönyve*, Medicina Könyvkiadó Zrt., 2011.

[29] OpenCV documentation, „Mat - the basic container,” [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/core/mat_the_basic_image_container/mat_the_basic_image_container.html. [Hozzáférés dátuma: 29 09 2020].

[30] GeekforGeeks, „Synchronized in Java,” 2019. [Online]. Available: https://www.geeksforgeeks.org/synchronized-in-java/. [Hozzáférés dátuma: 01 10 2020].

[31] I. Czigler és I. Vinkler, „Kognitív pszichofiziológia: Agyi elektromos változások és humán megismerési folyamatok,” *Magyar Tudományok,* pp. 788-796, 1999.

[32] C. AndrewJoseph, „Mental Commands,” 2018. [Online]. Available: https://www.emotiv.com/knowledge-base/training-mental-commands/. [Hozzáférés dátuma: 15 10 2020].

[33] HeadScape, „Power nap,” [Online]. Available: https://www.headspace.com/sleep/power-nap. [Hozzáférés dátuma: 20 10 2020].

[34] M. Rodríguez, R. Giménez, P. Diez, E. Avila, E. Laciar, L. Orosco és A. G. Correa, „Playing with your mind,” *Journal of Physics,477.kötet*, 2013.