МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра радіоелектронних та біомедичних комп'ютеризованих
засобів і технологій

Пояснювальна записка до кваліфікаційної роботи

магістра
(освітньо-кваліфікаційний рівень)

на тему: «Розробка підсистеми для аналізу та використання нейрокомп'ютерного інтерфейсу Emotiv EPOC»

ХАІ.502.560М.23О.172. 093134 ПЗ
(шифр)

Виконав: студент 2 курсу групи 560М
Галузь знань 17 «Електроніка та телекомунікації»
(код та найменування)
Спеціальність 172 «Телекомунікації та радіотехніка»
Освітня програма «Радіоелектронні комп'ютеризовані засоби»
(шифр і назва напряму підготовки (спеціальності))
Євсіков В.А.
(прізвище й ініціали здобувача)
Керівник: Ломоносов Ю.В.
(прізвище й ініціали)
Рецензент: Невлюдов І. Ш.
(прізвище й ініціали)

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій
(повне найменування)
Кафедра радіоелектронних та біомедичних комп'ютеризованих засобів і технологій
(повне найменування)
Рівень вищої освіти другий (магістерський)
Галузь знань 17 «Електроніка та телекомунікації»
(код та найменування)
Спеціальність 172 «Телекомунікації та радіотехніка»
(код та найменування)
Освітня програма «Радіоелектронні комп'ютеризовані засоби»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. В. Висоцька
(підпис) (ініціали та прізвище)
«10» жовтня 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Євсіков Віктор Андрійович
(прізвище, ім'я та по батькові)

1. Тема роботи «Розробка підсистеми для аналізу та використання нейрокомп'ютерного інтерфейсу Emotiv EPOC».
Керівник кваліфікаційної роботи Ломоносов Ю. В., канд. техн. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом університету № 1873а -уч від «10» жовтня 2023 р.
2. Термін подання здобувачем вищої освіти кваліфікаційної роботи 11.01.2024.
3. Вихідні дані до роботи: нейрокомп'ютерний інтерфейс Emotiv EPOC, бібліотека зчитування даних з пристрія.
4. Зміст пояснювальної записки (перелік завдань, які потрібно розв'язати):
4.1. Аналіз роботи нейрокомп'ютерних інтерфейсів.
4.2. Проектування додатка.
4.3. Програмна реалізація додатка.
4.4. Аналіз ефективності отриманих рішень.
5. Перелік графічного матеріалу (додатків):
5.1. Загальна схема роботи НКІ (плакат, арк. А3).
5.2. Нейрокомп'ютерний інтерфейс Emotiv EPOC (плакат, арк. А3).
5.3. Діаграма класів додатка (плакат, арк. А3).

5.4. Аналіз сигналів датчиків (плакат, арк. А3).
5.5. Аналіз сигналів гіроскопа (плакат, арк. А3).
5.6. Аналіз емоційного стану (плакат, арк. А3).
5.7. Робота з паттернами (плакат, арк. А3).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Усі розділи	Ломоносов Ю.В., доцент	23.10.23 р.	

Нормоконтроль _____ Олійник В. М. «11» січня 2024 р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «23» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Отримання завдання	23.10.2023 р.	
2	Аналіз роботи нейрокомп'ютерних інтерфейсів	24.10.2023–01.11.2023	
3	Проектування додатка	02.11.2023–16.11.2023	
4	Програмна реалізація додатка	17.11.2023–14.12.2023	
5	Аналіз ефективності отриманих рішень. Підсумкова оцінка дослідження	15.12.2023–26.12.2023	
6	Оформлення пояснювальної записки	27.12.2023–05.01.2024	
7	Попередній захист роботи та усунення зауважень.	11.01.2024–17.01.2024	
8	Захист кваліфікаційної роботи.	19.01.2024	

Здобувач вищої освіти _____ В.А. Євсіков
(підпис) (ініціали та прізвище)

Керівник кваліфікаційної роботи _____ Ю.В.Ломоносов
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Розробка підсистеми для аналізу та використання нейрокомп'ютерного інтерфейсу Emotiv EPOC. – Євсіков Віктор Андрійович, випускна робота магістра, Харків, Національний аерокосмічний університет ім. Н. Е. Жуковського «ХАІ», кількість сторінок 92, кількість рисунків 15, кількість таблиць 8, кількість джерел літератури 7.

КЛЮЧОВІ СЛОВА: нейрокомп'ютерний інтерфейс, нейро-сигнали, емулятор, гарнітура, БД, електроенцефалограма.

ЦІЛЬ РОБОТИ: створення програми яка буде забезпечувати аналіз і перевірку якості роботи нейрокомп'ютерного інтерфейсу.

ОБ’ЕКТ ДОСЛІДЖЕННЯ: нейрокомп'ютерний інтерфейс.

ПРЕДМЕТ ДОСЛІДЖЕННЯ: методи роботи інтерфейсу «мозок-комп'ютер».

ЗАДАЧИ ДОСЛІДЖЕННЯ: аналіз роботи нейрокомп'ютерного інтерфейсу, розробка алгоритму зчитування даних з пристрою, розробка UI, розробка бази даних для зберігання результатів перевірок.

МЕТОДИ ДОСЛІДЖЕННЯ: методи роботи з сигналами, методи розробки програм, методи розробки БД.

ОТРИМАНІ РЕЗУЛЬТАТИ: проаналізовано методи роботи нейрокомп'ютерних інтерфейсів, розроблений алгоритм зчитування даних з пристрою, розроблено UI, розроблена база даних для зберігання результатів перевірок.

ABSTRACT

Development of a subsystem for analysis and use neurocomputing interface Emotiv EPOC. – Evsikov Viktor Andreevich, master graduation work, Kharkiv, National Aerospace University “Kharkiv Aviation Institute”, amount of pages 92, amount of pictures 15, amount of tables 8, amount of literature’s sources 7.

KEY WORDS: Neurocomputing interface, neuro- signals emulator, headset, DB, electroencephalogram.

GOAL OF WORK: creating an application that provides analysis and validation of neurocomputing interface.

OBJECT OF RESEARCH: neurocomputing interface.

SUBJECT OF RESEARCH: methods of the interface " brain-computer".

TASKS OF RESEARCH: analysis of neurocomputing interface to develop an algorithm to read data from the device, the development of UI, the development of a database to store test results.

METHODS OF RESEARCH: work with signals, methods of application development, database development methods.

RESULTS: the methods of neurocomputing interfaces developed an algorithm to read data from the device, designed UI, developed a database to store test results.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

НКІ – нейрокомп'ютерний інтерфейс
BCI – brain-computer interface
БД – база даних
ЕЕГ – електроенцефалографія
СУБД – система управління базами даних
UI – user interface
ПП – програмний продукт

ВСТУП

Розробка різних варіантів інтерфейсу «мозок-комп'ютер» (BCI) останніми роками перестала бути суто експериментальним напрямом і знаходить дедалі більшого практичного застосування.

Методи реєстрації електричної активності мозку було розроблено 1929 року німецьким фізіологом Гансом Бергером. Вже в тридцяті роки електроенцефалографія (ЕЕГ) почала сприйматися не тільки як діагностична процедура, а як щось більш універсальне і перспективне. З'явилася навіть ідея читати думки і використовувати ЕЕГ для мислення управління зовнішніми пристроями.

Незважаючи на значний інтерес, помітних успіхів у розшифровці окремих сенсорних імпульсів і сигналів мозку, що управляють, вчені досягли тільки до сімдесятих років. Великий внесок зробили дослідження Наталії Петрівни Бехтерєвої та роботи Едмонда Девана.

Приблизно тоді стало ясно, що реєстрація потенціалів жодного відношення до читання думок не має навіть у перспективі. Натомість було показано можливість розпізнавати шаблони сумарної електричної активності мозку та використовувати їх для формування уявних наказів електроніці.

Загальна модель нейрокомп'ютерного інтерфейсу є замкнутим потоком інформації. При намірі вчинити будь-яку дію у користувача підвищується електрична активність відповідних зон головного мозку. Ці сигнали знімаються електроенцефалографом і у вигляді цифрових даних надходять до комп'ютера, де проводиться обчислення ознак сигналу, характерних для того чи іншого уявного бажання. Далі набір ознак поділяють за типами, і комп'ютер виробляє команду, яка управляє виконавчим пристроєм. Користувач у режимі реального часу спостерігає за реакцією системи на свою уявну дію.

Базуючись на нових розробках у нейро-технології, група розробників австралійської компанії Emotiv Systems зуміла створити революційно новий особистий інтерфейс для взаємодії людини з комп'ютером – Emotiv EPOC. Даний пристрій є високоточним, бездротовим приймачем нейросигналів. Цей невеликий прилад визначає думки, почуття та емоції людини і передає їх комп'ютеру по бездротовому зв'язку.

Гарнітура виконана в незвичайному (досить простому для такого роду приладів) дизайні і складається з 14 датчиків, двовісного гіроскопа (генерує оптимальні позиційні дані для курсора та камери). Високоточний бездротовий зв'язок дозволяє користувачеві довільно рухатися, не відчуваючи себе скутим проводом. Прилад сумісний з інтерфейсом USB і не вимагає встановлення жодних спеціальних драйверів. Літієва батарея забезпечує 12 годин безперервної роботи.

Новий комп'ютерний маніпулятор відкриває воістину небачені можливості для розваг та керування різними пристроями.

Для людей творчих він дає можливість зображати свої думки прямо на екрані комп'ютера: створювати динамічні кольори, музику та різні форми.

Emotiv EPOC також допоможе людям з обмеженими можливостями керувати технікою на відстані: керувати інвалідним візком, побутовими приладами, відкрити для себе чудовий світ віртуального простору: грати в комп'ютерні ігри та подорожувати сторінками інтернету.

Отже, далі в цій роботі буде розроблено додаток, який буде використовувати весь функціонал пристрою для визначення його можливостей.

1 АНАЛІЗ РОБОТИ НЕЙРОКОМП'ЮТЕРНИХ ІНТЕРФЕЙСІВ

1.1 Історія розвитку нейрокомп'ютерних інтерфейсів

Розробка різних варіантів інтерфейсу «мозок-комп'ютер» (BCI) останніми роками перестала бути суто експериментальним напрямом і знаходить дедалі більшого практичного застосування.

Будь-який рух, сприйняття чи внутрішня розумова діяльність пов'язані з певним патерном активації нейронів, які взаємодіють один з одним за допомогою електричних імпульсів. Ці струми утворюють електромагнітне поле, яке можна зареєструвати зовні голови за допомогою методів електроенцефалографії (ЕЕГ).

Вже в тридцяті роки електроенцефалографія стала сприйматися не тільки як діагностична процедура, а як щось більш універсальне та перспективне. З'явилася навіть ідея читати думки і використовувати ЕЕГ для мислення управління зовнішніми пристроями.

Незважаючи на значний інтерес, помітних успіхів у розшифруванні окремих сенсорних імпульсів та керуючих сигналів мозку вчені досягли лише до сімдесятих років. Великий внесок зробили дослідження Наталії Петрівни Бехтерєвої та роботи Едмонда Девана.

Приблизно тоді стало ясно, що реєстрація потенціалів жодного відношення до читання думок не має навіть у перспективі. Натомість було показано можливість розпізнавати шаблони сумарної електричної активності мозку та використовувати їх для формування уявних наказів електроніці. Метод ЕЕГ, розроблений Гансом Бергером у 1929 році, протягом багатьох років успішно використовується для 3 цілей:

1) діагностики неврологічних розладів у клініках та шпиталях;

2) на дослідження функцій мозку в нейрофізіологічних лабораторіях;

3) для терапевтичних цілей на основі біологічного зворотного зв'язку.

Весь цей час висувалися припущення, що ЕЕГ можна використовувати для читання думок або хоча б для управління зовнішніми пристроями безпосередньо. Це четверте застосування ЕЕГ отримало назву brain-computer interface (BCI), а російськомовної літературі – нейрокомп'ютерного інтерфейсу (НКИ). Низка вчених неодноразово робила спроби керувати пристроями за допомогою ЕЕГ (Dewan 1967, Vidal 1973), а інтерес широкої публіки до цього напрямку постійно підігрівався фантастичними фільмами, де використовувалися аналоги подібного інтерфейсу, наприклад, фільми Firefox, Johnny Mnemonic, Matrix.

Практична потреба у такому інтерфейсі давно назріла. Десятки тисяч хворих уже зараз потребують такого інтерфейсу. Насамперед – це повністю паралізовані люди, пацієнти з тяжкими інсультами та травмами.

Вочевидь, що у основі BCI має лежати розпізнавання патернів біопотенціалів мозку. Якщо випробуваний може змінювати характер своїх біопотенціалів, наприклад, виконуючи певні розумові завдання, то система BCI могла б транслювати ці зміни в контрольні коди, наприклад, переміщення курсору миші на екрані комп'ютера або руки робота-маніпулятора. Також ці коди можна використовувати для вибору літер на «віртуальній клавіатурі» або контролю інвалідної коляски.

У 1988 році Фарвел і Дончин (Farwell 1988) вперше реалізували систему "віртуальної клавіатури", що дозволила друкувати текст, розпізнаючи компонент P300 при зніманні зорових викликаних потенціалів (ВП). Після цього було розроблено багато різних модифікацій BCI систем з зростаючими можливостями, що вже знайшли своє застосування як у клініці для спілкування з пацієнтами, що повністю втратили можливість руху, так і інноваційні технологічні проекти з дистанційного керування роботами.

Швидкість передачі (або пропускна спроможність) цього нового комунікаційного каналу поки невелика. Однак постійний прогрес у галузі техніки реєстрації ЕЕГ, алгоритмів обробки сигналів та розпізнавання патернів, більш глибоке розуміння нейрофізіології та залучення все більшої кількості вчених у ці роботи забезпечують неухильне збільшення цієї швидкості, зростання числа додатків та динамічний прогрес усього напряму в цілому.

Якщо 1994 року було лише 6 дослідницьких груп, що займалися BCI, то перший міжнародний з'їзд по BCI 1999 року приїхали дослідники з 2-х десятків лабораторій. На другому з'їзді у 2002 році були дослідники, які представляли 38 дослідницьких груп, включаючи США, Німеччину, Китай, Фінляндію, Швейцарію, Англію, Канаду та ін.

Зростає та фінансування цих розробок:

У 1999-2001 роках Європейський Союз профінансував міжнародний проект створення адаптивної BCI системи, здатної до подальшого навчання в ході її використання Adaptive Brain Interface (ABI).

Національний інститут здоров'я (NIH) США у 2002 році виділив 3.3 мільйони доларів на подальшу розробку клінічних BCI систем.

Американське Агентство Передових Дослідницьких Проектів (DARPA), відоме своїм ключовим внеском у появу технології Інтернет, виділило 26 мільйонів доларів на покращення технології BCI.

Починаючи з 2001 року, раз на два роки проводиться змагання між BCI системами. З цією метою кілька груп розміщують для вільного доступу набори ЕЕГ даних. Половина записів служить на навчання, а друга використовується для тестування алгоритму.

Вживлювані сенсори та електроди.

Початок цього напряму було покладено дослідами на тваринах. Вивчення нейропроцесів зазвичай починається з вивчення нейронів равликів, як найпростіших і великих клітин такого типу. Але у сфері НКІ результати, що мають значно більше значення, з'явилися в результаті дослідів на мавпах. Саме тоді позначився важливий ривок у розвитку пристроїв, які здатні інтерпретувати «мозкову електрику», простіше кажучи, нейронні імпульси (і

хвилі) в логічний ряд команд за допомогою звичайних алгоритмів і транслювати ці команди в обчислювальні пристрої.

Досліди на мавпах.

У 2001 році Мігель Ніколеліс з університету Дюка (Durham, South Carolina) проводив одні з найвідоміших у цій галузі експерименти. Ніколеліс, вводячи електроди в мозок і "перекодуючи сигнали", зумів синхронізувати рухи "руки" мавпи та "кіборг-руки" - штучного механізму, що повторює форму і функції "руки".

У 2004 році Річард Андерсен та його колеги з Каліфорнійського технологічного інституту (California Institute of Technology) навчилися за допомогою мозкових імплантатів читати думки мавп: передбачати, що вони збираються робити, і навіть дізнаватися, наскільки їм це подобається. Пізнавальні мозкові сигнали такого високого рівня були розшифровані вперше.

Вчені впровадили у парієтальну (лат. parietalis стінний, від paries стіна) кору мозку мавпи 96 електродів, що дало змогу з 67-відсотковою точністю прогнозувати дії тварини. Точність передбачення досягла 88 відсотків, коли дослідники з'ясовували, яку саме нагороду мавпа хоче отримати за виконання завдання, наприклад, чи бажає вона сік чи воду.

У 2008 році було проведено ще один експеримент із мавпами. Ідеї та методи, придумані авторами, мають допомогти медикам та інженерам у розробці протезів нового покоління з «думкою» управління.

Вживлення імплантантів у людини.

У жовтні 2004 року американська компанія Cyberkinetics завершила розпочате у червні 2004 року випробування своєї системи BrainGate: чіп, впроваджений у мозок 24-річного паралітика, дозволив йому «силою думки» керувати телевізором та комп'ютером, зокрема – користуватися електронною поштою.

Чіп BrainGate впроваджується безпосередньо у кору головного мозку. На думку авторів пристрою, це ефективніше, ніж інші підходи, використовувані творцями аналогічних за призначенням інтерфейсів людина-машина (зовнішні електроди, зняття мозкових хвиль). Хірурги впровадили чіп у певну «моторну» ділянку кори мозку. Цей пристрій знімає сигнал одночасно із ста нейронів.

За допомогою спеціальних програм ця людина змогла грати в деякі комп'ютерні ігри, читати та надсилати електронну пошту, керувати телевізором виключно за допомогою «сили думок».

У 2006 році група нейрохірургів, нейробіологів та інженерів з Університету Вашингтона в Сент-Луїсі, США (Washington University in St.Louis) провела експеримент, головним учасником якого став підліток, який страждає на епілепсію. Щоб виявити ділянку мозку, де зароджуються епілептичні напади, підлітку хірургічним шляхом помістили на поверхню мозку мережу електродів. Електричні імпульси з поверхні мозку передаються в комп'ютер та аналізуються за допомогою спеціальних програм.

У 2009 році Група вчених з Університету Брауна (Brown University) в Род-Айленді розпочала другу фазу випробувань на людях у сфері НКІ. Декілька надтонких електродів вживлюються в мозок пацієнта. Нервові імпульси, що

випускаються мозком, прилад перетворює на команди для комп'ютера. Пацієнт силою думки буде здатний пересувати курсор мишки чи іншими підключеними пристроями.

1.2 Визначення та склад нейрокомп'ютерного інтерфейса

Загальна модель нейрокомп'ютерного інтерфейсу є замкнутим потоком інформації. При намірі вчинити будь-яку дію у користувача підвищується електрична активність відповідних зон головного мозку. Ці сигнали знімаються електроенцефалографом і у вигляді цифрових даних надходять до комп'ютера, де проводиться обчислення ознак сигналу, характерних для того чи іншого уявного бажання. Далі набір ознак поділяють за типами, і комп'ютер виробляє команду, яка управляє виконавчим пристроєм. Користувач у режимі реального часу спостерігає за реакцією системи на свою уявну дію.

Найбільше визнання отримало це визначення BCI:

BCI – це комунікаційна система, у якій повідомлення чи команди, надсилані індивідуумом у світ, не проходять через звичайні нормальні вихідні канали мозку як периферійних нервів і м'язів.

Відповідно до цього визначення моргання не можуть бути використані системою BCI. Використовуються або біопотенціали мозку, зареєстровані з поверхні скальпу – ЕЕГ, або поверхні кори.

Є й інші визначення BCI, наприклад:

BCI – це інтерфейс між людиною та комп'ютером, який отримує команди безпосередньо від мозку без здійснення будь-якого фізичного руху.

Або:

BCI використовує електрофізіологічні сигнали для керування зовнішніми пристроями (Bayliss 2001)

Існує і зворотний інтерфейс:

CBI (computer-to-brain interface) - це система реального часу, яка використовується для запису повідомлень або команд прямо в мозок без використання звичайних вхідних каналів мозку. До складу BCI системи входять:

- електроди для відведення біопотенціалів. Мінімальна кількість – 2, частіше записи виробляють за допомогою 21, 64 і навіть 128 каналів. При великій кількості електродів використовують електродні шоломи для швидкості встановлення та збільшення точності позиціонування електродів над певними полями мозку, а також відтворюваності їх розташування від експерименту до експерименту.

- підсилювач біопотенціалів, що підключається до комп'ютера або безпосередньо (наприклад, через порт USB), або через інтерфейсну A/D карту.

- персональний комп'ютер для реєстрації сигналів та їх обробки. Так як у багатьох системах використовується елементи biofeedback, то цей же

комп'ютер, або додатковий ПК показує випробуваному стимули і результати розпізнавання, наприклад, текст, що вводиться.

- програмне забезпечення для реєстрації та обробки ЕЕГ, розпізнавання патернів та пред'явлення стимулів та результатів розпізнавання.

1.3 Опис нейрокомп'ютерного інтерфейса

Загальна модель нейрокомп'ютерного інтерфейсу є замкнутим потоком інформації. При намірі вчинити будь-яку дію у користувача підвищується електрична активність відповідних зон головного мозку. Ці сигнали знімаються електроенцефалографом і у вигляді цифрових даних надходять до комп'ютера, де проводиться обчислення ознак сигналу, характерних для того чи іншого уявного бажання. Далі набір ознак поділяють за типами, і комп'ютер виробляє команду, яка управляє виконавчим пристроєм. Користувач у режимі реального часу спостерігає за реакцією системи на свою уявну дію.

Нейрокомп'ютерний інтерфейс (називається також прямий нейронний інтерфейс, мозковий інтерфейс, інтерфейс «мозок – комп'ютер») – система, створена обмінюватись інформацією між мозком і електронним пристроєм (наприклад, комп'ютером). В односпрямованих інтерфейсах зовнішні пристрої можуть приймати сигнали від мозку, або посилати йому сигнали (наприклад, імітуючи сітківку ока при відновленні зору електронним імплантатом). Двонаправлені інтерфейси дозволяють мозку та зовнішнім пристроям обмінюватися інформацією в обох напрямках.

На рис. 1.1 представлено схему роботи НКІ.

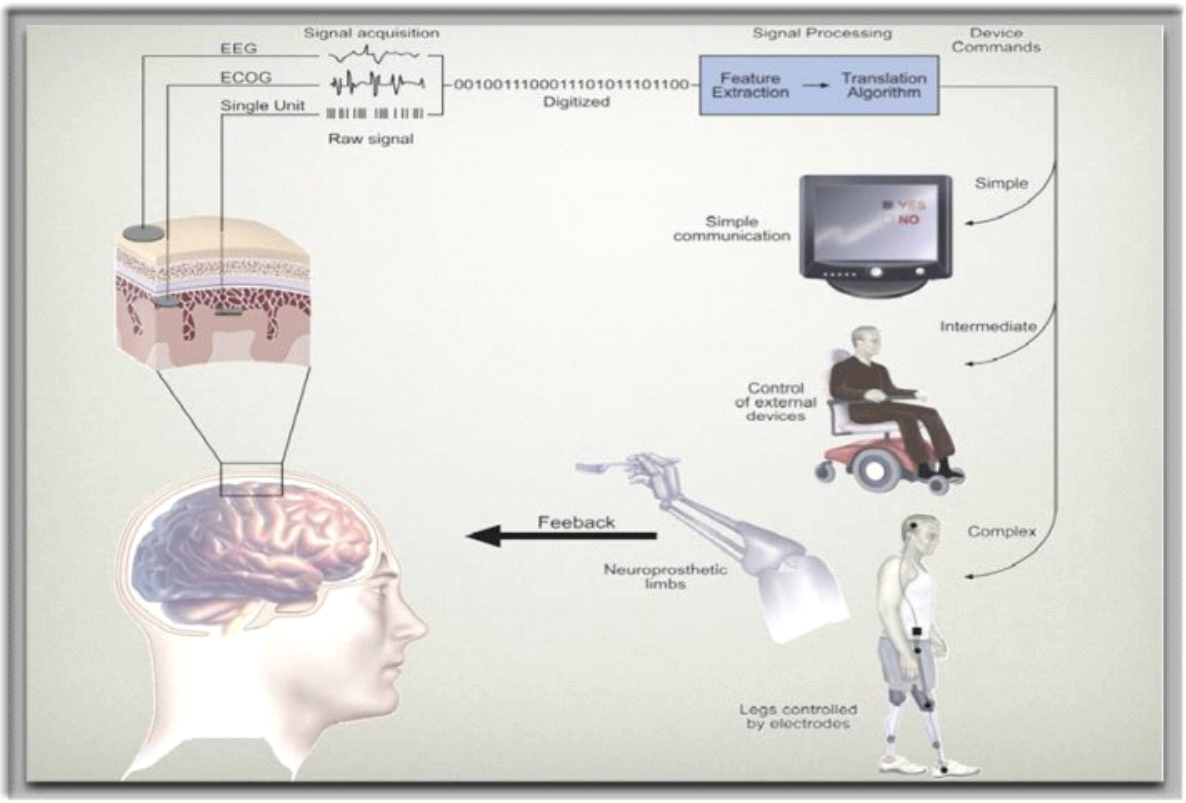


Рисунок 1.1 – Схема роботи НКІ

1.4 Електроенцефалографія

Електроенцефалографія – розділ електрофізіології, що вивчає закономірності сумарної електричної активності мозку, яка відводиться з поверхні шкіри голови, а також метод запису таких потенціалів (формування електроенцефалограм). Також ЕЕГ – неінвазивний метод дослідження функціонального стану головного мозку шляхом реєстрації його біоелектричної активності.

ЕЕГ – чутливий метод дослідження, він відображає найменші зміни функції кори головного мозку та глибинних мозкових структур, забезпечуючи мілісекундний тимчасовий дозвіл, не доступний іншим методам дослідження мозкової активності, зокрема ПЕТ та ФМРТ.

Електроенцефалографія дає можливість якісного та кількісного аналізу функціонального стану головного мозку та його реакцій при дії подразників. Запис ЕЕГ широко застосовується у діагностичній та лікувальній роботі, в анестезіології, а також при вивченні діяльності мозку, пов'язаної з реалізацією таких функцій, як сприйняття, пам'ять, адаптація тощо.

На електроенцефалограм помітна ритмічність електричної активності мозку.

Електроенцефалограма – графічне зображення складного коливального електричного процесу, що реєструється за допомогою електроенцефалографа при розміщенні його електродів на мозку або поверхні скальпу, результат електричної сумації та фільтрації елементарних процесів у нейронах.

Характеристики ЕЕГ.

Для виділення на ЕЕГ значних ознак її аналізують. Основними поняттями, на які спирається характеристика ЕЕГ, є:

1) середня частота коливань

2) їх максимальна амплітуда

3) їхня фаза

4) також оцінюються відмінності кривих ЕЕГ на різних каналах та їх тимчасова динаміка.

Сумарна фонова електрограма кори та підкіркових утворень мозку пацієнта, варіюючи залежно від рівня філогенетичного розвитку та відображаючи цитоархітектонічні та функціональні особливості структур мозку, також складається з різних за частотою повільних коливань.

На рис. 1.2 зображено електроенцефалограму.

Рисунок 1.2 – Електроенцефалограма

Однією з основних характеристик ЕЕГ є частота. Однак через обмежені можливості сприйняття при візуальному аналізі ЕЕГ, що застосовується в клінічній електроенцефалографії, цілий ряд частот не може бути досить точно охарактеризований оператором, так як око людини виділяє тільки деякі основні частотні смуги, явно присутні в ЕЕГ. Відповідно до можливостей ручного аналізу було введено класифікацію частот ЕЕГ за деякими основними діапазонами, яким присвоєно назви букв грецького алфавіту (альфа – 8–13 Гц, бета – 14–40 Гц, тета – 4–8 Гц, дельта – 0,5– 3 Гц, гама - вище 40 Гц та ін.).

Залежно від частотного діапазону, а також від амплітуди, форми хвилі, топографії та типу реакції розрізняють ритми ЕЕГ, які також позначають грецькими літерами. Наприклад, альфа-ритм, бета-ритм, гамма-ритм, дельта-ритм, тета-ритм, каппа-ритм, мю-ритм, сигма-ритм та ін. Вважається, що кожен такий «ритм» відповідає деякому певному стану мозку пов'язаний із певними церебральними механізмами.

Початок вивчення електричних процесів мозку було покладено Д.Реймоном в 1849 році, який показав, що мозок, також як нерв і м'яз, має електрогенні властивості.

24 серпня 1875 року англійський лікар Річард Кетон (1842 – 1926) зробив доповідь на засіданні Британської медичної асоціації. У цій доповіді він представив науковій спільноті свої дані щодо реєстрації від мозку кроликів та мавп слабких струмів. У тому року незалежно від Кэтона російський фізіолог В.Я.Данилевський у докторській дисертації виклав дані, отримані при вивченні електричної активності мозку у собак. У своїй роботі він зазначив наявність спонтанних потенціалів, а також зміни, що викликаються різними стимулами.

У 1882 році І.М.Сєченов опублікував роботу «Гальванічні явища на довгастому мозку жаби», в якій вперше було встановлено факт наявності ритмічної електричної активності мозку. У 1884 року Н.Е.Введенский вивчення роботи нервових центрів застосував телефонічний спосіб реєстрації, прослуховуючи в телефон активність довгастого мозку жаби і кори великих півкуль кролика. Введенський підтвердив основні спостереження Сєченова і показав, що спонтанну ритмічну активність можна виявити і в корі великих півкуль ссавців.

Початок електроенцефалографічним дослідженням поклав В.В.Правдич-Немінський, опублікувавши 1913 першу електроенцефалограму записану з мозку собаки. У своїх дослідженнях він використав струнний гальванометр. Також Правдич-Немінський запроваджує термін електрограма.

Перший запис ЕЕГ людини отримано німецьким психіатром Гансом Бергером у 1928 році. Він запропонував запис біострумів мозку називати «електроенцефалограма». Роботи Бергера, а також сам метод енцефалографії здобули широке визнання лише після того, як у травні 1934 року Едріан і Меттьюс вперше переконливо продемонстрували «ритм Бергера» на зборах Фізіологічного товариства в Кембриджі.

Реєстрація ЕЕГ проводиться приладом електроенцефалографа через спеціальні електроди. Нині найчастіше використовується розташування електродів за міжнародними системами «10 – 20%» чи «10 – 10%». Кожен електрод підключено до підсилювача. Для запису ЕЕГ може бути використана паперова стрічка, або сигнал може перетворюватися за допомогою АЦП і записуватися у файл на комп'ютері. Найбільш поширений запис із частотою дискретизації 250 Гц. Запис потенціалів з кожного електрода здійснюється щодо нульового потенціалу референта, за який, як правило, приймається мочка вуха або соскоподібний відросток скроневої кістки, розташований за вухом і містить заповнені повітрям кісткові порожнини.

1.5 Огляд нейрокомп'ютерного інтерфейса Emotiv EPOC

Базуючись на нових розробках у нейро-технології, група розробників австралійської компанії Emotiv Systems зуміла створити революційно новий особистий інтерфейс для взаємодії людини з комп'ютером – Emotiv EPOC. Даний пристрій є високоточним, бездротовим приймачем нейросигналів. Цей

невеликий прилад визначає думки, почуття та емоції людини і передає їх комп'ютеру по бездротовому зв'язку.

На рис. 1.3 зображено гарнітуру Emotiv EPOC.

Рисунок 1.3 – Emotiv EPOC

Гарнітура виконана в незвичайному (досить простому для такого роду приладів) дизайні і складається з 14 датчиків, двовісного гіроскопа (генерує оптимальні позиційні дані для курсора та камери). Високоточний бездротовий зв'язок дозволяє користувачеві довільно рухатися, не відчуваючи себе скутим проводом. Прилад сумісний з інтерфейсом USB і не потребує встановлення жодних спеціальних драйверів. Літієва батарея забезпечує 12 годин безперервної роботи.

Новий комп'ютерний маніпулятор відкриває воістину небачені можливості для розваг та керування різними пристроями.

Для людей творчих він дає можливість зображати свої думки прямо на екрані комп'ютера: створювати динамічні кольори, музику та різні форми.

Emotiv EPOC також допоможе людям з обмеженими можливостями керувати технікою на відстані: керувати інвалідним візком, побутовими приладами, відкрити для себе чудовий світ віртуального простору: грати в комп'ютерні ігри та подорожувати сторінками інтернету.

Emotiv Systems – австралійська компанія, що розробляє електроніку нейрокомп'ютерних інтерфейсів на основі електроенцефалографії (ЕЕГ).

Emotiv Systems має лише один поточний продукт Emotiv EPOC (англ. Emotiv System's products are the Emotiv) – периферичний пристрій для ігор на

Windows ПК. Emotiv Systems заявляє, що гарнітура дозволить контролювати та впливати на ігри думками та виразом обличчя гравця. Вона з'єднується з комп'ютером бездротової технології, і в майбутньому може працювати на інших платформах, таких як консолі. EPOC був розроблений Emotiv Systems спільно з "Sydney based Industrial Design consultancy 4design".

Отримання інформації.

EPOC має 14 електродів. Самі електроди є пасивними – вловлюють сигнал і передають його далі, кріпляться на поверхні шкіри (незанурюваний інтерфейс) та вимагають змочування спеціальною рідиною для кращого контакту (мокрий інтерфейс). Також має двовісний гіроскоп для вимірювання обертання голови.

Гарнітуру спочатку потрібно «навчити» розпізнавати, яка думка повинна відповідати певній дії. Прилад може вимірювати чотири види даних, але деякі користувачі кажуть, що знімаються головним чином дані з виразу обличчя:

1) Розуміння думки (Cognitiv Suite): уявляється 12 видів руху - 6 напрямків (ліворуч, праворуч, вгору, вниз, вперед і «Зум») і 6 поворотів (обертання по і проти годинникової стрілки, поворот ліворуч і праворуч, нахил вперед і назад) – плюс ще одна візуалізація «зникнення», яку виявляють у Мю-ритмі. Ідеомоторні реакції або сильніші сторонні струми ЕЕГ - ці "думки" команди фактично стають "гарячими клавішами". Через складні алгоритми виявлення викликів є невелике відставання у виявленні думки.

2) Емоції (Affectiv Suite): "Порушення", "Захоплення / Нудьга", "Задумливість", і "Розчарування" зараз можна виміряти. Emotiv визнає, що ці назви можуть відображати не саме ті емоції, які використовуються, і кажуть, що можуть уточнити назви перед виходом на ринок.

3) Вираз обличчя (Expressiv Suite): Індивідуальні позиції повік і брів, положення очей у горизонтальній площині, посмішки, сміх, стиснутість зубів та усмішки зараз можна виявити. Інші вирази можуть бути додані до випуску. Вирази виявляються датчиками ЕЕГ, що збирають сигнали м'язів обличчя, а не шляхом читання мозкових хвиль. На відміну від зчитування психічної активності, виявити зміни в такий спосіб можна дуже швидко (10 мс) надаючи вирішальну перевагу та роблячи їх підходящими для швидких темпів гри у жанрі FPS.

4) Обертання головою: кутову швидкість голови можна виміряти за допомогою нишпорення та тангажу (але не крену). Це реєструється гіроскопами і не пов'язане з особливостями ЕЕГ.

На рис. 1.4 описані складові Emotiv EPOC.

Рисунок 1.4 – Складові частини Emotiv EPOC

На малюнку 1.5 показано, як повинен розташовуватись Emotiv EPOC на голові користувача.

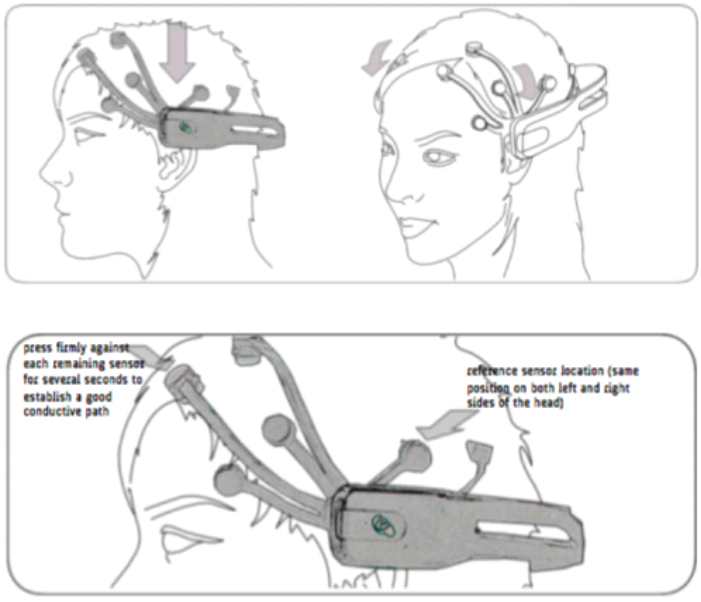


Рисунок 1.5 – Розташування на голові Emotiv SDK.

Emotiv SDK включає бездротову гарнітуру, власні програмні інструменти, які представляються API, і бібліотеки детектування. SDK включає панель

управління, EmoComposer, EmoKey, файли заголовків і бібліотеки імпорту, і приклади коду.

На рис. 1.6 зображено панель управління.

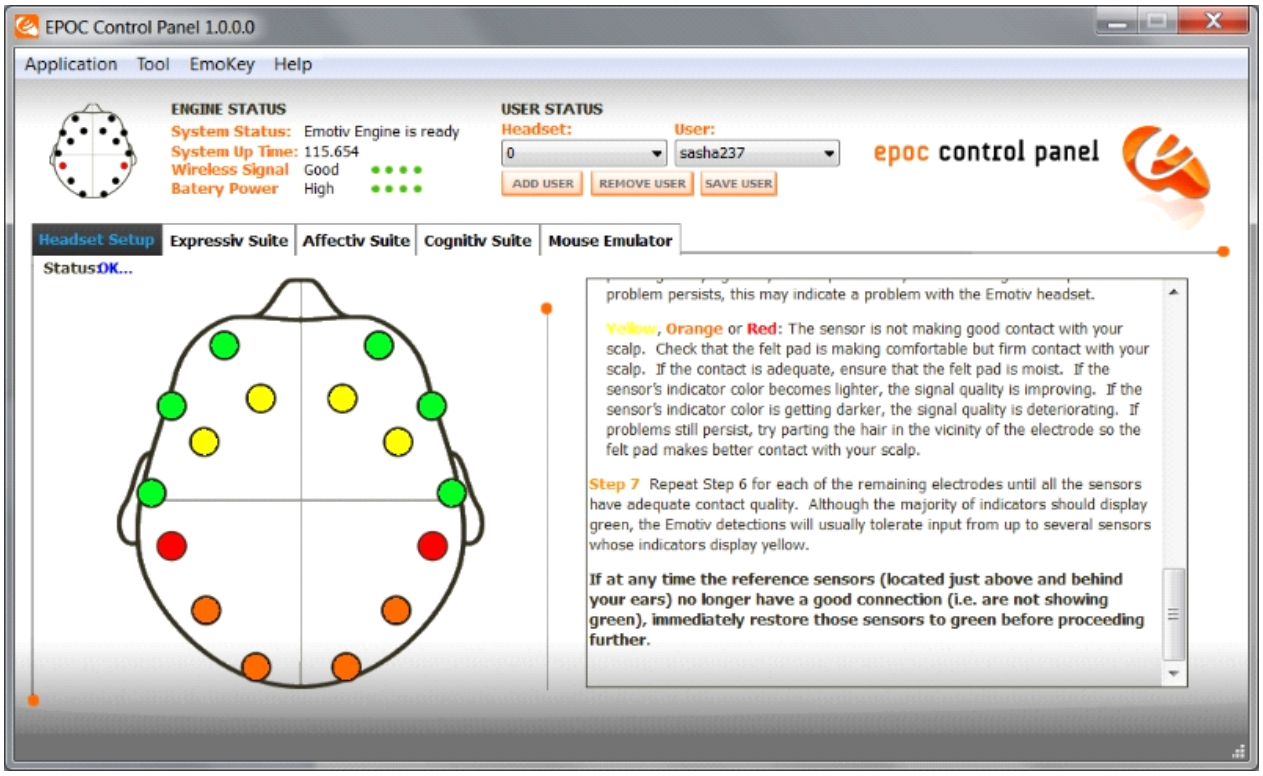


Рисунок 1.6 – Панель керування

EmoComposer & EmoKey це емулятори апарату. SDK забезпечує ефективне середовище розробки, яке добре інтегрується з новими та існуючими структурами.

На рис. 1.7 зображено емулятор гарнітури EmoComposer.

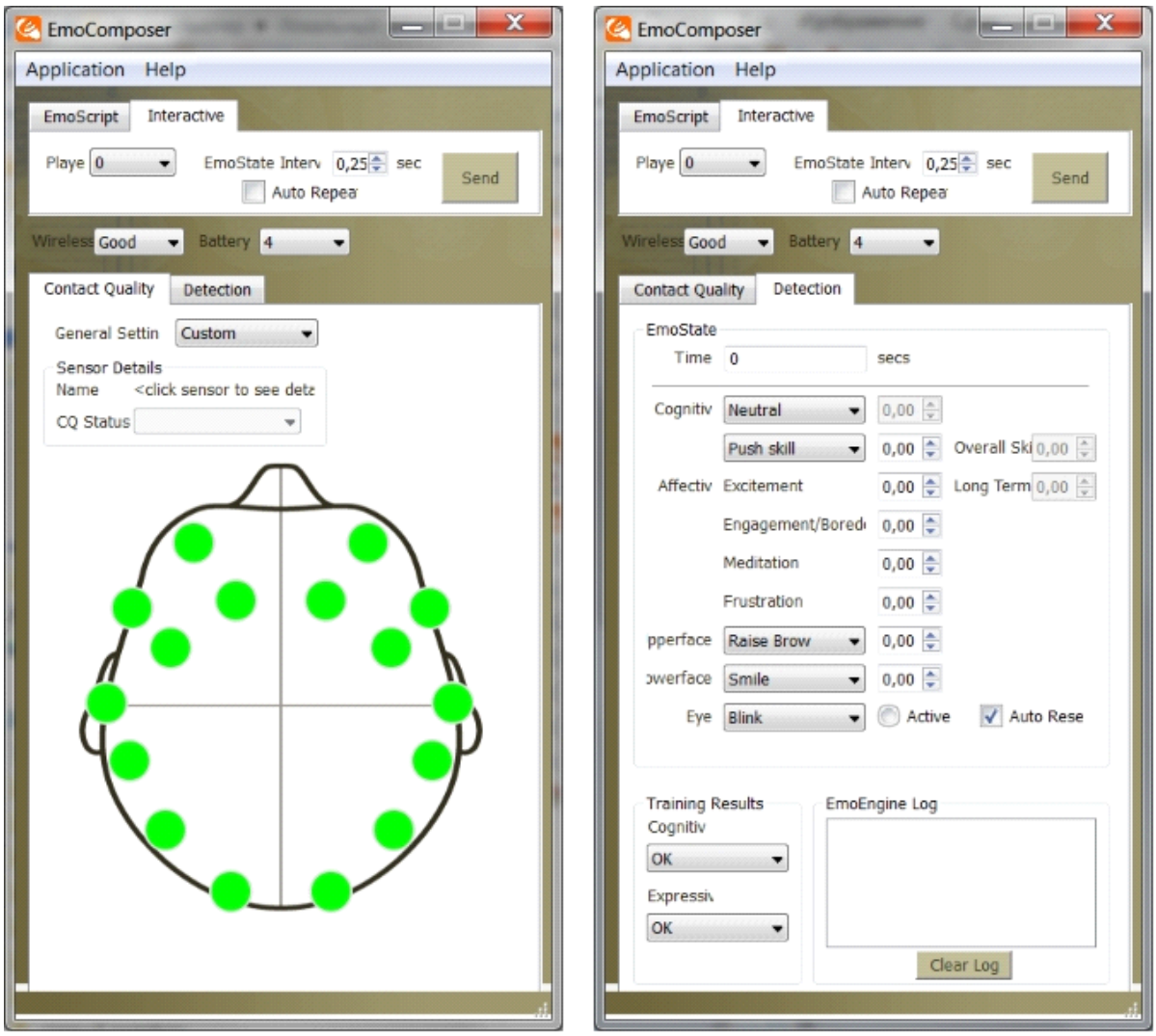


Рисунок 1.7 – EmoComposer

За допомогою цієї гарнітури людина може взаємодіяти з комп'ютером без рук і вибирати потрібну музику або відео, переглядати зображення та грати в ігри.

На рис. 1.8 зображені області, в яких користувач може використовувати пристрій і якими зможе керувати.

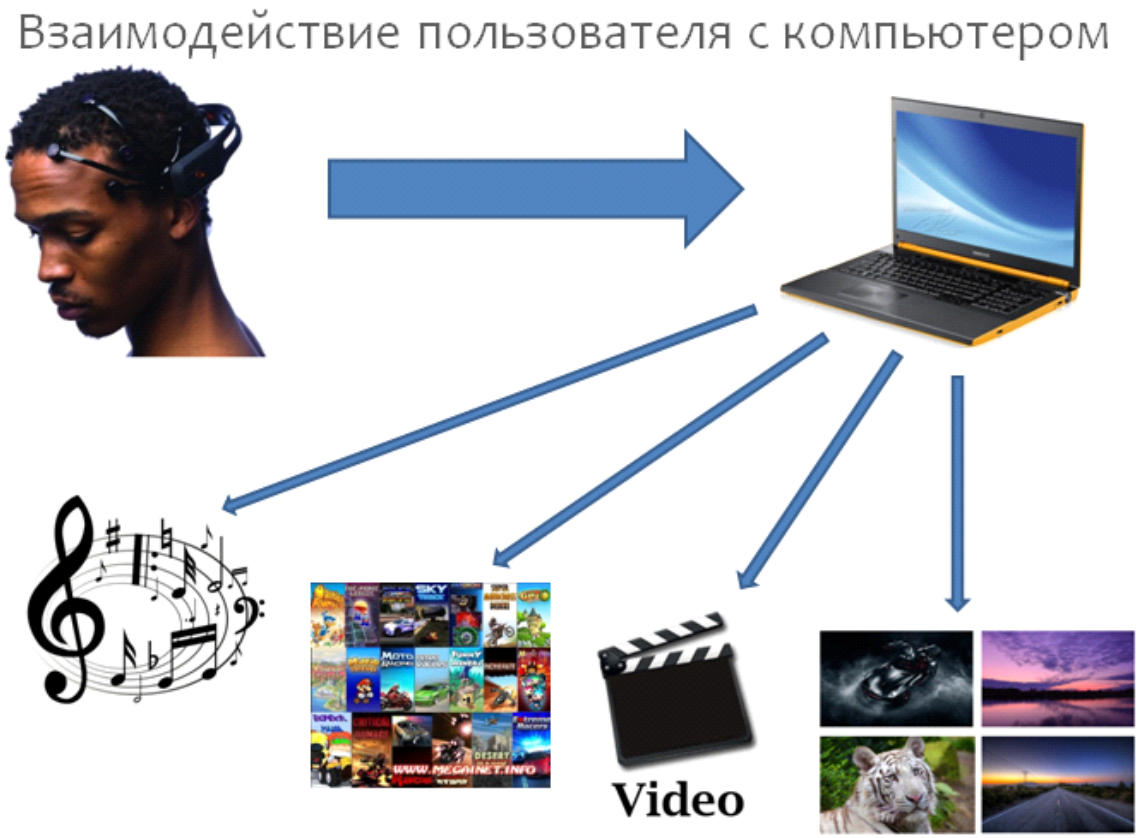


Рисунок 1.8 – Сфери взаємодії

1.6 Постановка задачі

У результаті ясно одне: нейрокомп'ютерний інтерфейс – це майбутнє у сфері IT технологій. Чим швидше освоїть цю технологію фахівці, тим швидше з'являться нові можливості у сфері розваг, бізнесу та медицини. Отже, метою даної дипломної роботи є створення програми, що використовує весь функціонал Emotiv EPOC, оцінка його можливостей та точності роботи.

Завдання програми:

- аналіз роботи нейрокомп'ютерного інтерфейсу;
- розробка алгоритму зчитування даних із пристрою;
- Розробка UI;
- Розробка бази даних для зберігання результатів перевірок.

Для отримання даних із пристрою використовується бібліотека edk.dll.

2 ПРОЕКТУВАННЯ ДОДАТКУ

2.1 Структура програми

Даний додаток включає сім класів. Діаграма класів представлена малюнку 2.1 і 2.2.

Цей додаток заснований на багатопоточності.

Клас MainWindow - клас, який дає команди основному керуючому класу Engine та іншим.

Клас Engine - клас, що виконує основну роботу з отримання даних з пристрою, їх обробку та передачу виконавчим класам.

Клас SensorsWindow відповідає за відображення отриманих даних з кожного сенсора пристрою. Цей клас будує графіки у часі і виводить на екран. Також цей клас представляє можливість обробляти дані за допомогою перетворення Фур'є. Також тут обчислюється якість сигналу та будується фазовий портрет.

Клас GyroWindow – клас, який показує показання гіроскопа. Так як гіроскоп в даному двоосний пристрої то графіки будуються тільки по двох осях. Тут також формується фазовий портрет.

Клас EmotionWindow – клас, що відображає настрій користувача за чотирма категоріями (Frustration, Engagement, Meditation, Excitement). Тут можна стежити за зміною настрою залежно від вибраної аудіодоріжки або зображення. Також зміни видно на спеціальному індикаторі. Даний клас дозволяє запам'ятовувати стан користувача, коли він прослуховує певний трек та записувати ці дані у БД. Це дозволяє надалі зробити зворотне - підібрати музику під поточний стан.

Клас PatternsWindow – клас для аналізу патернів та запису їх у файл.

Клас FilePLS – клас для роботи з аудіофайлами.

На рис. 2.1 зображено діаграму класів:

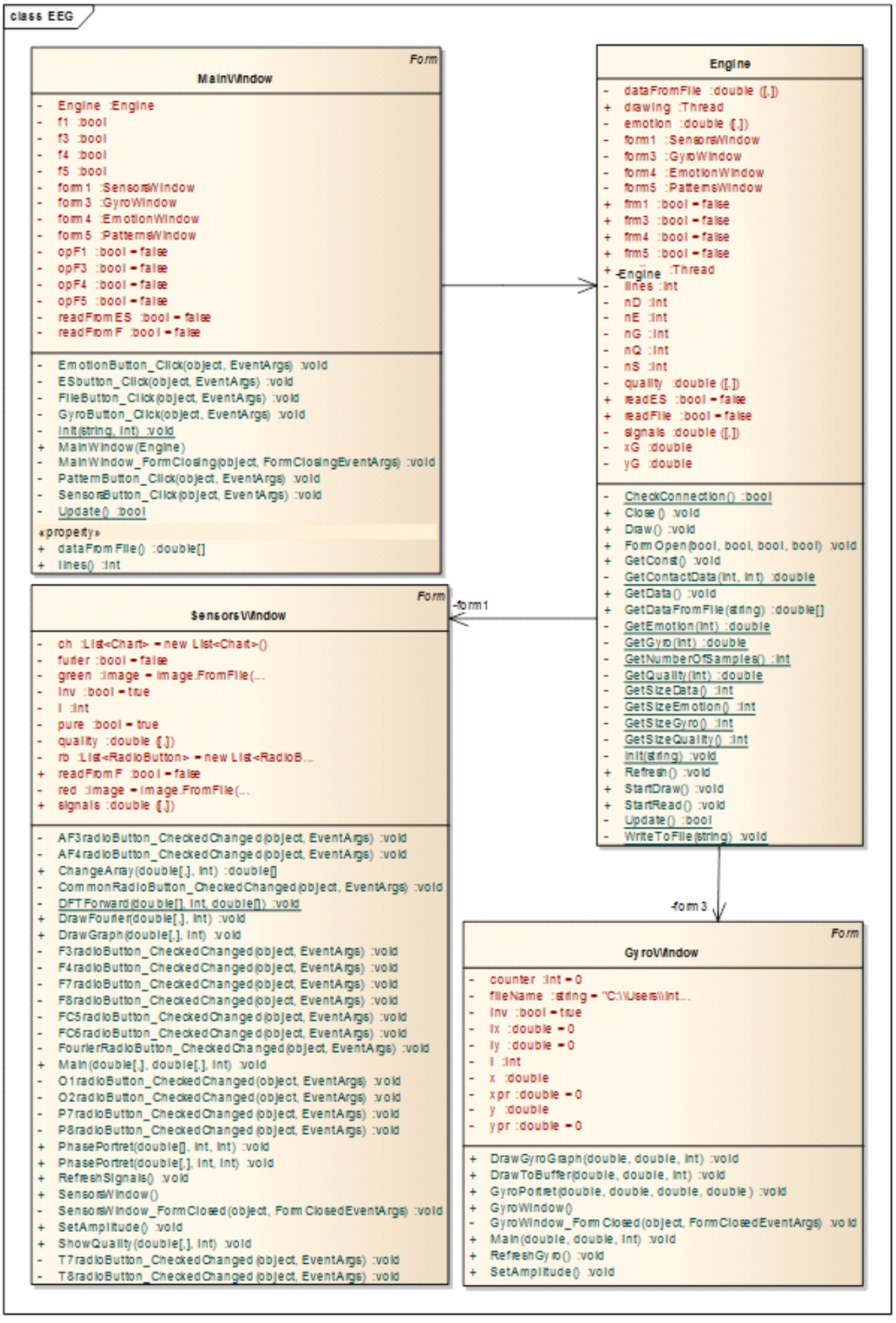


Рисунок 2.1 – Діаграма класів

На рис. 2.2 зображено продовження діаграми класів:

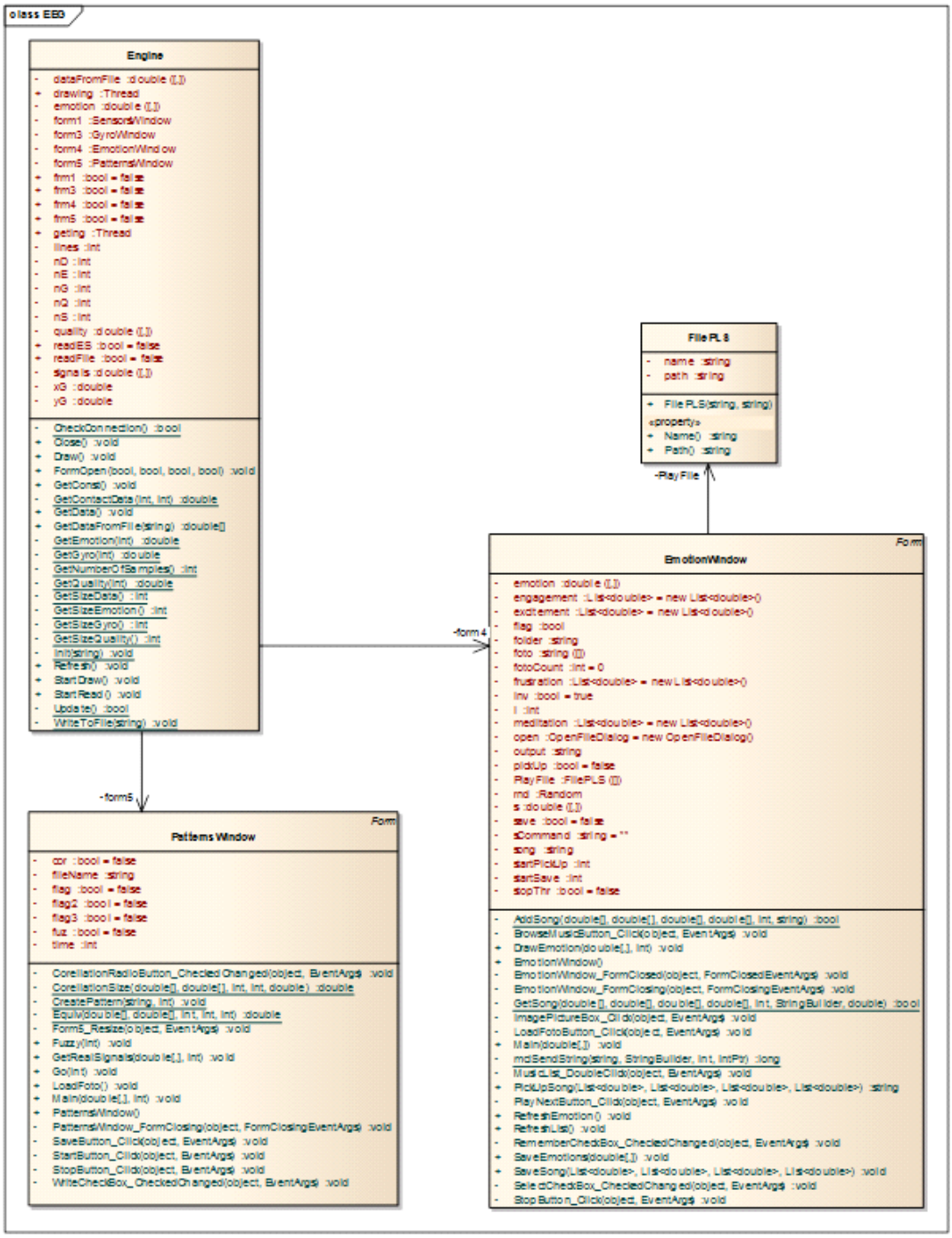


Рисунок 2.2 – Діаграма класів

2.2 Проектування бази даних

Дана БД створена для зберігання показників стану користувача та назви прослуханих аудіофайлів. Вона складається із трьох, пов'язаних між собою таблиць (рис. 2.3).

Таблиця №1 - "User" - зберігає ім'я користувача.

Таблиця №2 - "Mood" - зберігає показник настрою користувача. Містить такі поля як: Mood - показник настрою; «id_us» - поле, яким таблиця зв'язується з таблицею «User».

Таблиця №3 - "Tracks" - зберігає список аудіофайлів. Містить такі поля як: Tracks - назва треку; "Mood" - показник настрою. Цим полем таблиця пов'язані з таблицею «User».

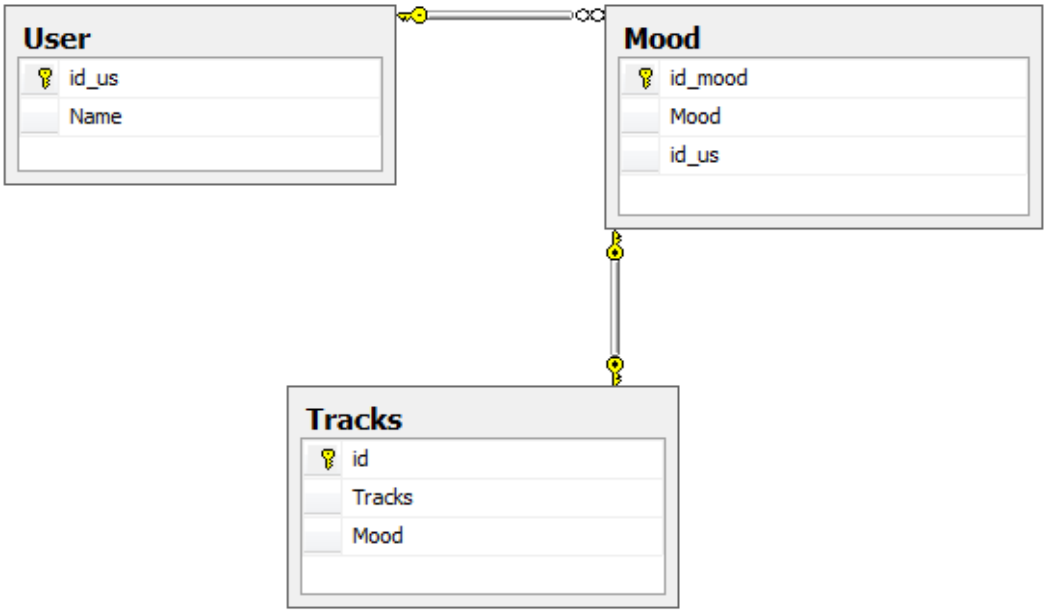


Рисунок 2.3 – Діаграма БД

Як показано малюнку 2.3, все три таблиці пов'язані між собою. При виборі в першій таблиці (“User”) певного користувача, у другій таблиці (“Mood”) завантажаться усі зафіксовані показники настрою вибраного користувача. Також, у разі збігу поточного показника настрою з одним із таблиці “Mood”, у третій таблиці (“Tracks”) завантажиться відповідна аудіодоріжка.

3 ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ДОДАТКА

3.1 Вибір та обґрунтування інструментальних засобів розробки

3.1.1 Переваги .NET

Для реалізації web-програми було обрано технологію .NET і мову програмування C#.

.NET Framework – програмна платформа, випущена компанією Microsoft у 2002 році. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), яке підходить для різних мов програмування. Функціональні можливості CLR доступні у будь-яких мовах програмування, що використовують це середовище.

Переваги .NET:

1) Об'єктно-орієнтоване програмування. І середовище .NET Framework, і C# спочатку повністю базувалися на об'єктно-орієнтованих принципах.

2) Гарний дизайн. Бібліотека базових класів, яка спроектована "з нуля", є виключно інтуїтивно зрозумілим чином.

3) Незалежність від мови. Завдяки .NET, код всіх мов, тобто VisualBasic. Це означає, що всі ці мови мають можливості взаємодії, як ніколи раніше.

4) Найкраща підтримка динамічних Web-сторінок. Хоча ASP пропонував високий рівень гнучкості, він також був і неефективний через свої інтерпретовані сценарні мови, а недолік об'єктно-орієнтованого дизайну часто приводив до заплутаного коду ASP.NET пропонує інтегровану підтримку Web-сторінок із застосуванням нової технології – ASP.NET. В ASP.NET код ваших сторінок компілюється і може бути написаний мовою високого рівня, що підтримує .NET, такою як C# або Visual Basic 2005.5) Ефективний доступом до даних. Набір компонентів .NET, відомий під загальною назвою ADO.NET, надає ефективний доступ до реляційних баз даних та широкого розмаїття інших джерел даних. Також є компоненти, що надають доступ до файлової системи та каталогів. Зокрема, в .NET вбудована підтримка XML, що дозволяє маніпулювати даними, які можуть бути імпортовані з інших, не-Winclows платформ, і експортовані на них.

6) Поділ коду. Середовище .NET повністю змінило спосіб поділу коду між додатками, ввівши концепцію збирання (assembly), яка замінила традиційні бібліотеки DLL. Складання мають форматні засоби для вказівки версій, і одночасно в системі можуть існувати різні версії тих самих збірок.

7) Підвищена безпека. Кожна збірка також може містити вбудовану інформацію безпеки, яка точно описує, кому і яким категоріям користувачів або процесів які методи яких класів дозволено викликати. Це забезпечує дуже високий ступінь контролю за тим, як можуть використовуватися збірки, які ви постачаєте.

8) Інсталяція з нульовим впливом. Існує два типи збірок: поділяються і характерні. Складання, що розділяються, — це звичайні бібліотеки, доступні всьому програмному забезпеченню, тоді як приватні зборки призначені для використання цілком певними програмами. Приватні складання повністю самодостатні, тому процес інсталяції простий.

9) Підтримка Web-служб. .NET пропонує повністю інтегровану підтримку розробки Web-служб — так само просто, як і створення додатків будь-якого іншого типу.

10) Visual Studio 2005. .NET поставляється із середовищем розробки Visual Studio 2005, яке однаково добре справляється з мовами C++, C#, J# та Visual Basic 2005, а також з ASP.NET. Visual Studio 2005 інтегрує у собі всі найкращі засоби відповідних специфічних мовних середовищ Visual Studio .NET 2002/2003 та Visual Studio 6.

11) C#. C# являє собою нову об'єктно-орієнтовану мову, призначену для застосування з .NET.

3.1.2 Переваги Microsoft SQL Server

Для розробки БД було обрано СУБД Microsoft SQL Server 2005.

Microsoft SQL Server – система управління реляційними базами даних (СУБД), розроблена корпорацією Microsoft. Основна мова запитів – Transact-SQL, створена спільно Microsoft і Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO із структурованої мови запитів (SQL) з розширеннями. Використовується до роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД у цьому сегменті ринку.

Основні переваги Microsoft SQL Server:

1) масштабованість;

2) чудова продуктивність;

3) простота використання;

4) готовність до використання в Інтернеті, інтрамережах та для електронної комерції;

5) сховища даних;

6) інтеграція коїться з іншими продуктами Microsoft.

3.2 Особливості програмної розробки

3.2.1 Отримання даних

Дані з пристрою допомагає отримувати стандартну бібліотеку C++, яка додавалася до нього. Імпорт необхідних функцій у C# проект представлений нижче. Усього таких функцій дванадцять.

```
[DllImport("DataLayer.dll", EntryPoint = "Init", CharSet = CharSet.Ansi)]
static extern void Init(string filename);

[DllImport("DataLayer.dll", EntryPoint = "Update", CharSet = CharSet.Unicode)]
new static extern bool Update();
```

Організацію зчитування даних і передачу їх класам, що виконуються, бере на себе клас Engine. Функцію зчитування даних представлено нижче. Операції зчитування даних та виведення їх на екран організовані в окремому потоці.

```
/**
 * @function GetData
 * Function that gets data
 */
public void GetData()
{
   Update();
   WriteToFile(" ");

   nSamples = GetNumberOfSamples();

   signals = new double[nData, nSamples];
   quality = new double[nQuality, nSamples];
   emotion = new double[nEmotion, nSamples];

   if (nSamples != 0)
   {
      for (int i = 0; i < nSamples; i++)
      {
         for (int j = 0; j < nData; j++)
         {
            signals[j, i] = GetContactData(j, i);
         }
         for (int k = 0; k < nQuality; k++)
         {
            quality[k, i] = GetQuality(k);
         }
         for (int q = 0; q < nEmotion; q++)
         {
            emotion[q, i] = GetEmotion(q);
         }
      }
      gyroX = GetGyro(0);
      gyroY = GetGyro(1);
   }
}
```

Інформування класів, що виконуються, про надходження нових даних забезпечує функція “Draw”, яка представлена нижче. Вона ж і передає нові дані класам, що виконуються.

```
/**
 * @function Draw
 * Function that send to windows information about redrawing
 */
public void Draw()
{
```

```
            while (true)
            {
                Thread.Sleep(300);
                if (nSamples != 0)
                {
                    if (sensorsOpen == true)
                    {
                        if (Application.OpenForms["SensorsWindow"] != null)
                        {
                            sensorsWindow.RedrawSensors(signals, quality, nSamples);
                        }
                    }
                    if (gyroOpen == true)
                    {
                        if (Application.OpenForms["GyroWindow"] != null)
                        {
                            gyroWindow.RedrawGyro(gyroX, gyroY, nSamples);
                        }
                    }
                    if (emotionOpen == true)
                    {
                        if (Application.OpenForms["EmotionWindow"] != null)
                        {
                            emotionWindow.RedrawEmotions(emotion);
                        }
                    }
                    if (patternsOpen == true)
                    {
                        if (Application.OpenForms["PatternsWindow"] != null)
                        {
                            patternsWindow.Compare(signals, nSamples);
                        }
                    }
                }
            }
        }
```

3.2.2 Виведення даних

Коли клас “Engine” обробив та передав дані виконуючим класам, а виконуючі класи прийняли дані, то починається робота з їхньої візуалізації. Так, клас SensorsWindow відповідає за виведення на екран даних, отриманих з датчиків. Функція, наведена нижче, будує графіки по всіх датчиках.

```
        /**
         * @function DrawPureGraph
         * Function that draws graphs
         * @param d data pockets from 14 sensors
         * @param l length of the pockets
         */
        public void DrawPureGraph(double[,] d, int l)
        {
            // 1. Drawing graphs
            for (int i = 1; i < l; i++)
            {
                for (int j = 0; j < 14; j++)
                {
                    if (d[j, i] != 0)
                    {
```

```
                if (rButtons[j].Focused == true)
                {
                    LargeSensorChart.Series[0].Name = rButtons[j].Text;
                    LargeSensorChart.Series[0].Points.AddY(d[j, i]);
                }
                charts[j].Series[0].Points.AddY(d[j, i]);
                if (charts[j].Series[0].Points.Count > 100)
                {
                    charts[j].Series[0].Points.RemoveAt(0);
                }
            }
        }
        // 2. Removes points if their count to large
        if (LargeSensorChart.Series[0].Points.Count > 200)
        {
            LargeSensorChart.Series[0].Points.RemoveAt(0);
        }
        if (SensorPortretChart.Series[0].Points.Count > 300)
        {
            SensorPortretChart.Series[0].Points.RemoveAt(0);
        }
    }
}
```

Дані з координатами, прийнятими від гіроскопа, візуалізуються за допомогою функції DrawLocation, яка належить класу GyroWindow. Вона представлена нижче.

```
/**
 * @function DrawLocation
 * Function that draws a location of the point relatively the center
 * @param x coordinate X
 * @param y coordinate Y
 */
public void DrawLocation(double x, double y)
{
    int radius = 150; int mean = 2000;

    // 1. Loading image from file to picturebox
    Bitmap btm = new Bitmap(fileName);
    LocationPictureBox.Image = (Image)btm;

    // 2. Calculating x & y coordinates
    double xLocation = x * radius / mean;
    double yLocation = y * radius / mean;

    // 3. Draws point with x & y coordinates
    Graphics g = Graphics.FromImage(LocationPictureBox.Image);
    g.FillRectangle(Brushes.Red, (int)(xLocation + radius),              (int)(yLocation + radius), 5, 5);
}
```

Клас “EmotionWindow” відповідає за виведення інформації про емоційний стан користувача. Це завдання полягає у виведенні чотирьох показників у вигляді гістограми. Ці чотири показники (емоційні стани) називаються: engagement, meditation, frustration, excitement. Функція, що відображає ці дані, називається "DrawEmotion", вона представлена нижче.

```
/**
 * @function DrawEmotion
 * Function that draws graphs
 * @param emotion pockets of 4 emotions
 * @param l length of the pockets
```

```
        */
        public void DrawEmotion(double[,] emotion, int l)
        {
            int size = 100;

            // 1. Clear previous values
            chart20.Series[0].Points.Clear();
            chart20.Series[1].Points.Clear();
            chart20.Series[2].Points.Clear();
            chart20.Series[3].Points.Clear();

            // 2. Draws graph and change colour of indicator
            for (int i = 0; i < l; i++)
            {
                chart20.Series["Engagement"].Points.AddXY(1, emotion[0, i] * size);
                chart20.Series["Frustration"].Points.AddXY(2, emotion[1, i] * size);
                chart20.Series["Meditation"].Points.AddXY(3, emotion[2, i] * size);
                chart20.Series["Excitement"].Points.AddXY(4, emotion[3, i] * size);
                Indicator.BackColor = Color.FromArgb((int)(emotion[3, i] * 255.0), (int)(emotion[2, i] * 255.0), (int)(emotion[1, i] * 255.0));
            }
            int stop = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second);

            // 3. Select song accordingly user's emotions
            if (fPickUp == true)
            {
                PlaySelectSong(stop);
            }

            // 4. Save name of song and user's emotions
            if (fSave == true)
            {
                if (stop != startSave)
                {
                    SaveEmotions(emotion);
                }
                if (stop == startSave)
                {
                    fSave = false;
                    SaveSong(frustration, meditation, engagement, excitement);
                    song = "";
                }
            }
        }
```

Робота з патернами полягає у їх збереженні у файл та подальшому порівнянні з поточними діями в реальному часі. Це дозволяє зрозуміти, які ділянки мозку відповідають за ту чи іншу дію. Також, за допомогою цього модуля можна перевірити чи схожі мозкові імпульси різних людей.

3.3 Використовувані бібліотеки

using System.Threading - створює та контролює потік, задає пріоритет та повертає статус.

using System.IO – надає статичні методи для створення, копіювання, видалення, переміщення та відкриття файлів, а також допомагає при створенні об'єктів FileStream.

using System.Drawing – простір імен System.Drawing забезпечує доступ до функціональних можливостей графічного інтерфейсу GDI+. Простір імен System.Drawing.Drawing2D, System.Drawing.Imaging, та System.Drawing.Text забезпечують додаткові функціональні можливості.

Клас Graphics надає методи малювання пристрою відображення. Такі класи, як Rectangle та Point, інкапсулюють елементи GDI+. Клас Pen використовується для малювання ліній та кривих, а класи, похідні від абстрактного класу Brush, використовуються для заливання фігур.

using System.Windows.Forms – містить класи для створення програм Windows, які дозволяють найбільш ефективно використовувати розширені можливості користувальницького інтерфейсу, доступні в операційній системі Microsoft Windows.

using System.Linq – містить класи та інтерфейси, які підтримують запити, які використовують LINQ.

using System.Data – забезпечує доступ до класів, що представляють архітектуру ADO.NET. ADO.NET дозволяє створювати компоненти, що ефективно управляють даними з декількох джерел даних.

У сценарії від'єднання, наприклад Інтернет, ADO.NET надає засоби запиту, оновлення та узгодження даних у багаторівневих системах. Архітектура ADO.NET реалізується також у клієнтських додатках, таких як Windows Forms або сторінки HTML, створені в ASP.NET.

Наріжним каменем архітектури ADO.NET є DataSet. Один набір даних DataSet може містити декілька об'єктів DataTable, кожен з яких містить дані з одного джерела даних, наприклад SQL Server.

using System.Collections.Generic – отримає інтерфейси та класи, що визначають універсальні колекції, які дозволяють користувачам створювати строго типізовані колекції, що забезпечують підвищену продуктивність та безпеку типів у порівнянні з неуніверсальними строго типізованими колекціями.

3.4 Опис роботи програми

3.4.1 Візуалізація сигналів датчиків

Ця програма складається з чотирьох модулів: модуль візуалізації сигналів датчиків, модуль візуалізації показань гіроскопа, модуль візуалізації емоційного стану користувача, модуль роботи з патернами.

На малюнку 3.1 показаний модуль, який відповідає за відображення даних із датчиків пристрою. Тут будується чотирнадцять графіків, кожен із яких відповідає одному з датчиків. Для більш детального перегляду будь-якого сигналу є ще один компонент, розташований в нижній частині форми. Також сигнали відображаються двома методами: чисті дані з датчиків і перетворені за

допомогою перетворення Фур'є. Вигляд сигналів можна вибрати зверху форми. Якість сигналу візуалізується за допомогою зелених або червоних кружків (зелений – добрий сигнал, червоний – поганий). Останнє, що виконує даний модуль, це побудова фазового портрета сигналу. Він знаходиться у лівому нижньому кутку форми.

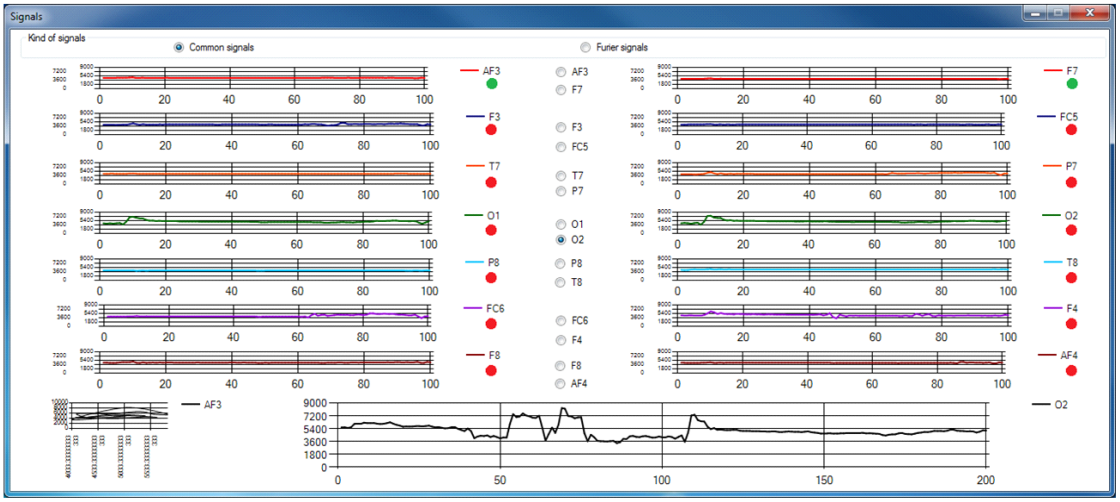


Рисунок 3.1 – Візуалізація сигналів датчиків

Послідовність операцій:

1. Читання даних із 14 датчиків в окремому потоці;
2. Малювання графіків у окремому потоці;
3. Реалізація перетворення Фур'є та виведення результату у вигляді графіків;
4. Читання даних із файлу.

Вхідні дані – сигнали із 14 датчиків.

3.4.2 Візуалізація показань гіроскопа

На малюнку 3.2 показаний модуль, який відповідає за відображення показників гіроскопа.

Тут будується два графіки. Один із них відображає координату X. Другий – координату У. Також будуються два фазові портрети за координатами. Більше наочно зміну координат можна помітити за допомогою червоної точки на координатному полі.

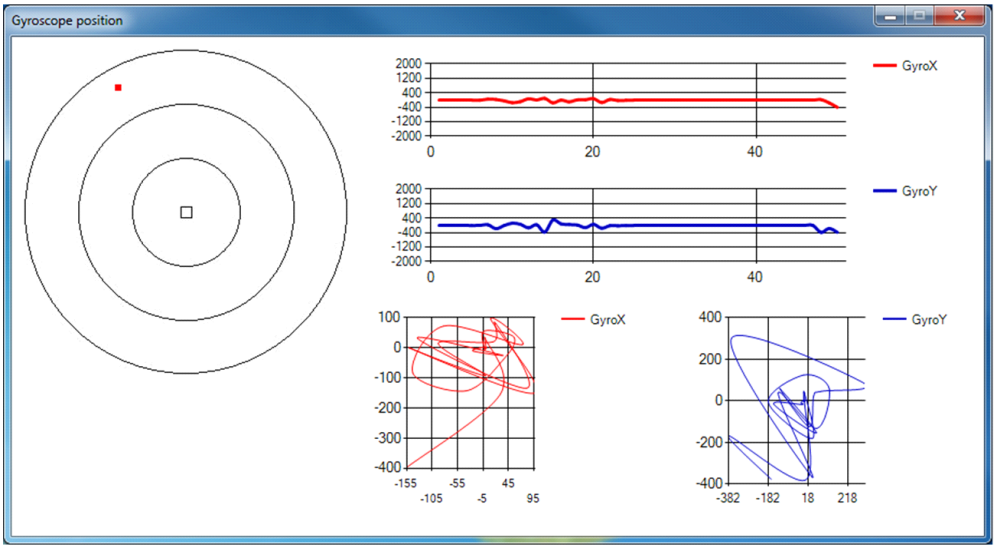


Рисунок 3.2 – Візуалізація показників гіроскопа

Послідовність операцій:

1. Читання даних гіроскопа в окремому потоці;
2. Малювання графіків у окремому потоці.

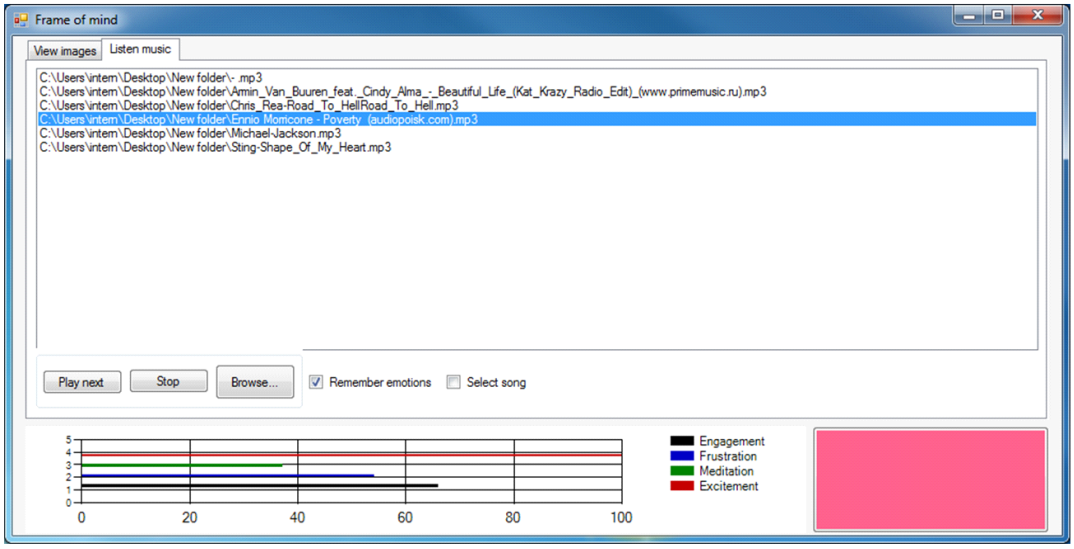
Оскільки гіроскоп є двовісним, то вхідними параметрами це координати X і Y.

3.4.3 Візуалізація емоційного стану

На малюнку 3.3 показаний модуль, який відповідає за відображення емоційного стану користувача.

Тут будується гістограма, що відображає частку присутності всіх чотирьох емоційних станів у користувача (заклопотаність, розчарування, роздуми, збудження). Також цей модуль має можливість завантажувати музику та фото, щоб стежити за реакцією на якусь пісню чи картинку.

Цей модуль пов'язаний з БД, куди записується ім'я користувача, назва пісні та емоційний стан користувача. Це робиться для того, щоб надалі можна було б підібрати пісню під поточний настрій користувача.



Малюнок 3.3 – Візуалізація емоційного стану

Послідовність операцій:
1. Читання значень чотирьох емоцій в окремому потоці;
2. Малювання гістограми в окремому потоці;
3. Можливість завантажувати зображення;
4. Можливість завантажувати та слухати музику.

Вхідні параметри значення чотирьох емоцій.

3.4.4 Модуль роботи з патернами

На малюнку 3.4 показаний модуль, який відповідає за роботу з патернами.

Тут реалізовано функцію запам'ятовування сигналів датчиків та запис їх у файл. Також реалізовано функції порівняння патернів. Якщо індикатор червоний, це означає, що сигнали схожі, якщо білий – навпаки.

Вхідні дані – сигнали із 14 датчиків.

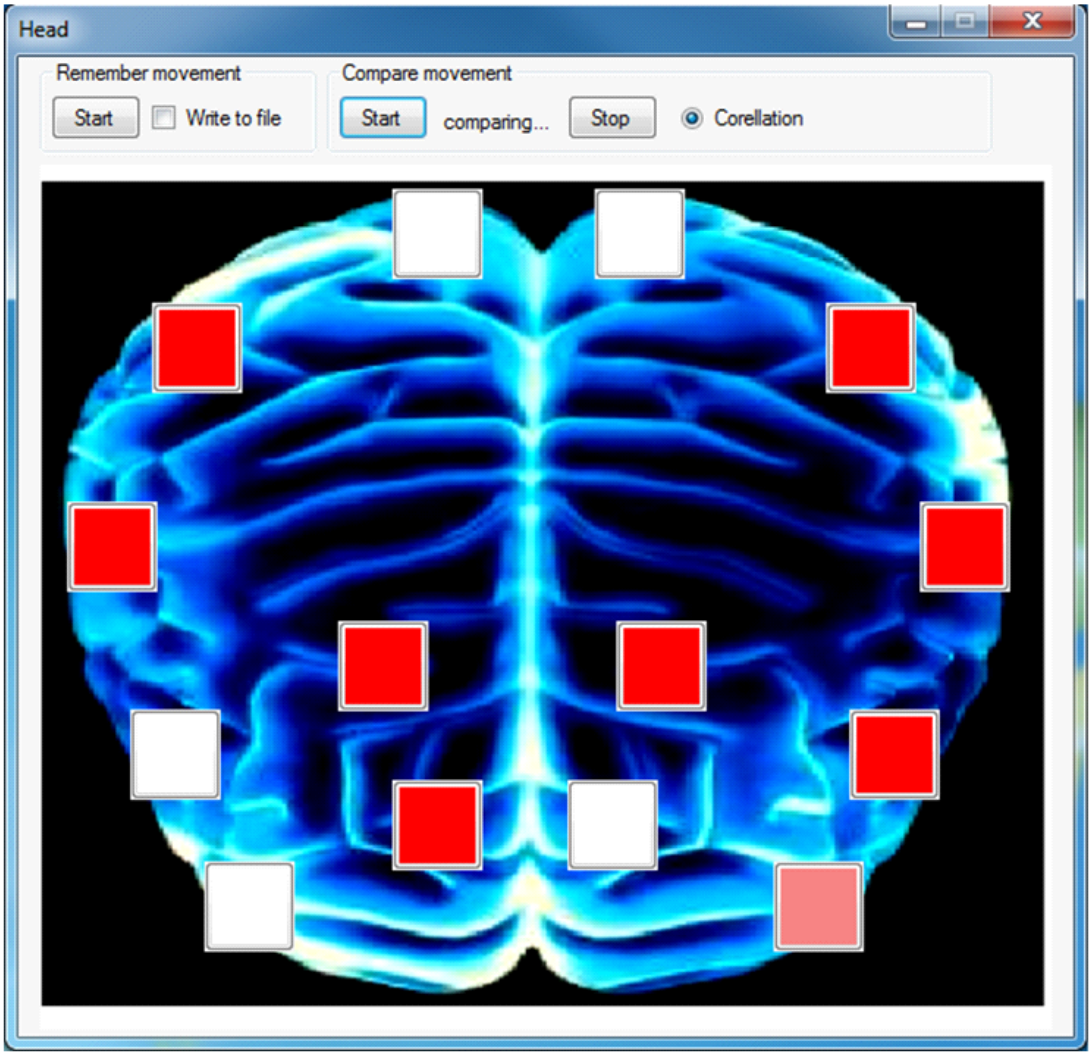


Рисунок 3.4 – Модуль роботи з патернами

4 РОЗРОБКА СТАРТАП ПРОЕКТУ ДОДАТКА ДЛЯ ВИКОРИСТАННЯ НЕЙРОКОМП'ЮТЕРНОГО ІНТЕРФЕЙСУ EMOTIV EPOC

Метою даного розділу дипломного проекту є економічне обґрунтування розробки та впровадження програмного продукту, де проводимо аналіз ефективності функціонування каналів розподільчої логістики.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести сегментацію ринку програмних продуктів та визначити конкурентоспроможність розробленої програми;
- розрахувати трудомісткість виконання основних робіт проекту;
- скласти кошториси витрат на розробку програми;
- розрахувати очікуваний прибуток від реалізації продукції та рентабельність випуску цього виду продукту;

- розрахувати точку беззбитковості та побудувати графік досягнення беззбитковості.

4.1 Опис виробу

Програмний продукт призначений для використання на виробничих підприємствах. Пропонований продукт немає ідентичних аналогів. Перевагами цієї програми є база даних, що зберігає результати проведених дослідів; наочність результатів; ведення статистики, за підсумками якої розробник ухвалює рішення.

Основне призначення системи – перевірка нейрокомп'ютерного інтерфейсу на точність та коректність.

4.2 Сегментація та оцінка ринку збуту

Основними потенційними споживачами даного продукту є науково-дослідні центри та IT-компанії. Продукт, що розробляється, буде поширюватися також серед підприємств, на яких присутні виробничі технологічні лінії. Також продукція має зацікавити підприємства з виробництва медичного обладнання, відеоігор та пересічних фахівців пов'язаних із розробкою програмного забезпечення.

У таблиці 4.1 представлені результати сегментування та ємності ринку збуту продукції.

Матеріали, які використовуються для випуску програмного продукту, не є гостродефіцитними, їх передбачається купувати у традиційних постачальників підприємств, що територіально розташовані в Україні.

Таблиця 4.1 - Сегментування та ємність ринку

Основні замовники	Годовая емкость, шт./год
науково досліднецькі центри	30
IT-компанії	100
підприємства з виробництва медичного обладнання	20
разом	150

Таким чином, прогнозована потреба становить 150 шт./рік.

Для випуску програмного продукту використовуються обладнання та потужності, витрати на модернізацію обладнання мінімальні.

4.3 Аналіз конкурентоспроможності програмного продукту

Конкурентоспроможність товару – це ступінь його відповідності обраному ринку за комерційними, технічними та економічними показниками, що забезпечують можливість збуту товару на цьому ринку. Оскільки в даному програмному продукті зручний інтерфейс у використанні користувача, приведення складського обліку, а також спрощена система розрахунків. Це ті характеристики, які вигідно відрізняють цей товар від товарів-конкурентів.

Проведемо оцінку конкурентоспроможності розробленого програмного продукту.

Технічна прогресивність нового програмного продукту визначається коефіцієнтом еквівалентності. Розрахунок цього коефіцієнта здійснюється шляхом порівняння технічного рівня товару-конкурента та програмного продукту, що розробляється по відношенню до еталонного рівня даного напрямку.

$$K_{ЭК} = \frac{K_{ТН}}{K_{ТБ}}, \qquad (4.1)$$

де $К_{эк}$ – коефіцієнт еквівалентності.

$К_{тн}$ и $К_{тб}$ – коефіцієнти технічного рівня нового та базового програмного продукту, які можна розрахувати за формулою

$$K_T = \sum_{i=1}^{n} \beta \cdot \frac{П_i}{П_Э}, \qquad (4.2)$$

де β - коефіцієнт вагомості і-го технічного параметра, n - число параметрів, $П_i$ - чисельне значення і-го технічного параметра порівнюваного програмного продукту, $П_э$ - чисельне значення і-го технічного параметра еталона.

Як зразок виступатиме програмний продукт з такими параметрами: обсяг пам'яті – 6 Мбайт, час обробки даних – 0,1 з, надійність – 1.

Значення параметрів програмних продуктів, що порівнюються, подамо в таблиці 4.2.

Таблиця 4.2 - Чисельні значення порівнюваних параметрів

Найменування параметру	Вага параметра, β	Значення параметра			Пб/Пэ	Пн/Пэ	β* Пб/Пэ	β* Пн/Пэ
		Пб	Пн	Пэ				
Обсяг пам'яті	0,2	10	8	6	1,6	1,3	0,3	0,3
Час обробки даних	0,3	0,5	0,3	0,12	4,2	2,5	1,26	0,75

Кількість відмов	0,5	2	1	1	0,5	1	0,25	0,5
Разом:							1,81	1,55

Таким чином, коефіцієнти технічного рівня нового програмного продукту за обсягом пам'яті однакові, порівняно з базовим, за часом обробки новий ПП менше на 0,51 і за кількістю відмов базовий продукт менше від нового на 0,25 значення. З таблиці видно, що коефіцієнти технічного рівня нового та базового ПП дорівнюють 1,55 і 1,81 відповідно. Користуючись даними з таблиці 4.2, розрахуємо

$$K_{ЭК} = \frac{1.55}{1.81} = 0.86$$

Далі обчислимо коефіцієнт зміни функціональних можливостей нового програмного продукту за формулою

$$K_{ФВ} = \frac{K_{ФВН}}{K_{ФВБ}}, \qquad (4.3)$$

де $К_{фвн}$, $К_{фвб}$ - бальна оцінка незмірних показників нового та базового виробу відповідно.

Параметри, що описують можливості програмного продукту, наведені в таблиці 4.3.

Таблиця 4.3 – Параметри програмного продукту

Найменування показника	Оцінка базового програмного продукту (бал)	Оцінка нового програмного продукту (бал)
Обсяг пам'яті	3	4
Функціональні можливості	3	5
Швидкодія	3	3
Зручність інтерфейсу	4	4
Ступінь стомлюваності	2	4
Продуктивність праці	3	3
Разом:	18	23

Виходячи з таблиці, можемо сказати що за всіма параметрами оцінка нового програмного продукту вище на 4 бали, ніж оцінка базового програмного продукту, а саме за показниками обсягу пам'яті, функціональними можливостями і ступенем стомлюваності.

За даними таблиці знаходимо коефіцієнт функціональних можливостей.

$$K_{ФВ} = \frac{22}{18} = 1{,}2$$

Коефіцієнт функціональних можливостей перевищує одиницю, тобто новий програмний продукт перевищує за своїми функціональними можливостями базовий у 1,2 рази.

Також конкурентоспроможність нового програмного продукту по відношенню до базового можна оцінити за допомогою інтегрального коефіцієнта конкурентоспроможності, що враховує раніше розраховані показники.

$$K_{И} = K_{ЭК} \cdot K_{ФВ} \cdot K_{Н} / K_{Ц}, \qquad (4.4)$$

де $К_н$ – коефіцієнт відповідності нового програмного продукту нормативам (1), $К_ц$ – коефіцієнт ціни споживання.

Коефіцієнт ціни споживання обчислюється як відношення договірної ціни нового ПП до договірної ціни базового, тобто

$$K_{Ц} = \frac{63625}{89020} = 0{,}72$$

Інтегральний коефіцієнт конкурентоспроможності дорівнює

$$K_{И} = 0{,}86 \cdot 1{,}2 \cdot 1 / 0{,}72 = 1{,}43$$

$К_И$>1, отже новий програмний продукт конкурентоспроможніший, ніж базовий програмний продукт.

Для наочності всі розраховані коефіцієнти занесемо до таблиці 4.4.

Таблиця 4.4 - Показники конкурентоспроможності

Коефіцієнти	Значение
Коефіцієнт еквівалентності	0,86
Коефіцієнт зміни функціональних можливостей	1,2

Коефіцієнт відповідності нормативам	1
Коефіцієнт ціни споживання	0,72
Інтегральний коефіцієнт конкурентоспроможності	1,43

Таким чином, у таблиці представлені показники конкурентоспроможності і виходячи з них, можемо сказати, що новий програмний продукт перевершує за своїми функціональними можливостями базовий у 1,2 рази, коефіцієнт еквівалентності дорівнює 0,86, інтегральний коефіцієнт конкурентоспроможності дорівнює 1,43 та коефіцієнт ціни споживання дорівнює 0,72.

4.4. Переліки робіт для створення програмного продукту

Наведемо переліки виконавців роботи та їх оклади. Тривалість робочого місяця в середньому рахує 22 дні.

Таблиця 4.5 – Склад виконавців роботи

Посади	Посадові оклади, грн.	
	Місячні	Денні
Керівник	15000	680
Програміст	10000	455

Таким чином, ми склали таблицю виконавців роботи, де розрахували денні та місячні оклади керівника та програміста.

Наведемо переліки робіт для розробників. Розрахуємо тривалість розробки за видами робіт. Результати розрахунків містяться у таблиці 4.6

Таблиця 4.6 – Розрахунок трудомісткості робіт

Вид робот	Тривалість, дні	Трудомісткість, (люд/дні)	Виконавець	
			Керівник (люд/дні)	Програ-міст (люд/дні)
Постановка задачі	1	1	+	
Вибір методів розв'язання задач	1	2	+	+

Розробка графіка ходу робіт	1	1	+	
Організаційна підготовка до створення ПП	1	2	+	+
Разробка ТЗ	3	3	+	
Погодження та затвердження ТЗ	3	6	+	+
Підготовчі роботи	2	4	+	+
Розробка алгоритмів	3	6	+	+
Розробка програми	10	10	+	+
Налагодження програми	2	2		+
Прийомо-здавальні роботи	4	8	+	+
Налагодження та випробування	3	6	+	+
Випробування та здавання продукту в експлуатацію	1	2	+	+
Разом	35	53	23	30

За підсумками таблиці можемо сказати, що трудомісткість керівника становить 23 чол/днів, а програміста 21 чол/днів.

Далі обчислимо основну заробітну плату (ОЗП) розробників програмного продукту, з урахуванням трудовитрат, кількості виконавців та середньоденної заробітної плати. Для цього кількість днів, відпрацьованих окремими виконавцями на стадіях, множать на їх денні оклади:

$$ОЗП = \sum_{i=1}^{4} дневн.зп \cdot кол.дней \qquad (4.5)$$

ОЗП=23*680+30*455=29290(грн)

Розрахуємо додаткову заробітну плату (ДЗП):

ДЗП=15% от ОЗП
ДЗП=(5050*15)/100=4393,50

Розрахуємо вартість матеріалів та комплектуючих, необхідних для написання програми (таблиця 4.7)

Таблиця 4.7 - Перелік покупних матеріалів та комплектуючих

Материалы	Кількість	Ціна, грн.	Сума, грн.	Призначення
Диски CD-R	10шт.	2.50	25	Зберігання копій, перенесення програми
Флеш-карта	2шт.	90	180	Зберігання копій, перенесення та збереження доробок
Інтернет послуги (форма безліміт)	1(месяц)	80	80	Пошук літератури, скачування інформації
Бумага (500л.)	1	35.00	35	Документація, роздруківка
Друк документації	500л.	0.35	175	Різний друк
Разом			495.00	

За підсумком можемо сказати, що всього на комплектуючі нам знадобиться 495,00 гривень.

Вартість обладнання 25 000 грн.

Розрахунок собівартості та договірної ціни наводимо в таблиці 4.8

Таблиця 4.8 - Розрахунок собівартості та ціни виробу за статтями

№ П/П	Статті	Сума,грн
1	Основна заробітна плата (ОЗП)	29290
2	Додаткова заробітна плата (15% от ОЗП)	4393.50
3	Відрахування до соціальних фондів	
	Пенсійний фонд (0,332*(ОЗП+ДЗП)	14117.8
	Фонд зайнятості (0,013*(ОЗП+ДЗП)	437.9
	Соц. страхування (0,015*(ОЗП+ДЗП)	505.25
	Страхування від нещасних випадків (0,0086*(ОЗП+ДЗП)	289.7

4	Матеріали та покупні вироби	495.00
5	Витрати на утримання обладнання (40% от ОЗП)	11716
6	Амортизація (25% від ст. обладнання)	6250
7	Позавиробничі витрати (40% от ОЗП)	11716
8	Собівартість (С)	72967.4
9	Прибуток (20% от С)	14593.48
10	Ціна без НДС(П+С)	87560.88
11	НДС (20% от цены без НДС)	17512.2
12	Ціна з НДС (Цена без НДС+ НДС)	105073.1

Таким чином, ми в таблиці представили основні розрахунки собівартості та ціну нашого програмного продукту, а також визначили ціну програмного продукту, яка становить без урахування ПДВ – 87560.88грн., з урахуванням ПДВ – 105073.1грн.

Рентабельність продукції - це відношення загальної суми прибутку до витрат виробництва та реалізації продукції (відносна величина прибутку, що припадає на 1 грн. поточних витрат):

$$P_n = \frac{Ц - С}{С} \cdot 100\ \%, \qquad (4.6)$$

де Ц – ціна одиниці продукції;

С – собівартість одиниці продукції.

Таким чином рентабельність:

$$P_n = \frac{105073.1 - 72967.4}{72967.4} \cdot 100 = 44\ \%.$$

4.5 Розрахунок точки беззбитковості

При впровадженні у виробництво блоку управління важливо знати стане цей виробничий процес рентабельним і чи приноситиме він бажаний прибуток. Для цього необхідно визначити точку беззбитковості (ТБ) та зобразити її графічно. Для підтвердження стійкості проекту необхідно, щоб значення ТБ було меншим від значень номінальних обсягів виробництва. Чим далі від них значення ТБ (у відсотковому співвідношенні), тим стійкіший проект. Проект зазвичай визнається стійким.

Точку беззбитковості можна розрахувати за такою формулою:

$$N_б = \frac{К}{Ц - С}, \qquad (4.7)$$

де К - умовно-постійні витрати, приймаємо рівними ціні плати;

Ц – вартість виробу;

С - Собівартість одиниці виробу.

Розрахована ціна плати є переддоговірною ціною розробника – це мінімально допустима ціна, що враховує кошторис витрат на розробку виробу та прибуток, розрахований за настановним коефіцієнтом рентабельності.

При остаточному призначенні ціни товару необхідно враховувати надбавки, пов'язані зі збутом виробу. Податок на додану вартість приймається 20% від ціни плати:

$$N_б = \frac{87560.88}{105073.1 - 72967.4} = 2{,}73 \approx 3 \ шт.$$

Таким чином, показник точки беззбитковості = 3 штук.

Графічне уявлення Точки Беззбитковості зображено малюнку 4.1.

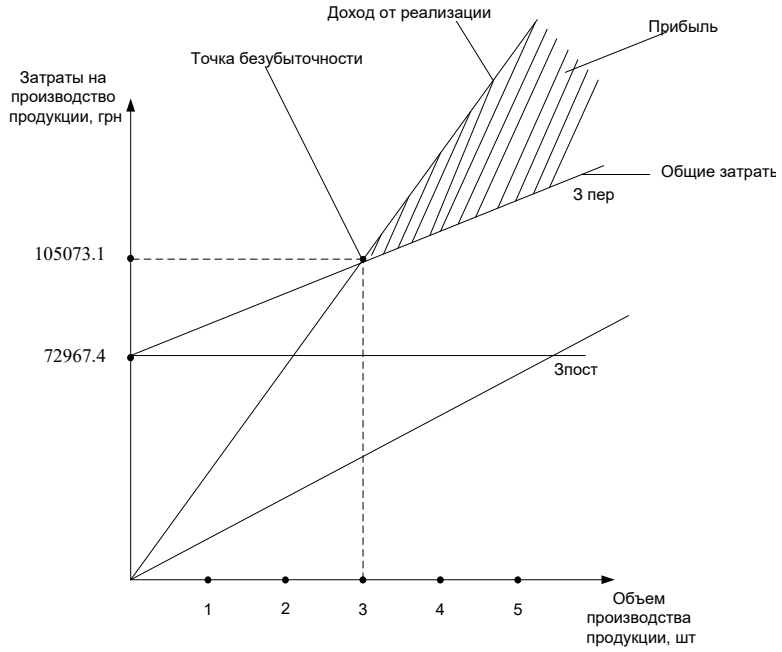


Рисунок 4.1 – Точка Беззбитковості

4.6 Висновок розділу розробки стартапу

У цьому проекті фахівця було досягнуто наступних результатів:

1) проведена сегментація ринку споживачів та збуту програмного продукту та була доведена конкурентоспроможність даного ПП. Коефіцієнт

конкурентоспроможності дорівнює більше одиниці, а прогнозована потреба становить 150 прим./год.

2) розрахована трудомісткість цього товару. Трудомісткість керівника становить 23 чол/днів, а програміста 30 чол/днів.

3) складено кошториси витрат на розробку даного ПП серед яких собівартість 72967.4 грн., ціна виробу 105073.1 грн., витрати на придбання покупних виробів становлять 495,00 грн. інші витрати наведені у таблицях вище;

4) розрахована рентабельність випуску вона становила 44 % і прибуток становив 14593.48 грн;

5) розрахована точка беззбитковості у кількості 3-х шт та побудований графік точки беззбитковості. Результат показують, що після реалізації трьох одиниць продукту, ми покриємо всі витрати на реалізацію і будемо отримувати прибуток.

Даний продукт є актуальним у галузі виробництва, оскільки він допомагає знайти оптимальне рішення за короткий термін. Таким чином, поставлених цілей було досягнуто.

ВИСНОВКИ

Новий комп'ютерний маніпулятор відкриває воістину небачені можливості для розваг та керування різними пристроями.

Для людей творчих він дає можливість зображати свої думки прямо на екрані комп'ютера: створювати динамічні кольори, музику та різні форми.

Emotiv EPOC також допоможе людям з обмеженими можливостями керувати технікою на відстані: керувати інвалідним візком, побутовими приладами, відкрити для себе чудовий світ віртуального простору: грати в комп'ютерні ігри та подорожувати сторінками інтернету.

Зроблено розрахунок собівартості програмного продукту та побудовано графік точки беззбитковості. Здійснено розрахунок заземлювального пристрою, який ефективно захищає від небезпечних струмів та надзвичайної ситуації техногенного характеру, причиною якої є пожежа.

Розроблений додаток дає зрозуміти, що цей пристрій не завжди видає дані безпомилково. Це зумовлено досить великою кількістю перешкод. Перешкоди виникають внаслідок тертя датчиків голову користувача. Але ця проблема вирішувана, так що цей пристрій є досить надійним і перспективним.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Emotiv” http://emotiv.com/
2. “Habrahabr” http://habrahabr.ru/post/115055/
3. “Wikipedia” http://ru.wikipedia.org/wiki/Emotiv_Systems
4. “Wikipedia” http://ru.wikipedia.org/wiki/Нейрокомпьютерный_интерфейс
5. “Neurobotics” http://neurobotics.ru/research/bci/
6. “Computerra” http://www.computerra.ru/tag/neyrokompyuternyiy-interfeys/
7. “Wikia” http://ru.cybernetics.wikia.com/wiki/Нейрокомпьютерный_интерф
8. “BCI Guide” www.emotiv.com/bci-guide/
9. “How BCI can elevate the AR/VR experience” www.emotiv.com/blog/bci-applications-for-vr-ar/
10. “Summary of over Fifty Years with Brain-Computer Interfaces” www.ncbi.nlm.nih.gov/pmc/articles/PMC7824107/
11. “Brain Controlled Technology using Emotiv's Algorithms – EMOTIV” www.emotiv.com/brain-controlled-technology/
12. “Brain-Computer Interface Based on Generation of Visual Images” www.ncbi.nlm.nih.gov/pmc/articles/PMC3112189/

ДОДАТОК А
Лістинг MainWindow.cs

```
/**
* @file MainWindow.cs
* @brief Control window
* @author Viktor Evsikov
* @date Created August 16, 2013 11:40:00
**/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Threading;
using System.IO;

/**
 * MainWindow is a module of control by application. It implement functions that selects data source
 * and open needed windows with needed data.
 */

namespace WindowsFormsApplication1
{
   public partial class MainWindow : Form
    {
```

```
[DllImport("DataLayer.dll", EntryPoint = "Update", CharSet = CharSet.Unicode)]
new static extern bool Update();

[DllImport("DataLayer.dll", EntryPoint = "Init", CharSet = CharSet.Unicode)]
static extern void Init(string filename, int type);

Engine Engine;

SensorsWindow  sWindow;
GyroWindow     gWindow;
EmotionWindow  eWindow;
PatternsWindow pWindow;

public double[,] dataFromFile;

bool fOpenSensors = false, fOpenGyro = false, fOpenEmotions = false, fOpenPatterns = false;
bool fReadFromFile = false, fReadFromES = false;

/**
 * Function that initialize components
 */
public MainWindow(Engine cl)
{
   Engine = cl;
   InitializeComponent();
}




/**
 * Must be called when button clicked
 * Start to read a data from Emotive EPOC
 */
private void ESbutton_Click(object sender, EventArgs e)
{
   fReadFromES = true;

   Engine.StartRead();
   Engine.StartDraw();

   SensorsButton.Enabled = true;
   GyroButton.Enabled    = true;
   EmotionButton.Enabled = true;
   PatternButton.Enabled = true;
}

/**
 * Must be called when button clicked
 * Open file and read data from it
 */
private void FileButton_Click(object sender, EventArgs e)
{
   fReadFromFile = true;

    // 1. Close windows
   if (Application.OpenForms["EmotionWindow"] != null)
   { eWindow.Close(); }
   if (Application.OpenForms["GyroWindow"] != null)
   { gWindow.Close(); }
   if (Application.OpenForms["SensorsWindow"] != null)
   { sWindow.Close(); }
   if (Application.OpenForms["PatternsWindow"] != null)
   { pWindow.Close(); }
```

```
    // 2. Open FileDialog to choose a file with data
    openFileDialog1.ShowDialog();
    string inputFile = openFileDialog1.FileName.ToString();
    Engine.GetDataFromFile(inputFile);

    // 3. Make needed components active and disactive
    if (inputFile != "openFileDialog1")
    {
        SensorsButton.Enabled = true;
        GyroButton.Enabled    = false;
        EmotionButton.Enabled = false;
        PatternButton.Enabled = false;
    }
    else
    {
        SensorsButton.Enabled = false;
        GyroButton.Enabled    = false;
        EmotionButton.Enabled = false;
        PatternButton.Enabled = false;
    }
}
```

```
/**
 * Must be called when button clicked
 * Open SensorsWindow
 */
private void SensorsButton_Click(object sender, EventArgs e)
{
    fOpenSensors = true;
    if (fReadFromFile == true)
    {
        Engine.readFile = true;
        Engine.StartDraw();
    }
    Engine.FormOpen(fOpenSensors, fOpenGyro, fOpenEmotions, fOpenPatterns);
    SensorsButton.Enabled = false;
    fOpenSensors = false;
}

/**
 * Must be called when button clicked
 * Open GyroWindow
 */
private void GyroButton_Click(object sender, EventArgs e)
{
    fOpenGyro = true;
    Engine.FormOpen(fOpenSensors, fOpenGyro, fOpenEmotions, fOpenPatterns);
    GyroButton.Enabled = false;
    fOpenGyro = false;
}

/**
 * Must be called when button clicked
 * Open EmotionWindow
 */
private void EmotionButton_Click(object sender, EventArgs e)
{
```

```
            fOpenEmotions = true;
            Engine.FormOpen(fOpenSensors, fOpenGyro, fOpenEmotions, fOpenPatterns);
            EmotionButton.Enabled = false;
            fOpenEmotions = false;
        }

        /**
         * Must be called when button clicked
         * Open PatternWindow
         */
        private void PatternButton_Click(object sender, EventArgs e)
        {
            fOpenPatterns = true;
            Engine.FormOpen(fOpenSensors, fOpenGyro, fOpenEmotions, fOpenPatterns);
            PatternButton.Enabled = false;
            fOpenPatterns = false;
        }

        /**
         * Must be called when button clicked
         */
        private void MainWindow_FormClosing(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Лістинг SensorsWindow.cs

```
/**
* @file SensorsWindow.cs
* @brief Sensors Visualization
* @author Viktor Evsikov
* @date Created August 17, 2013 9:15:00
**/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Runtime.InteropServices;
using System.Windows.Forms.DataVisualization.Charting;

/**
 * SensorsWindow is a module of application that displayed signals from sensors in graphs.
 * Includes functions to drawing graphs, phase portrait and Fourier transform.
 */

namespace WindowsFormsApplication1
{
    public partial class SensorsWindow : Form
    {
        [DllImport("DataLayer.dll", EntryPoint = "DFTForward", CharSet = CharSet.Unicode)]
        static extern void DFTForward(double[] mas, int length, double[] buf);
```

```
        Image red = Image.FromFile("C:\\Users\\intern\\Documents\\Visual Studio
2012\\Projects\\WindowsFormsApplication1\\WindowsFormsApplication1\\bin\\Release\\red.png");
        Image green = Image.FromFile("C:\\Users\\intern\\Documents\\Visual Studio
2012\\Projects\\WindowsFormsApplication1\\WindowsFormsApplication1\\bin\\Release\\green.png");

        public double[,] signals;
        double[,] quality;
        int pocketLength;
        List<Chart> charts = new List<Chart>();
        List<RadioButton> rButtons = new List<RadioButton>();
        List<PictureBox> pictureBoxes = new List<PictureBox>();

        public bool fReadFromF = false, fMakeInvoke = true, fFurier = false, fPure = true;

        /**
         * Function that initialize components
         */
        public SensorsWindow()
        {
            InitializeComponent();
            pictureBoxes.Add(SensorAF3PictureBox);
            pictureBoxes.Add(SensorF7PictureBox);
            pictureBoxes.Add(SensorF3PictureBox);
            pictureBoxes.Add(SensorFC5PictureBox);
            pictureBoxes.Add(SensorT7PictureBox);
            pictureBoxes.Add(SensorP7PictureBox);
            pictureBoxes.Add(SensorO1PictureBox);
            pictureBoxes.Add(SensorO2PictureBox);
            pictureBoxes.Add(SensorP8PictureBox);
            pictureBoxes.Add(SensorT8PictureBox);
            pictureBoxes.Add(SensorFC6PictureBox);
            pictureBoxes.Add(SensorF4PictureBox);
            pictureBoxes.Add(SensorF8PictureBox);
            pictureBoxes.Add(SensorAF4PictureBox);

            rButtons.Add(AF3radioButton);
            rButtons.Add(F7radioButton);
            rButtons.Add(F3radioButton);
            rButtons.Add(FC5radioButton);
            rButtons.Add(T7radioButton);
            rButtons.Add(P7radioButton);
            rButtons.Add(O1radioButton);
            rButtons.Add(O2radioButton);
            rButtons.Add(P8radioButton);
            rButtons.Add(T8radioButton);
            rButtons.Add(FC6radioButton);
            rButtons.Add(F4radioButton);
            rButtons.Add(F8radioButton);
            rButtons.Add(AF4radioButton);

            charts.Add(SensorAF3Chart);
            charts.Add(SensorA7Chart);
            charts.Add(SensorF3Chart);
            charts.Add(SensorAC5Chart);
            charts.Add(SensorT7Chart);
            charts.Add(SensorP7Chart);
            charts.Add(SensorO1Chart);
            charts.Add(SensorO2Chart);
            charts.Add(SensorP8Chart);
            charts.Add(SensorT8Chart);
            charts.Add(SensorFC6Chart);
            charts.Add(SensorF4Chart);
            charts.Add(SensorF8Chart);
            charts.Add(SensorAF4Chart);

            SetAmplitude();
```

```
}

/**
 * @function RedrawSensors
 * Function that responsible for updating window
 * @param s data pockets from 14 sensors
 * @param q quality pockets from 14 sensors
 * @param l length of the pockets
 */
public void RedrawSensors(double[,] s, double[,] q, int l)
{
   // 1. Preparing data
   signals = s;
   quality = q;
   pocketLength = l;

   // 2. Updating if window available
   if (fMakeInvoke == true)
   {
      this.Invoke(new Action(RefreshSignals));
   }
}




/**
 * @function RefreshSignals
 * Function that responsible for refreshing graphs
 */
public void RefreshSignals()
{
   if (fReadFromF == false)
   { ShowQuality(quality, pocketLength); }
   if (fReadFromF == true)
   { signals = TurnArray(signals, pocketLength); }
   if (fPure == true)
   {
      DrawPureGraph(signals, pocketLength);
      PurePhasePortrait(signals, pocketLength, pocketLength / 2);
   }
   if (fFurier == true)
   { DrawFourierGraph(signals, pocketLength); }
}

/**
* @function TurnArray
* Function that turns array from vertical to horizontal
* @param d vertical array
* @return arr horizontal array
*/
public double[,] TurnArray(double[,] d, int l)
{
   double[,] arr = new double[14,l];
   for (int j = 0; j < 14; j++)
   {
      for (int i = 0; i < l; i++)
      {
         arr[j,i] = d[i,j];
      }
   }
   return arr;
}
```

```
/**
 * @function ShowQuality
 * Function that shows quality of sygnals
 * @param d quality pockets from 14 sensors
 * @param l length of the pockets
 */
public void ShowQuality(double[,] d, int l)
{
   for (int i = 0; i < l; i++)
   {
      for (int j = 0; j < 14; j++)
      {
         if (d[j, i] == 0)
         {
            pictureBoxes[j].Image = red;
         }
         else
         {
            pictureBoxes[j].Image = green;
         }
      }
   }
}
```

```
/**
 * @function DrawGraph
 * Function that draws graphs
 * @param d data pockets from 14 sensors
 * @param l length of the pockets
 */
public void DrawPureGraph(double[,] d, int l)
{
   // 1. Drawing graphs
   for (int i = 1; i < l; i++)
   {
      for (int j = 0; j < 14; j++)
      {
         if (d[j, i] != 0)
         {
            if (rButtons[j].Focused == true)
            {
               LargeSensorChart.Series[0].Name = rButtons[j].Text;
               LargeSensorChart.Series[0].Points.AddY(d[j, i]);
            }
            charts[j].Series[0].Points.AddY(d[j, i]);
            if (charts[j].Series[0].Points.Count > 100)
            {
               charts[j].Series[0].Points.RemoveAt(0);
            }
         }
      }

      // 2. Removes points if their count to large
      if (LargeSensorChart.Series[0].Points.Count > 200)
      {
         LargeSensorChart.Series[0].Points.RemoveAt(0);
      }
      if (SensorPortretChart.Series[0].Points.Count > 300)
      {
         SensorPortretChart.Series[0].Points.RemoveAt(0);
      }
   }
```

```
}

/**
 * @function SetAmplitude
 * Function that sets amplitude of graphs
 */
public void SetAmplitude()
{
    // 1. sets amplitude if selected pure signal
    if (fPure == true)
    {
        for (int i = 0; i < 14; i++)
        {
            charts[i].ChartAreas["ChartArea1"].AxisY.Maximum = 9000;
            charts[i].ChartAreas["ChartArea1"].AxisY.Minimum = 0;
            LargeSensorChart.ChartAreas["ChartArea1"].AxisY.Maximum = 9000;
            LargeSensorChart.ChartAreas["ChartArea1"].AxisY.Minimum = 0;
        }
    }

    // 2. sets amplitude if selected Fourier signal
    if (fFurier == true)
    {
        for (int i = 0; i < 14; i++)
        {
            charts[i].ChartAreas["ChartArea1"].AxisY.Maximum = 9000;
            charts[i].ChartAreas["ChartArea1"].AxisY.Minimum = -9000;
            LargeSensorChart.ChartAreas["ChartArea1"].AxisY.Maximum = 9000;
            LargeSensorChart.ChartAreas["ChartArea1"].AxisY.Minimum = -9000;
        }
    }
}

/**
 * @function FourierPhasePortret
 * Function that draws Phase Portret for Fourier signals
 * @param d data pockets from 1 sensor
 * @param l length of the pocket
 * @param m step of the phase portrait
 */
public void FourierPhasePortrait(double[] d, int l, int m)
{
    // 1. Drawing phase portrait graph
    for (int i = 0; i < 14; i++)
    {
        if (rButtons[i].Focused == true)
        {
            for (int j = m; j < l; j += m)
            {
                SensorPortretChart.Series[0].Name = AF3radioButton.Text;
                SensorPortretChart.Series[0].Points.AddXY(d[j - m], d[j]);
            }
        }
    }

    // 2. Removes points if their count to large
    if (SensorPortretChart.Series[0].Points.Count > 50)
    {
        SensorPortretChart.Series[0].Points.RemoveAt(0);
    }
}

/**
 * @function PurePhasePortrait
 * Function that draws Phase Portret for pure signals
 * @param d data pockets from 14 sensors
 * @param l length of the pockets
```

```
 * @param m step of the phase portrait
 */
public void PurePhasePortrait(double[,] d, int l, int m)
{
   // 1. Drawing phase portrait graph
   for (int i = 0; i < 14; i++)
   {
      if (rButtons[i].Focused == true)
      {
         for (int j = m; j < l; j += m)
         {

            SensorPortretChart.Series[0].Name = AF3radioButton.Text;
            SensorPortretChart.Series[0].Points.AddXY(d[i, j - m], d[i, j]);
         }
      }
   }

   // 2. Removes points if their count to large
   if (SensorPortretChart.Series[0].Points.Count > 50)
   {
      SensorPortretChart.Series[0].Points.RemoveAt(0);

   }
}


/**
 * @function DrawFourier
 * Function that draws Phase Portret for pure signals
 * @param d data pockets from 14 sensors
 * @param l length of the pockets
 */
public void DrawFourierGraph(double[,] d, int l)
{
   for (int i = 0; i < 14; i++)
   {
      double[] signal = new double[l];
      for (int j = 0; j < l; j++)
      {
         signal[j] = d[i, j];
      }
      double[] fSignal = new double[l];
      DFTForward(signal, l, fSignal);
      FourierPhasePortrait(fSignal, l, l / 2);
      for (int j = 1; j < l; j++)
      {
         charts[i].Series[0].Points.AddY(fSignal[j]);
         if (charts[i].Series[0].Points.Count > 100)
         {
            charts[i].Series[0].Points.RemoveAt(0);
         }
         for (int k = 0; k < 14; k++)
         {
            if (rButtons[i].Focused == true)
            {
               LargeSensorChart.Series[0].Name = rButtons[i].Text;
               LargeSensorChart.Series[0].Points.AddY(fSignal[j]);
               if (LargeSensorChart.Series[0].Points.Count > 300)
               {
                  LargeSensorChart.Series[0].Points.RemoveAt(0);
               }
            }
         }
      }
   }
```

```
}
/**
 * Must be called when window closing
 */
private void SensorsWindow_FormClosed(object sender, FormClosedEventArgs e)
{
    fMakeInvoke = false;
}
/**
 * Must be called when AF3radioButton selected
 */
private void AF3radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}
/**
 * Must be called when F7radioButton selected
 */
private void F7radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when F3radioButton selected
 */
private void F3radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when FC5radioButton selected
 */
private void FC5radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when T7radioButton selected
 */
private void T7radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when P7radioButton selected
 */
private void P7radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when O1radioButton selected
 */
private void O1radioButton_CheckedChanged(object sender, EventArgs e)
```

```
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when O2radioButton selected
 */
private void O2radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}


/**
 * Must be called when P8radioButton selected
 */
private void P8radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}



/**
 * Must be called when T8radioButton selected
 */
private void T8radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when FC6radioButton selected
 */
private void FC6radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when F4radioButton selected
 */
private void F4radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when F8radioButton selected
 */
private void F8radioButton_CheckedChanged(object sender, EventArgs e)
{
    SensorPortretChart.Series[0].Points.Clear();
    Thread.Sleep(100);
}

/**
 * Must be called when AF4radioButton selected
 */
private void AF4radioButton_CheckedChanged(object sender, EventArgs e)
{
```

```
            SensorPortretChart.Series[0].Points.Clear();
            Thread.Sleep(100);
        }

        /**
         * Must be called when CommonRadioButton selected
         * Draw pure signals
         */
        private void CommonRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            if (CommonRadioButton.Focused == true)
            {
                fPure = true;
                fFurier = false;
                SetAmplitude();
            }
        }



        /**
         * Must be called when FourierRadioButton selected
         * Draw Fourier signals
         */
        private void FourierRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            if (FourierRadioButton.Focused == true)
            {
                fFurier = true;
                fPure = false;
                SetAmplitude();
            }
        }
    }
}
```

Лістинг GyroWindow.cs

```
/**
* @file GyroWindow.cs
* @brief Gyroscope visualization
* @author Viktor Evsikov
* @date Created August 17, 2013 10:45:00
**/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Runtime.InteropServices;
using System.Windows.Forms.DataVisualization.Charting;

/**
 * GyroWindow is a module that displayed coordinates X and Y.
 * It implement functions draw graph, draw phase portret and draws a location of the point relatively the center.
 */

namespace WindowsFormsApplication1
{
   public partial class GyroWindow : Form
   {
      string fileName = "C:\\Users\\intern\\Documents\\Visual Studio
2012\\Projects\\WindowsFormsApplication1\\WindowsFormsApplication1\\bin\\Release\\image.jpg";
      double xPrevious = 0, yPrevious = 0, xCoordinate, yCoordinate;
      int pocketCount;
      bool fMakeInvoke = true;

      /**
       * Function that initialize components
       */
      public GyroWindow()
      {
         InitializeComponent();
      }
      /**
       * @function RedrawGyro
       * Function that responsible for updating window
       * @param x coordinate X
       * @param y coordinate Y
       */
      public void RedrawGyro(double x, double y, int l)
      {
         // 1. Preparing data
         xCoordinate = x;
         yCoordinate = y;
         pocketCount = l;

         // 2. Updating if window available
         if (fMakeInvoke == true)
         {
            this.Invoke(new Action(RefreshGraphs));
```

```
    }
}

/**
 * @function RefreshGraphs
 * Function that responsible for refreshing graphs
 */
public void RefreshGraphs()
{
    SetAmplitude();
    DrawGyroGraph(xCoordinate, yCoordinate);
    DrawLocation(xCoordinate, yCoordinate);
    DrawGyroPortrait(xCoordinate, yCoordinate, xPrevious, yPrevious);
    xPrevious = xCoordinate;
    yPrevious = yCoordinate;
}

/**
 * @function DrawGyroGraph
 * Function that draws graphs of coordinates X and Y
 * @param x coordinate X
 * @param y coordinate Y
 */
public void DrawGyroGraph(double x, double y)
{
    // 1. Adds points to graph
    GyroXchart.Series[0].Points.AddY(x);
    GyroYchart.Series[0].Points.AddY(y);

    // 2. Removes points if their count more than 50
    if (GyroXchart.Series[0].Points.Count > 50)
    {
        GyroXchart.Series[0].Points.RemoveAt(0);
        GyroYchart.Series[0].Points.RemoveAt(0);
    }
}

/**
 * @function DrawGyroPortrait
 * Function that draws a phase portrait of coordinates X and Y
 * @param x coordinate X
 * @param y coordinate Y
 * @param xPr previous value of coordinate X
 * @param yPr previous value of coordinate Y
 */
public void DrawGyroPortrait(double x, double y, double xPr, double yPr)
{
    PortraitXchart.Series[0].Points.AddXY(xPr, x);
    PortraitYchart.Series[0].Points.AddXY(yPr, y);
    if (PortraitXchart.Series[0].Points.Count > 100)
    {
        PortraitXchart.Series[0].Points.RemoveAt(0);
        PortraitYchart.Series[0].Points.RemoveAt(0);
    }
}
```

```
        /**
         * @function SetAmplitude
         * Function that sets amplitude of graphs
         */
        public void SetAmplitude()
        {
            GyroXchart.ChartAreas["ChartArea1"].AxisY.Maximum = 2000;
            GyroXchart.ChartAreas["ChartArea1"].AxisY.Minimum = -2000;
            GyroYchart.ChartAreas["ChartArea1"].AxisY.Maximum = 2000;
            GyroYchart.ChartAreas["ChartArea1"].AxisY.Minimum = -2000;
        }

        /**
         * @function DrawLocation
         * Function that draws a location of the point relatively the center
         * @param x coordinate X
         * @param y coordinate Y
         */
        public void DrawLocation(double x, double y)
        {
            int radius = 150;
            int mean = 2000;

            // 1. Loading image from file to picturebox
            Bitmap btm = new Bitmap(fileName);
            LocationPictureBox.Image = (Image)btm;

            // 2. Calculating x & y coordinates
            double xLocation = x * radius / mean;
            double yLocation = y * radius / mean;

            // 3. Draws point with x & y coordinates
            Graphics g = Graphics.FromImage(LocationPictureBox.Image);
            g.FillRectangle(Brushes.Red, (int)(xLocation + radius), (int)(yLocation + radius), 5, 5);
        }

        /**
         * Must be called when window closing
         */
        private void GyroWindow_FormClosed(object sender, FormClosedEventArgs e)
        {
            fMakeInvoke = false;
        }
    }
}
```

Лістинг EmotionWindow.cs

```
/**
* @file EmotionWindow.cs
* @brief Emotions Analysis
* @author Viktor Evsikov
* @date Created August 20, 2013 10:20:00
**/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Runtime.InteropServices;
using System.Windows.Forms.DataVisualization.Charting;
using System.IO;
using Microsoft.DirectX.AudioVideoPlayback;
using Microsoft.DirectX;

/**
 * EmotionWindow is a module that charged drawing emotion graph and analyses user's emotions on reaction to the different mediafiles.
 * It implement functions draw graph, upload mediafiles, saving user's emotions and selecting music accordingly user's emotions.
 */

namespace WindowsFormsApplication1
{
   public partial class EmotionWindow : Form
   {
      [DllImport("winmm.dll")]
      private static extern long mciSendString(string strCommand, StringBuilder strReturn, int iReturnLength, IntPtr hwndCallback);
      private string sCommand = "";

      [DllImport("DataLayer.dll", EntryPoint = "AddSong", CharSet = CharSet.Ansi)]
      static extern bool AddSong(double[] frustration, double[] meditation, double[] boredom, double[] excitement, int length, string song);

      [DllImport("DataLayer.dll", EntryPoint = "GetSong", CharSet = CharSet.Ansi)]
      static extern bool GetSong(double[] frustration, double[] meditation, double[] boredom, double[] excitement, int length, StringBuilder song, double alfa);


      OpenFileDialog open = new OpenFileDialog();
      FilePLS[] PlayFile;

      List<double> frustration = new List<double>();
      List<double> meditation  = new List<double>();
      List<double> engagement  = new List<double>();
      List<double> excitement  = new List<double>();

      int startSave, startPickUp, pocketLength, fotoCount = 0;

      bool fSave      = false;
      bool fPickUp    = false;
```

```
bool fMakeInvoke = true;

string song;
Random rnd;
string[] foto;
double[,] emotion;

/**
 * Function that initialize components
 */
public EmotionWindow()
{
   InitializeComponent();
   chart20.ChartAreas["ChartArea1"].AxisY.Maximum = 100;

   chart20.Series[0].ChartType = SeriesChartType.Bar;
   chart20.Series[1].ChartType = SeriesChartType.Bar;
   chart20.Series[2].ChartType = SeriesChartType.Bar;
   chart20.Series[3].ChartType = SeriesChartType.Bar;
}

/**
 * @function RedrawEmotions
 * Function that responsible for updating window
 * @param d pockets of 4 emotions
 */
public void RedrawEmotions(double[,] d)
{
   emotion = d;
   pocketLength = emotion.Length / 4;
   if (fMakeInvoke == true)
   {
      this.Invoke(new Action(RefreshEmotion));
   }
}

/**
 * @function RefreshEmotion
 * Function that draw emotions to chart in real time
 */
public void RefreshEmotion()
{
   DrawEmotion(emotion, pocketLength);
}

/**
 * @function DrawEmotion
 * Function that draws graphs
 * @param emotion pockets of 4 emotions
 * @param l length of the pockets
 */
public void DrawEmotion(double[,] emotion, int l)
{
   int size = 100;

   // 1. Clear previous values
   chart20.Series[0].Points.Clear();
   chart20.Series[1].Points.Clear();
   chart20.Series[2].Points.Clear();
   chart20.Series[3].Points.Clear();

   // 2. Draws graph and change colour of indicator
   for (int i = 0; i < l; i++)
   {
      chart20.Series["Engagement"].Points.AddXY(1, emotion[0, i] * size);
      chart20.Series["Frustration"].Points.AddXY(2, emotion[1, i] * size);
```

```
            chart20.Series["Meditation"].Points.AddXY(3, emotion[2, i] * size);
            chart20.Series["Excitement"].Points.AddXY(4, emotion[3, i] * size);
            Indicator.BackColor = Color.FromArgb((int)(emotion[3, i] * 255.0), (int)(emotion[2, i] * 255.0), (int)(emotion[1, i]
* 255.0));
        }
        int stop = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second);

        // 3. Select song accordingly user's emotions
        if (fPickUp == true)
        {
            PlaySelectSong(stop);
        }

        // 4. Save name of song and user's emotions
        if (fSave == true)
        {
            if (stop != startSave)
            {
                SaveEmotions(emotion);
            }
            if (stop == startSave)
            {
                fSave = false;
                SaveSong(frustration, meditation, engagement, excitement);
                song = "";
            }
        }
    }

    /**
     * @function SelectSong
     * Function that plays selected song
     * @param stop time of selecting
     */
    public void PlaySelectSong(int stop)
    {
        if (stop != startPickUp)
        {
            SaveEmotions(emotion);
        }
        if (stop == startPickUp)
        {
            startPickUp = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) + 10;
            string pUSong = SelectSong(frustration, meditation, engagement, excitement);

            if (pUSong != "noSong")
            {
                for (int i = 0; i < MusicList.Items.Count; i++)
                {
                    if (MusicList.Items[i].ToString().IndexOf(pUSong) > -1)
                    {
                        fPickUp = false;
                        SelectCheckBox.Checked = false;
                        MusicList.SetSelected(i, true);
                        string music = MusicList.Items[i].ToString();
                        try
                        {
                            sCommand = "stop MediaFile";
                            mciSendString(sCommand, null, 0, IntPtr.Zero);
                            sCommand = "close MediaFile";
                            mciSendString(sCommand, null, 0, IntPtr.Zero);

                            Thread.Sleep(1000);

                            sCommand = "open \"" + music + "\" type mpegvideo alias MediaFile";
```

```
                    try
                    {
                        Thread.Sleep(1000);
                        mciSendString(sCommand, null, 0, IntPtr.Zero);
                    }
                    catch (Exception w)
                    {
                        startPickUp = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second)
+ 10;
                    }
                    sCommand = "play MediaFile";
                    mciSendString(sCommand, null, 0, IntPtr.Zero);
                }
                catch (Exception q)
                { startPickUp = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) +
10; }
            }
        }
    }
}

/**
 * @function RefreshList
 * Function that load music to the list
 */
public void RefreshList()
{
    MusicList.Items.Clear();
    for (int i = 0; i < PlayFile.Length; i++)
        MusicList.Items.Add(PlayFile[i].Path);
}

/**
 * @function SaveEmotions
 * Function that prepare data to saving
 * @param emotions pockets of 4 emotions
 */
public void SaveEmotions(double[,] emotions)
{
    for (int i = 0; i < pocketLength; i++)
    {
        frustration.Add(emotions[2, i]);
        meditation.Add(emotions[1, i]);
        engagement.Add(emotions[3, i]);
        excitement.Add(emotions[0, i]);
    }
}


/**
 * @function SaveSong
 * Function that recording data about song to file
 * @param frust one pocket of frustration emotion
 * @param med one pocket of meditation emotion
 * @param engage one pocket of engagement emotion
 * @param excite one pocket of excitement emotion
 */
public void SaveSong(List<double> frust, List<double> med, List<double> engage, List<double> excite)
{
    int length = frust.Count();
    double[] frustration = new double[length];
    double[] meditation  = new double[length];
    double[] engagement  = new double[length];
    double[] excitement  = new double[length];
```

```
    frustration = frust.ToArray();
    meditation  = med.ToArray();
    engagement  = engage.ToArray();
    excitement  = excite.ToArray();

    song = song.Remove(0,25);
    AddSong(frustration, meditation, engagement, excitement, length, song);
}

/**
 * @function SelectSong
 * Function that selected song from the list accordingly user's emotions
 * @param frust one pocket of frustration emotion
 * @param med one pocket of meditation emotion
 * @param engage one pocket of engagement emotion
 * @param excite one pocket of excitement emotion
 * @return retSong name of the selected song
 */
public string SelectSong(List<double> frust, List<double> med, List<double> engage, List<double> excite)
{
    string retSong;
    bool f = false;

    int length = frust.Count();
    double[] frustration = new double[length];
    double[] meditation  = new double[length];
    double[] engagement  = new double[length];
    double[] excitement  = new double[length];

    frustration = frust.ToArray();
    meditation  = med.ToArray();
    engagement  = engage.ToArray();
    excitement  = excite.ToArray();

    StringBuilder song = new StringBuilder(100);
    f = GetSong(frustration, meditation, engagement, excitement, length, song, 1.0);
    if (f == true)
    {
        retSong = song.ToString();
    }
    else
    {
        retSong = "noSong";
    }
    startPickUp = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) + 8;
    retSong = retSong.Remove(0,10);
    return retSong;
}


/**
 * Must be called when window closing
 * Stop invoke
 */
private void EmotionWindow_FormClosed(object sender, FormClosedEventArgs e)
{
    fMakeInvoke = false;
}

/**
 * Must be called when button clicked
 * Load foto to picturebox
 */
private void LoadFotoButton_Click(object sender, EventArgs e)
{
    fotoCount = 0;
```

```
    try
    {
        folderBrowserDialog1.ShowDialog();
        foto = Directory.GetFiles(folderBrowserDialog1.SelectedPath, "*.jpg");
        ImagePictureBox.Load(foto[0]);
    }
    catch (Exception w)
    { }
}

/**
 * Must be called when button clicked
 * Browse music to listbox
 */
private void BrowseMusicButton_Click(object sender, EventArgs e)
{
    open.Multiselect = true;
    open.Filter = "Музыка (*.mp3)|*.mp3";
    if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        string[] f = open.FileNames;
        PlayFile = new FilePLS[f.Length];
        for (int i = 0; i < PlayFile.Length; i++)
        {
            FileInfo file = new FileInfo(f[i]);
            PlayFile[i]   = new FilePLS(file.Name, file.FullName);
        }
        RefreshList();
    }
}

/**
 * Must be called when button clicked
 * Stop audiofile
 */
private void StopButton_Click(object sender, EventArgs e)
{
    sCommand = "close MediaFile";
    mciSendString(sCommand, null, 0, IntPtr.Zero);
    fSave = false;
}



/**
 * Must be called when checkBox changed
 */
private void RememberCheckBox_CheckedChanged(object sender, EventArgs e)
{
    if (RememberCheckBox.Checked == true)
    {
        fSave = true;
        SelectCheckBox.Checked = false;
    }
    else
    { fSave = false; }
}

/**
 * Must be called when checkBox changed
 */
private void SelectCheckBox_CheckedChanged(object sender, EventArgs e)
{
    if (SelectCheckBox.Checked == true)
    {
        sCommand = "stop MediaFile";
```

```
            mciSendString(sCommand, null, 0, IntPtr.Zero);
            sCommand = "close MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);

            fPickUp = true;
            startPickUp = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) + 5;
            RememberCheckBox.Checked = false;
        }
        else { fPickUp = false; }
    }

    /**
     * Must be called when window closing
     */
    private void EmotionWindow_FormClosing(object sender, FormClosingEventArgs e)
    {
        fSave = false;
        sCommand = "close MediaFile";
        mciSendString(sCommand, null, 0, IntPtr.Zero);
    }

    /**
     * Must be called when button clicked
     */
    private void PlayNextButton_Click(object sender, EventArgs e)
    {
        try
        {
            // 1. Stop audiofile
            sCommand = "stop MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);
            sCommand = "close MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);

            // 2. Select next song
            rnd = new Random();
            int number = rnd.Next(0, MusicList.Items.Count);
            MusicList.SetSelected(number, true);
            song = MusicList.Items[number].ToString();

            // 3. Play next song
            sCommand = "open \"" + song + "\" type mpegvideo alias MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);
            sCommand = "play MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);

            if (RememberCheckBox.Checked == true)
            {
                fSave = true;
            }
            if (fSave == true)
            {
                startSave = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) + 180;
            }
        }
        catch (Exception w)
        { }
    }

    /**
     * Must be called when double clicked on the list
     */
    private void MusicList_DoubleClick(object sender, EventArgs e)
    {
        try
        {
```

```
            // 1. Stop audiofile
            sCommand = "stop MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);
            sCommand = "close MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);

            // 2. Select song
            song = MusicList.SelectedItem.ToString();

            // 3. Play next song
            sCommand = "open \"" + song + "\" type mpegvideo alias MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);
            sCommand = "play MediaFile";
            mciSendString(sCommand, null, 0, IntPtr.Zero);

            if (RememberCheckBox.Checked == true)
            {
                fSave = true;
            }
            if (fSave == true)
            {
                startSave = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) + 180;
            }
        }
        catch (Exception w)
        { }
    }

    /**
     * Must be called when clicked on the pictureBox
     */
    private void ImagePictureBox_Click(object sender, EventArgs e)
    {
        Thread.Sleep(50);
        try
        {
            if (fotoCount == foto.Count())
            {
                fotoCount = 0;
            }
            ImagePictureBox.Load(foto[fotoCount]);
            fotoCount += 1;
        }
        catch (Exception q)
        { fotoCount += 1; }
    }
  }
}
```

Лістинг PatternsWindow.cs

```
/**
* @file PatternsWindow.cs
* @brief Patterns Analysis
* @author Viktor Evsikov
* @date Created August 19, 2013 16:15:00
**/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using System.Runtime.InteropServices;
using System.Threading;
using System.IO;
using System.Timers;

/**
 * PatternsWindow is a module of application that work with patterns.
 * Includes functions are save patterns, recording to the file patterns and compare patterns.
 */

namespace WindowsFormsApplication1
{
   public partial class PatternsWindow : Form
   {
      [DllImport("DataLayer.dll", EntryPoint = "CorellationSize", CharSet = CharSet.Unicode)]
      static extern double CorellationSize(double[] A, double[] B, int sizeA, int sizeB, double e);

      [DllImport("DataLayer.dll", EntryPoint = "CreatePattern", CharSet = CharSet.Ansi)]
      static extern void CreatePattern(string filename, int time);

      [DllImport("FuzzyLogic.dll", EntryPoint = "Equiv", CharSet = CharSet.Unicode)]
      static extern double Equiv(double[] A, double[] B, int sizeA, int sizeB, int step);


      int time;
      string fileName;

      List<double> signal1  = new List<double>();
      List<double> signal2  = new List<double>();
      List<double> signal3  = new List<double>();
      List<double> signal4  = new List<double>();
      List<double> signal5  = new List<double>();
      List<double> signal6  = new List<double>();
      List<double> signal7  = new List<double>();
      List<double> signal8  = new List<double>();
      List<double> signal9  = new List<double>();
      List<double> signal10 = new List<double>();
      List<double> signal11 = new List<double>();
      List<double> signal12 = new List<double>();
      List<double> signal13 = new List<double>();
      List<double> signal14 = new List<double>();

      List<double> realSignal1  = new List<double>();
      List<double> realSignal2  = new List<double>();
      List<double> realSignal3  = new List<double>();
```

```
List<double> realSignal4  = new List<double>();
List<double> realSignal5  = new List<double>();
List<double> realSignal6  = new List<double>();
List<double> realSignal7  = new List<double>();
List<double> realSignal8  = new List<double>();
List<double> realSignal9  = new List<double>();
List<double> realSignal10 = new List<double>();
List<double> realSignal11 = new List<double>();
List<double> realSignal12 = new List<double>();
List<double> realSignal13 = new List<double>();
List<double> realSignal14 = new List<double>();

List<Button> buttons        = new List<Button>();
List<double> corellationList = new List<double>();

bool fSavePattern = false, fStartCompare = false, fWritePattern = false, fCorellation = false;

/**
 * Function that initialize components
 */
public PatternsWindow()
{
   InitializeComponent();

   buttons.Add(button2);
   buttons.Add(button7);
   buttons.Add(button5);
   buttons.Add(button11);
   buttons.Add(button9);
   buttons.Add(button13);
   buttons.Add(button15);
   buttons.Add(button16);
   buttons.Add(button14);
   buttons.Add(button10);
   buttons.Add(button12);
   buttons.Add(button6);
   buttons.Add(button8);
   buttons.Add(button4);

   label2.Visible = false;
   LoadFoto();
}

/**
 * @function Compare
 * Function that compare patterns
 * @param signals data pockets from 14 sensors
 * @param l length of the pockets
 */
public void Compare(double[,] signals, int l)
{
   int timeStop = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second);
   if (time == timeStop)
   {
      fSavePattern = false;
   }

   // 1. Start pattern saving
   if (fCorellation == true)
   {
      for (int i = 0; i < l; i++)
      {
         signal1.Add(signals[0, i]);
         signal2.Add(signals[1, i]);
         signal3.Add(signals[2, i]);
         signal4.Add(signals[3, i]);
```

```
            signal5.Add(signals[4, i]);
            signal6.Add(signals[5, i]);
            signal7.Add(signals[6, i]);
            signal8.Add(signals[7, i]);
            signal9.Add(signals[8, i]);
            signal10.Add(signals[9, i]);
            signal11.Add(signals[10, i]);
            signal12.Add(signals[11, i]);
            signal13.Add(signals[12, i]);
            signal14.Add(signals[13, i]);
        }
    }
    // 2. Start Corellation comparing
    if (fStartCompare == true)
    {
        GetRealSignals(signals, l);
        Thread.Sleep(100);
        if (fCorellation == true)
        {
            Corellation(l);
        }
    }
}

/**
 * @function GetRealSignals
 * Function that gets signals in real time
 * @param signals data pockets from 14 sensors
 * @param l length of the pockets
 */
public void GetRealSignals(double[,] signals,int l)
{
    realSignal1.Clear();
    realSignal2.Clear();
    realSignal3.Clear();
    realSignal4.Clear();
    realSignal5.Clear();
    realSignal6.Clear();
    realSignal7.Clear();
    realSignal8.Clear();
    realSignal9.Clear();
    realSignal10.Clear();
    realSignal11.Clear();
    realSignal12.Clear();
    realSignal13.Clear();
    realSignal14.Clear();

    for (int i = 0; i < l; i++)
    {
        realSignal1.Add(signals[0, i]);
        realSignal2.Add(signals[1, i]);
        realSignal3.Add(signals[2, i]);
        realSignal4.Add(signals[3, i]);
        realSignal5.Add(signals[4, i]);
        realSignal6.Add(signals[5, i]);
        realSignal7.Add(signals[6, i]);
        realSignal8.Add(signals[7, i]);
        realSignal9.Add(signals[8, i]);
        realSignal10.Add(signals[9, i]);
        realSignal11.Add(signals[10, i]);
        realSignal12.Add(signals[11, i]);
        realSignal13.Add(signals[12, i]);
        realSignal14.Add(signals[13, i]);
    }
}
/**
```

```
 * @function Corellation
 * Function that makes Corellation comparing and indicates it
 * @param l length of the pockets
 */
public void Corellation(int l)
{
   // 1. Calculated corellation
   double cor1 = CorellationSize(signal1.ToArray(), realSignal1.ToArray(), signal1.Count(), l, 0.11);
   double cor2 = CorellationSize(signal2.ToArray(), realSignal2.ToArray(), signal2.Count(), l, 0.11);
   double cor3 = CorellationSize(signal3.ToArray(), realSignal3.ToArray(), signal3.Count(), l, 0.11);
   double cor4 = CorellationSize(signal4.ToArray(), realSignal4.ToArray(), signal4.Count(), l, 0.11);
   double cor5 = CorellationSize(signal5.ToArray(), realSignal5.ToArray(), signal5.Count(), l, 0.11);
   double cor6 = CorellationSize(signal6.ToArray(), realSignal6.ToArray(), signal6.Count(), l, 0.11);
   double cor7 = CorellationSize(signal7.ToArray(), realSignal7.ToArray(), signal7.Count(), l, 0.11);
   double cor8 = CorellationSize(signal8.ToArray(), realSignal8.ToArray(), signal8.Count(), l, 0.11);
   double cor9 = CorellationSize(signal9.ToArray(), realSignal9.ToArray(), signal9.Count(), l, 0.11);
   double cor10 = CorellationSize(signal10.ToArray(), realSignal10.ToArray(), signal10.Count(), l, 0.11);
   double cor11 = CorellationSize(signal11.ToArray(), realSignal11.ToArray(), signal11.Count(), l, 0.11);
   double cor12 = CorellationSize(signal12.ToArray(), realSignal12.ToArray(), signal12.Count(), l, 0.11);
   double cor13 = CorellationSize(signal13.ToArray(), realSignal13.ToArray(), signal13.Count(), l, 0.11);
   double cor14 = CorellationSize(signal14.ToArray(), realSignal14.ToArray(), signal14.Count(), l, 0.11);

   corellationList.Add(cor1);
   corellationList.Add(cor2);
   corellationList.Add(cor3);
   corellationList.Add(cor4);
   corellationList.Add(cor5);
   corellationList.Add(cor6);
   corellationList.Add(cor7);
   corellationList.Add(cor8);
   corellationList.Add(cor9);
   corellationList.Add(cor10);
   corellationList.Add(cor11);
   corellationList.Add(cor12);
   corellationList.Add(cor13);
   corellationList.Add(cor14);

   // 2. Indicates result
   for (int i = 0; i < 14; i++)
   {
      if (corellationList[i] < 0.5)
      {
         buttons[i].BackColor = Color.White;
      }
      if (corellationList[i] > 0.5 && corellationList[i] < 0.7)
      {
         buttons[i].BackColor = Color.LightCoral;
      }
      if (corellationList[i] > 0.7 && corellationList[i] < 1.0)
      {
         buttons[i].BackColor = Color.Red;
      }
   }
}

/**
 * @function LoadFoto
 * Function that load foto to the picturebox
 */
public void LoadFoto()
{
   pictureBox1.Load("C:\\Users\\intern\\Desktop\\EEG System\\EEG System\\bin\\Release\\Brain.png");
}


/**
```

```
        * Must be called when button clicked
        * Start saving pattern
        */
       private void SaveButton_Click(object sender, EventArgs e)
       {
          time = (((DateTime.Now.Hour * 60) + DateTime.Now.Minute * 60) + DateTime.Now.Second) + 10;

          fSavePattern = true;
          if (fWritePattern == true)
          {
             CreatePattern(fileName, 10000);
          }
          Thread.Sleep(100);
       }

       /**
        * Must be called when button clicked
        * Start comparing pattern
        */
       private void StartButton_Click(object sender, EventArgs e)
       {
          fStartCompare = true;
          label2.Visible = true;
          Thread.Sleep(100);
       }

       /**
        * Must be called when button clicked
        * Stop comparing
        */
       private void StopButton_Click(object sender, EventArgs e)
       {
          fStartCompare = false;
          label2.Visible = false;
       }

       /**
        * Must be called when checkbox selected
        * Sets file for recording
        */
       private void WriteCheckBox_CheckedChanged(object sender, EventArgs e)
       {
          if (WriteCheckBox.Checked == true)
          {
             Thread.Sleep(100);
             openFileDialog1.ShowDialog();
             fileName = openFileDialog1.FileName;

             fWritePattern = true;
          }
       }

       /**
        * Must be called when radioButton selected
        */
       private void CorellationRadioButton_CheckedChanged(object sender, EventArgs e)
       {
          if (CorellationRadioButton.Focused == true)
          {
             fCorellation = true;
          }
       }
    }
}
```

Лістинг Engine.cs

```
/**
* @file Engine.cs
* @brief Module of getting data
* @author Viktor Evsikov
* @date Created August 18, 2013 14:15:00
**/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using System.Runtime.InteropServices;
using System.IO;
using System.Windows.Forms;

/**
 * Engine is a main module of application. It implement functions to reading data and draw graphs.
 * Includes functions to reading data from Emotive EPOC and from file, functions to send data to other classes,
 * functions to send information about drawing, functions to open windows.
 */

namespace WindowsFormsApplication1
{
   public class Engine
   {
      [DllImport("DataLayer.dll", EntryPoint = "Init", CharSet = CharSet.Ansi)]
      static extern void Init(string filename);

      [DllImport("DataLayer.dll", EntryPoint = "Update", CharSet = CharSet.Unicode)]
      new static extern bool Update();

      [DllImport("DataLayer.dll", EntryPoint = "GetContactData", CharSet = CharSet.Unicode)]
      static extern double GetContactData(int numberOfSamples, int sizeData);

      [DllImport("DataLayer.dll", EntryPoint = "GetNumberOfSamples", CharSet = CharSet.Unicode)]
      static extern int GetNumberOfSamples();

      [DllImport("DataLayer.dll", EntryPoint = "GetSizeData", CharSet = CharSet.Unicode)]
      static extern int GetSizeData();

      [DllImport("DataLayer.dll", EntryPoint = "WriteToFile", CharSet = CharSet.Ansi)]
      static extern void WriteToFile(string ch);

      [DllImport("DataLayer.dll", EntryPoint = "CheckConnection", CharSet = CharSet.Unicode)]
      static extern bool CheckConnection();

      [DllImport("DataLayer.dll", EntryPoint = "GetQuality", CharSet = CharSet.Unicode)]
      static extern double GetQuality(int i);

      [DllImport("DataLayer.dll", EntryPoint = "GetSizeQuality", CharSet = CharSet.Unicode)]
      static extern int GetSizeQuality();


      [DllImport("DataLayer.dll", EntryPoint = "GetGyro", CharSet = CharSet.Unicode)]
      static extern double GetGyro(int i);


      [DllImport("DataLayer.dll", EntryPoint = "GetSizeEmotion", CharSet = CharSet.Unicode)]
      static extern int GetSizeEmotion();
```

```
[DllImport("DataLayer.dll", EntryPoint = "GetEmotion", CharSet = CharSet.Unicode)]
static extern double GetEmotion(int i);


SensorsWindow sensorsWindow;
GyroWindow gyroWindow;
EmotionWindow emotionWindow;
PatternsWindow patternsWindow;

double[,] dataFromFile;

double[,] signals;
double[,] emotion;
double[,] quality;

double gyroX;
double gyroY;


public bool readFile = false, readES = false;

public Thread drawing;
public Thread geting;

int nData, nQuality, nEmotion, nSamples, fileLines;
public bool sensorsOpen = false, gyroOpen = false, emotionOpen = false, patternsOpen = false;

/**
 * @function StartRead
 * Function that start reading data from Emotion EPOC
 */
public void StartRead()
{
    Init(" ");
    GetConst();
    geting = new Thread(Refresh);
    geting.Start();
}

/**
* @function Refresh
* Function that refresh data pockets
*/
public void Refresh()
{
    while (true)
    {
        Thread.Sleep(300);
        GetData();
    }
}



/**
 * @function GetConst
 * Function that gets size of pockets
 */
public void GetConst()
{
    nData = GetSizeData();
    nQuality = GetSizeQuality();
    nEmotion = GetSizeEmotion();
}

/**
```

```
 * @function GetData
 * Function that gets data
 */
public void GetData()
{
   Update();
   WriteToFile(" ");

   nSamples = GetNumberOfSamples();

   signals = new double[nData, nSamples];
   quality = new double[nQuality, nSamples];
   emotion = new double[nEmotion, nSamples];

   if (nSamples != 0)
   {
      for (int i = 0; i < nSamples; i++)
      {
         for (int j = 0; j < nData; j++)
         {
            signals[j, i] = GetContactData(j, i);
         }
         for (int k = 0; k < nQuality; k++)
         {
            quality[k, i] = GetQuality(k);
         }
         for (int q = 0; q < nEmotion; q++)
         {
            emotion[q, i] = GetEmotion(q);
         }
      }
      gyroX = GetGyro(0);
      gyroY = GetGyro(1);
   }
}

/**
 * @function GetDataFromFile
 * Function that gets data from file
 * @param filePath path of the file
 * @return dataFromFile array of data
 */
public double[,] GetDataFromFile(string filePath)
{
   string str = "";
   char[] ch;
   List<double> line = new List<double>();
   int count = File.ReadAllLines(filePath).Count();
   string[] data = new string[count];

   for (int i = 0; i < count; i++)
   {
      data = File.ReadLines(filePath).ToArray();
   }

   for (int i = 0; i < count; i++)
   {
      str = "";
      ch = data[i].ToCharArray();
      for (int j = 0; j < ch.Length; j++)
      {
         if (ch[j].ToString() != "\t" && ch[j].ToString() != " ")
         {
            str += ch[j].ToString();
         }
         else
```

```
            {
                if (str != "")
                {
                    line.Add(Convert.ToDouble(str));
                }
                str = "";
            }
        }
    }

    fileLines = Convert.ToInt32(line[0]);
    line.RemoveAt(0);
    dataFromFile = new double[fileLines, 14];

    for (int i = 0; i < fileLines; i++)
    {
        for (int j = 0; j < 14; j++)
        {
            if (line.Count != 0)
            {
                dataFromFile[i, j] = line[0];
                line.RemoveAt(0);
            }
        }
    }

    signals = dataFromFile;
    nSamples = fileLines;
    signals = dataFromFile;
    return dataFromFile;
}

/**
 * @function StartDraw
 * Function that start drawing graphs
 */
public void StartDraw()
{
    drawing = new Thread(Draw);
    drawing.Start();
}

/**
 * @function Draw
 * Function that send to windows information about redrawing
 */
public void Draw()
{
    while (true)
    {
        Thread.Sleep(300);
        if (nSamples != 0)
        {
            if (sensorsOpen == true)
            {
                if (Application.OpenForms["SensorsWindow"] != null)
                {
                    sensorsWindow.RedrawSensors(signals, quality, nSamples);
                }
            }
            if (gyroOpen == true)
            {
                if (Application.OpenForms["GyroWindow"] != null)
                {
                    gyroWindow.RedrawGyro(gyroX, gyroY, nSamples);
                }
```

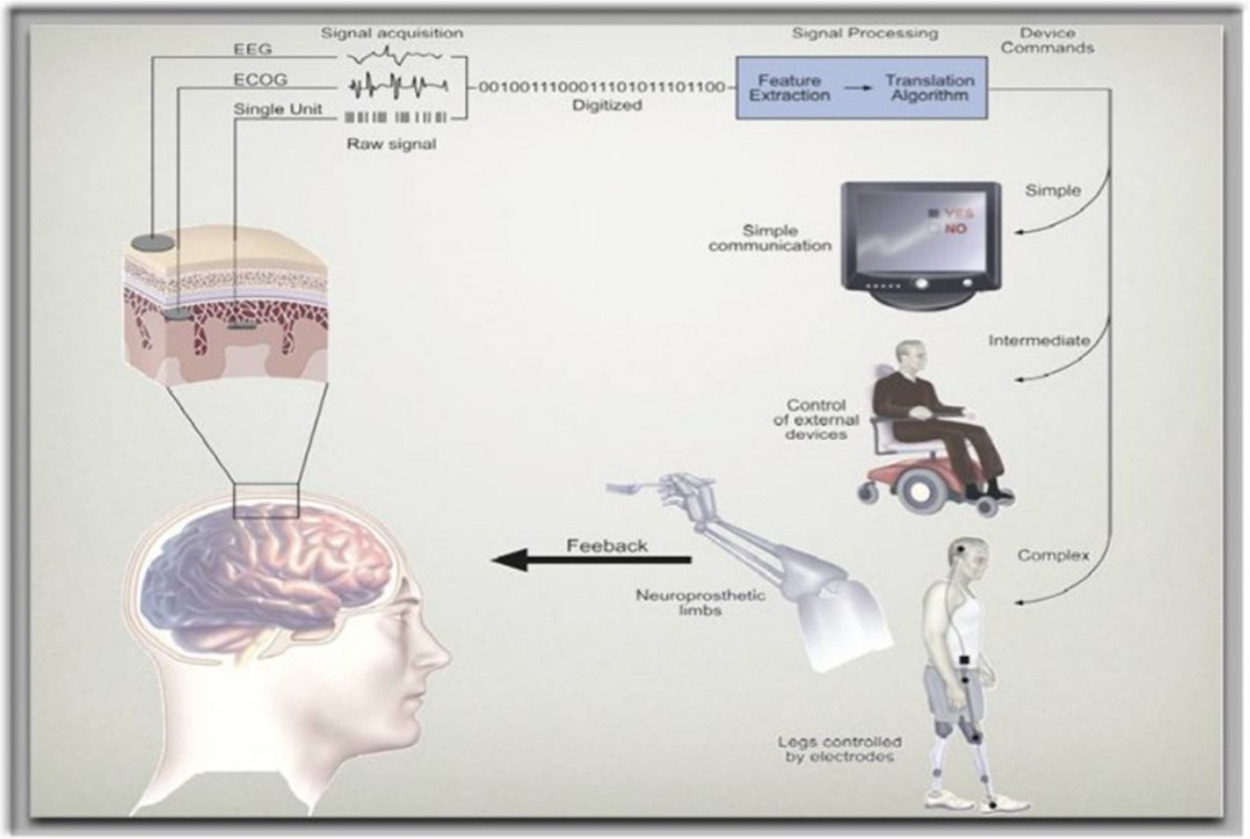
```
                }
                if (emotionOpen == true)
                {
                    if (Application.OpenForms["EmotionWindow"] != null)
                    {
                        emotionWindow.RedrawEmotions(emotion);
                    }
                }
                if (patternsOpen == true)
                {
                    if (Application.OpenForms["PatternsWindow"] != null)
                    {
                        patternsWindow.Compare(signals, nSamples);
                    }
                }
            }
        }
    }

    /**
     * @function FormOpen
     * Function that open windows and gives a data source
     */
    public void FormOpen(bool f1,bool f3,bool f4,bool f5)
    {
        if (f1 == true)
        {
            sensorsWindow = new SensorsWindow();
            if (this.readFile == true)
            {
                sensorsWindow.fReadFromF = true;
            }
            sensorsWindow.Show();
            sensorsOpen = true;
        }
        if (f3 == true)
        {
            gyroWindow = new GyroWindow();
            gyroWindow.Show();
            gyroOpen = true;
        }
        if (f4 == true)
        {
            emotionWindow = new EmotionWindow();
            emotionWindow.Show();
            emotionOpen = true;
        }
        if (f5 == true)
        {
            patternsWindow = new PatternsWindow();
            patternsWindow.Show();
            patternsOpen = true;
        }
    }
  }
}
```

ДОДАТОК Б
Графічна частина

Постановка задачи

Тема	«Разработка подсистемы для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC»
Руководитель	к.т.н., доцент Ломоносов Ю.В.
Исполнитель	студент 560м гр. Евсиков В.А.
Объект исследования	нейрокомпьютерный интерфейс
Предмет исследования	методы работы интерфейса «мозг-компьютер»
Цель	создание приложения обеспечивающего анализ и проверку корректности работы нейрокомпьютерного интерфейса
Задачи	- анализ работы нейрокомпьютерного интерфейса - разработка алгоритма считывания данных с устройства - разработка GUI - разработка базы данных для хранения результатов проверок
Методы исследования	- методы работы с сигналами, методы разработки приложений - методы разработки БД

Общая схема работы НКИ

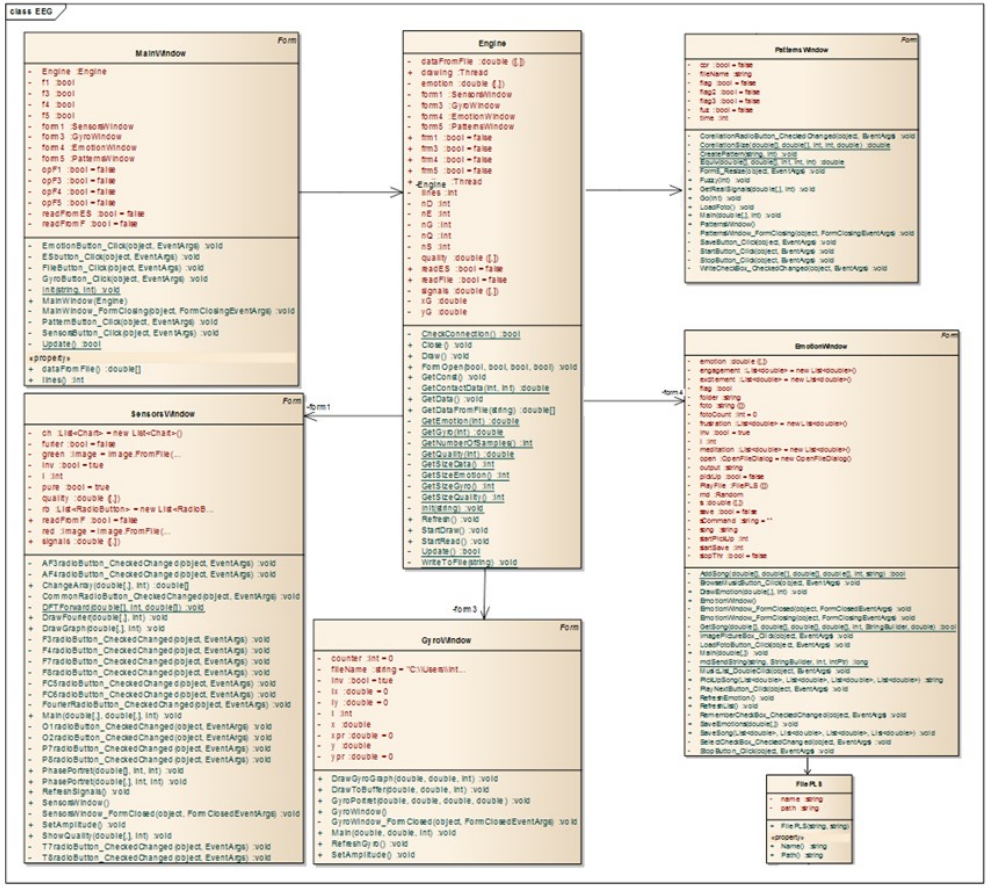


Изм.	Лист	№ Докум.	Подп.	Дата	ХАІ5.502.534.003 СБ			
Разраб.		Евсиков В.А.			Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит.	Масса	Масштаб
Проверил		Ломоносов Ю.В.				У		
						Лист 2	Листов 9	
Н. контр.						НАУ „ХАІ”		
Утверд.								

Нейрокомпьютерный интерфейс Emotiv EPOC

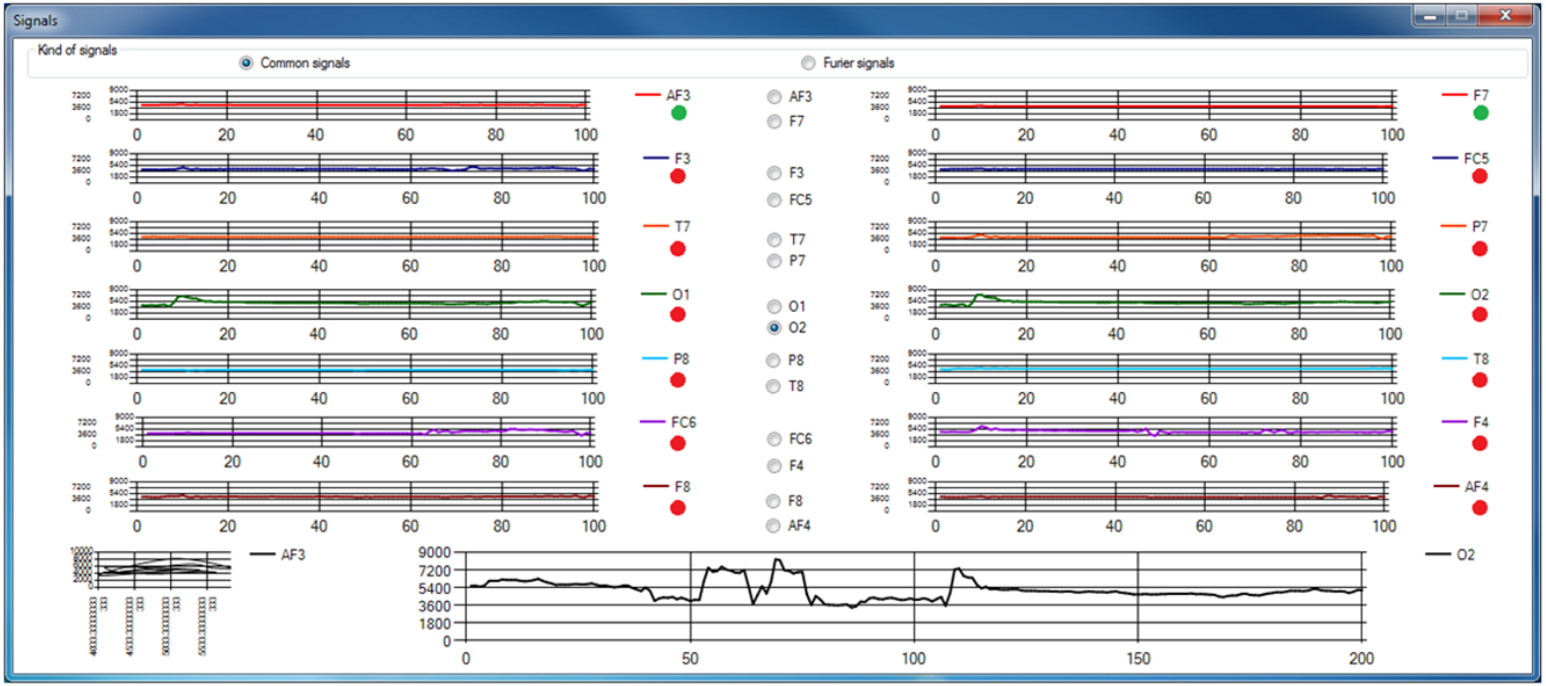
					ХАІ5.502.534.003 СБ		
Изм	Лист	№ Докум.	Подп.	Дата	Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит.	Масса / Масштаб
Разраб.		Евсиков В.А.				У	
Проверил		Ломоносов Ю.В.				Лист 3	Листов 9
Н. контр.						НАУ „ХАІ”	
Утверд.							

Диаграмма классов



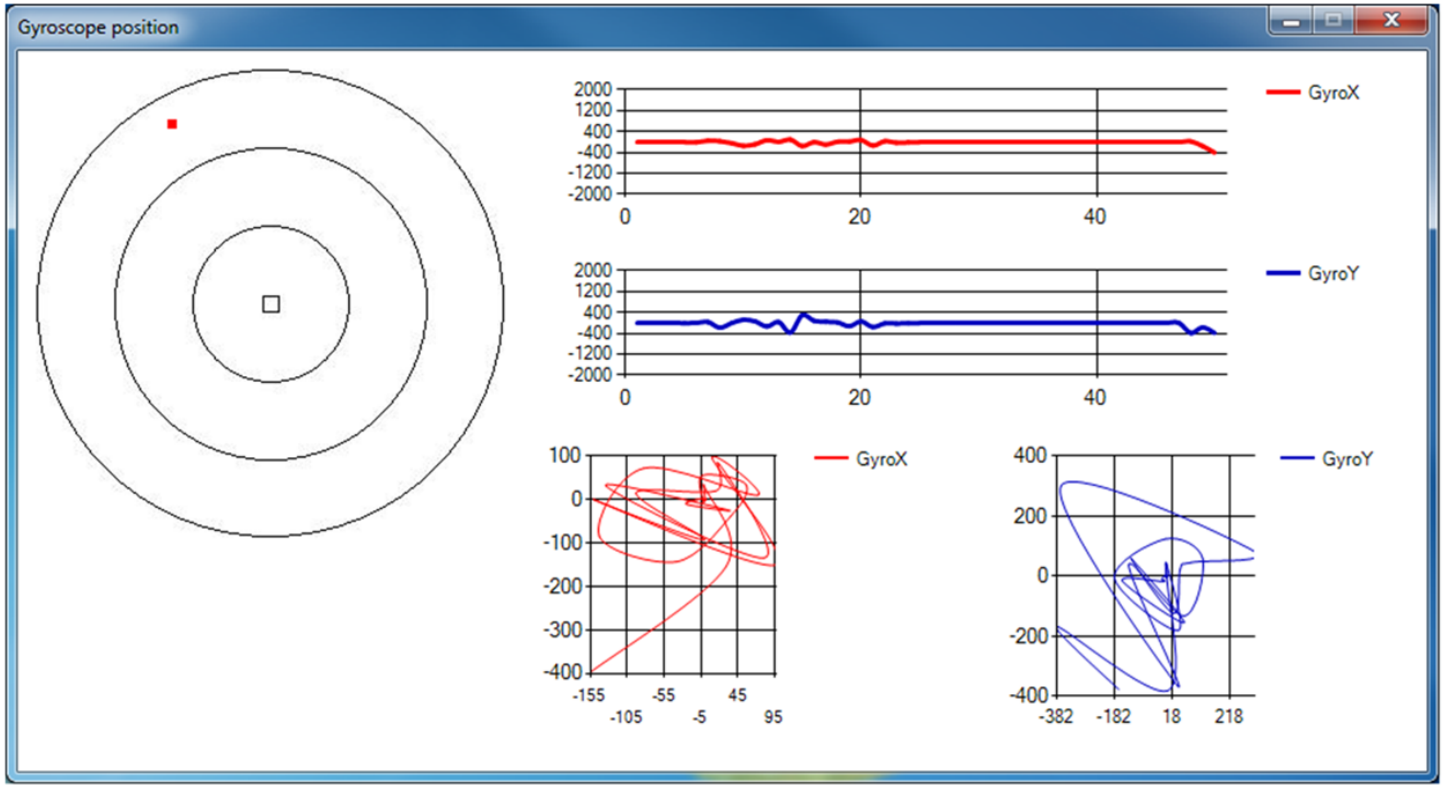
					ХАІ5.502.534.003 СБ	
Изм.	Лист	№ Докум.	Подп.	Дата	Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит. У / Масса / Масштаб
Разраб.		Евсиков В.А.				Лист 4 / Листов 9
Проверил		Ломоносов Ю.В.				НАУ „ХАІ”
Н. контр.						
Утверд.						

Анализ сигналов датчиков



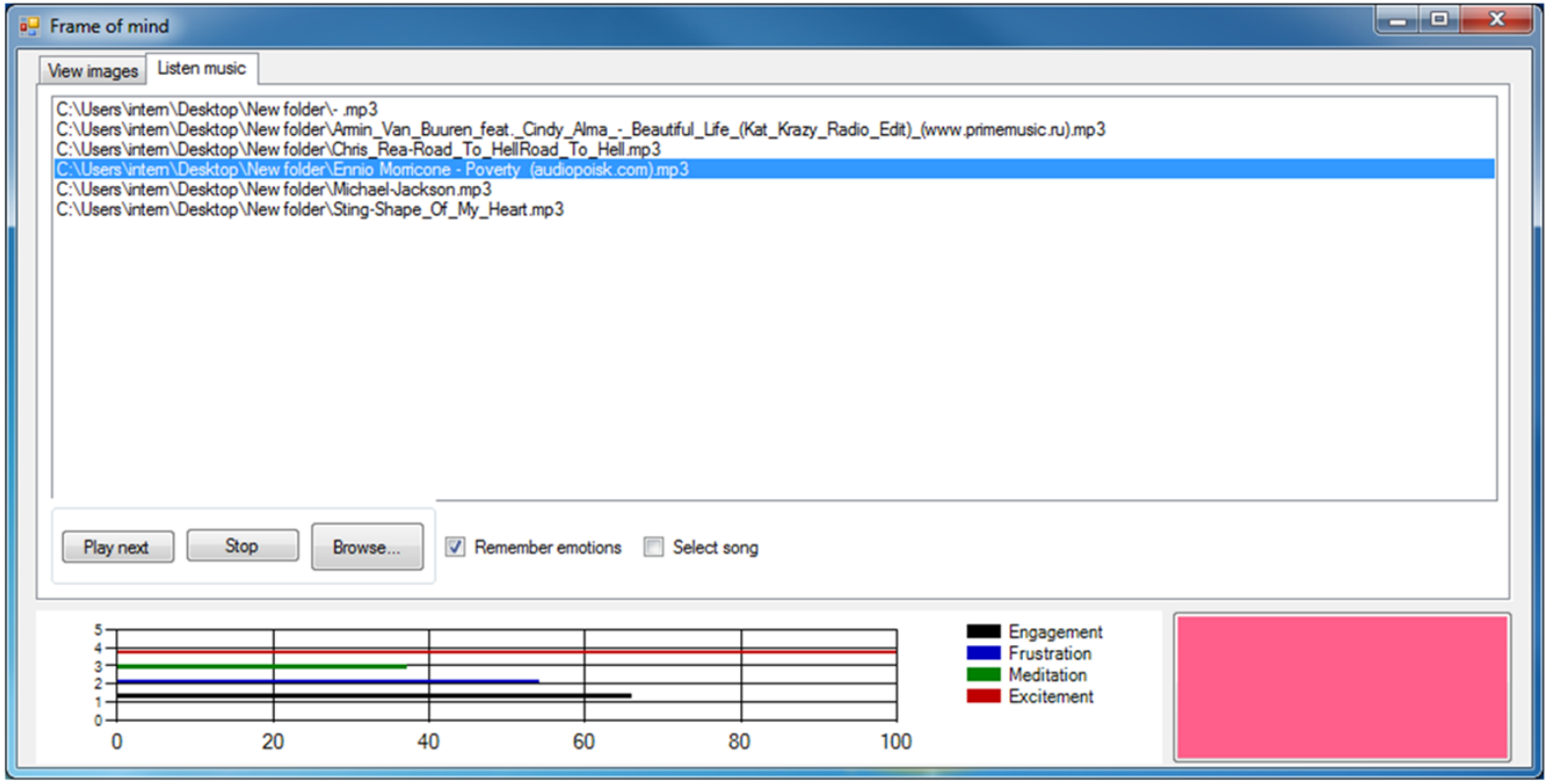
					ХАІ5.502.534.003 СБ	
Изм.	Лист	№ Докум.	Подп.	Дата	Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит. / Масса / Масштаб: У
Разраб.		Евсиков В.А.				Лист 5 / Листов 9
Проверил		Ломоносов Ю.В.				НАУ „ХАІ”
Н. контр.						
Утверд.						

Анализ сигналов гироскопа



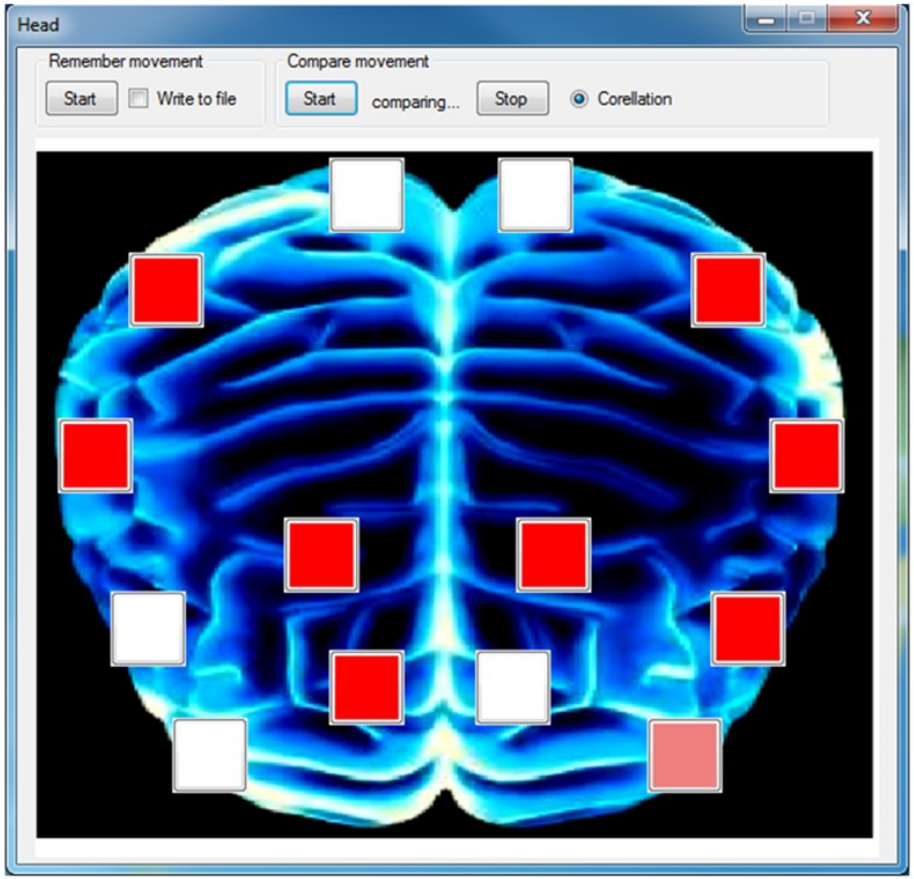
					ХАІ5.502.534.003 СБ	
Изм.	Лист	№ Докум.	Подп.	Дата	Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит. / Масса / Масштаб: У
Разраб.		Евсиков В.А.				Лист 6 / Листов 9
Проверил		Ломоносов Ю.В.				НАУ „ХАІ”
Н. контр.						
Утверд.						

Вывод эмоционального состояния



					ХАІ5.502.534.003 СБ	
Изм.	Лист	№ Докум.	Подп.	Дата	Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит. / Масса / Масштаб: У
Разраб.		Евсиков В.А.				Лист 7 / Листов 9
Проверил		Ломоносов Ю.В.				НАУ „ХАІ”
Н. контр.						
Утверд.						

Работа с паттернами



					ХАІ5.502.534.003 СБ	
Изм.	Лист	№ Докум.	Подп.	Дата	Подсистема для анализа и использования нейрокомпьютерного интерфейса Emotiv EPOC	Лит. / Масса / Масштаб: У
Разраб.		Евсиков В.А.				
Проверил		Ломоносов Ю.В.				Лист 8 / Листов 9
Н. контр.						НАУ „ХАІ”
Утверд.						