Санкт-Петербургский политехнический университет Петра Великого

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий



**ДИССЕРТАЦИЯ**

**на соискание ученой степени**

**МАГИСТРА**

Тема: **Разработка системы определения психоэмоционального состояния пользователя на основе данных ЭЭГ   
с использованием биологической обратной связи**

Направление: 09.04.01 – Информатика и вычислительная техника   
Магистерская программа: 09.04.01\_14 – Проектирование аппаратно-программных средств вычислительной техники

Выполнил студент гр. 63501/2

Перминов А.А.

Санкт-Петербург

2017

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Диссертация допущена к защите  
 зав. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.М. Ицыксон

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

**ДИССЕРТАЦИЯ  
на соискание ученой степени   
МАГИСТРА**

Тема: **Разработка системы определения психоэмоционального состояния пользователя на основе данных ЭЭГ   
с использованием биологической обратной связи**

Направление: 09.04.01 – Информатика и вычислительная техника   
Магистерская программа: 09.04.01\_14 – Проектирование аппаратно-программных средств вычислительной техники

|  |  |
| --- | --- |
| Выполнил студент гр. 63501/2 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Перминов А.А. |
| Научный руководитель,  доц. к.т.н. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Болсуновская М.В. |
| Консультант по нормоконтролю,  ст. преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Нестеров |

Санкт-Петербург

2017

**РЕФЕРАТ**

**Отчет, 120 стр., 40 рис., 13 табл., 13 ист., 5 прил.**

**РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, БАЗА ДАННЫХ, ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA, ОПЕРАЦИОННАЯ СИСТЕМА ANDROID, ФРАКТАЛЬНАЯ РАЗМЕРНОСТЬ, БИОЛОГИЧЕСКАЯ ОБРАТНАЯ СВЯЗЬ, ПСИХОЭМОЦИОНАЛЬНОЕ СОСТОЯНИЕ**

Цель работы – разработка программного обеспечения для системы определения психоэмоционального состояния по данным ЭЭГ с использованием биологической обратной связи на базе операционной системы Android.

В ходе работы проведен обзор существующих моделей эмоционального состояния и алгоритмов по определению эмоционального состояния по данным ЭЭГ, обзор существующих решений. Выполнено проектирование программного обеспечения системы определения психоэмоционального состояния с выделением функциональных компонентов, разработана структура базы данных для хранения информации о проведенных сеансах использования.

Разработано спроектированное программное обеспечение для ОС Android с использованием средств разработки Android SDK и языка программирования Java. Реализована структура базы данных с помощью библиотеки ORMLite.

Разработанное программное обеспечение предлагается к эксплуатации в стоматологической клинике при проведении медицинских процедур.

**ABSTRACT**Report, 120 pages, 40 figures, 13 tables, 13 references, 5 appendices  
 **SOFTWARE DEVELOPMENT, DATABASE, JAVA PROGRAMMING LANGUAGE, ANDROID OPERATION SYSTEM, FRACTAL DIMENSION, BIOLOGICAL FEEDBACK, EMOTICAL STATE**

The goal of the work is software development for the emotional state recognition system based on the Android operating system using EEG data and biofeedback.

In the course of the work, a review of existing models of the emotional state, algorithms for emotional state recognition according to EEG data, a review of existing solutions was made. The design of the software for emotional state recognition was made with the allocation of functional components. A database structure was developed to store information about the use sessions.

Software was developed for Android OS using such development tools as Android SDK and Java programming language. Structure of the designed database was implemented using library ORMLite.

The developed software is supposed to be used in a dental clinic for medical procedures.

Санкт-Петербургский Государственный Политехнический Университет

Институт информационных технологий и управления

Кафедра компьютерные системы и программные технологии

УТВЕРЖДАЮ

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 201 г.

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**ЗАДАНИЕ**

**НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

студенту Перминову Александру Александровичу

1. Тема проекта (работы): Разработка системы определения психоэмоционального состояния пользователя на основе данных ЭЭГ с использованием биологической обратной связи

2. Срок сдачи студентом законченного проекта (работы): 30 мая 2017 года

3. Исходные данные к проекту (работе):

- Целевая платформа: Android;

- Язык программирования: Java;

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):

1. Введение.

2. Анализ возможных решений определения психоэмоционального состояния по данным ЭЭГ, постановка цели и задач.

3. Проектирования программного обеспечения для системы определения психоэмоционального состояния.

4. Разработка спроектированного программного обеспечения.

5. Заключение и перспективы развития.

5. Консультанты по проекту (с указанием относящихся к ним разделов проекта, работы):

Нестеров С.А. – консультант по нормоконтролю

6. Дата выдачи задания 5 февраля 2017 года

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Болсуновская М.В.)

Задание принял к исполнению «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2017 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись студента)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 11](#_Toc483927409)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 13](#_Toc483927410)

[1.1. Модели эмоций 13](#_Toc483927411)

[1.2. Алгоритмы определения эмоций 19](#_Toc483927412)

[1.3. Фрактальная размерность 24](#_Toc483927413)

[1.4. Существующие решения 27](#_Toc483927414)

[1.5. Биологическая обратная связь 30](#_Toc483927415)

[1.6. Цель и задачи 31](#_Toc483927416)

[2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ ОПРЕДЕЛЕНИЯ ПСИХОЭМОЦИОНАЛЬНОГО СОСТОЯНИЯ 32](#_Toc483927417)

[2.1. Требования 32](#_Toc483927418)

[2.2. Компоненты 36](#_Toc483927419)

[2.3. Средства разработки 38](#_Toc483927420)

[2.4. Взаимодействие 39](#_Toc483927421)

[2.5. Обработчик данных ЭЭГ 40](#_Toc483927422)

[2.6. Работа с администрирующим устройством 42](#_Toc483927423)

[2.7. База данных 44](#_Toc483927424)

[2.7. Модуль определения состояния 50](#_Toc483927425)

[3. РАЗРАБОТКА ПРОГРАММНОЙ ЧАСТИ СИСТЕМЫ 54](#_Toc483927426)

[3.1. Система классов 54](#_Toc483927427)

[3.2. Обработка данных ЭЭГ 55](#_Toc483927428)

[3.3. Работа с администрирующим устройством 58](#_Toc483927429)

[3.4. Работа с базой данных 60](#_Toc483927430)

[3.5. Алгоритм определения состояния 66](#_Toc483927431)

[4. ТЕСТИРОВАНИЕ 70](#_Toc483927432)

[4.1 Тестирование реализованного алгоритма работы 70](#_Toc483927433)

[4.2 Тестирование взаимодействия основной и администрирующей программы 74](#_Toc483927434)

[ЗАКЛЮЧЕНИЕ 80](#_Toc483927435)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 82](#_Toc483927436)

[ПРИЛОЖЕНИЕ 1. Документация ПО. Спецификация 84](#_Toc483927437)

[ПРИЛОЖЕНИЕ 2. Документация ПО. Техническое задание 86](#_Toc483927438)

[ПРИЛОЖЕНИЕ 3. Документация ПО. Описание программы. 96](#_Toc483927439)

[ПРИЛОЖЕНИЕ 4. Документация ПО. Программа и методика испытаний 108](#_Toc483927440)

[ПРИЛОЖЕНИЕ 5. Документация ПО. Текст программы 119](#_Toc483927441)

# ВВЕДЕНИЕ

Выявление и распознавание информации об эмоциональном состоянии это важная тема в области аффективных вычислений, то есть при изучении человеческих эмоций с помощью технических систем и устройств. Изменения в эмоциональном состоянии часто проявляются в лицевых, речевых и жестовых проявлениях эмоций.

Так как изменение настроения сильно влияет на обычный эмоциональный процесс, распознавание эмоций это так же одна из приоритетных целей в области психопатологических расстройств. В последнее десятилетие были испробованы несколько подходов к получению надежной методики автоматического распознавания эмоционального состояния и настроения, начиная с лицевых выражений, поведенческих шаблонов и физиологических сигналов. Несмотря на это, в настоящий момент все еще практикуются простейшие эмоциональные опросники или интервью для эмоциональной оценки. В медицине, например, диагностика патологий, связанных с эмоциональными изменениями, производится в основном через опыт врача.

Несколько вычислительных методов для распознавания эмоций базируются на данных с центральной нервной системы, к примеру, электроэнцефалограмме (ЭЭГ). Такие методы оправданы тем, что эмоции человека возникают в кортикальном слое, затем распространяясь на несколько зон для регуляции и ощущений. Префронтальная кора и миндалины, по сути, представляют собой два основных пути: аффективные проявления позволяют префронтальной коре распознать информацию о стимуле и передать ее в другие зоны центральной автономной системы, в ствол мозга, таким образом формируя подходящий по контексту ответ. Кратко представленные стимулы получают быстрый доступ к распознаванию эмоций через миндалину. Кроме того, было установлено, что зрительная кора вовлечена в эмоциональные реакции на разные стимулы. Нарушение этого процесса в центральной нервной системе ведет к патологиям, таким как ангедония, то есть потеря удовольствия или интереса к ранее желанному стимулу, что является ключевой особенностью депрессии и других серьезных эмоциональных расстройств.

Кроме серьезности проблемы необходимо также отметить относительно широкую распространенность проблемы. Несколько эпидемиологических исследований сообщают о том, что 2 миллионам американцев диагностировали биполярное расстройство, а около 82,7 миллионам взрослых жителей Европы в возрасте от 18 до 65 диагностировано хотя бы одно психическое расстройство [1].

Таким образом, определение психоэмоционального состояния является актуальной проблемой. В рамках данной работы планируется разработать программное обеспечение для распознавания психоэмоционального состояния по данным ЭЭГ.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Модели эмоций

В попытках решить проблему определения эмоций, необходимо задаться вопросом, как формализовать эмоцию, какие в принципе бывают эмоции, и как их дифференцировать друг от друга.

Согласно теории «Discrete emotion theory», существует небольшое количество основных эмоций, своеобразный эмоциональный базис, который позволяет получить весь спектр существующих эмоций человека. Однако ученые спорят по поводу того, какие эмоции и в каком количестве можно считать базовыми.

Имея эмоциональный базис, можно определить пространство, в котором будет определено все множество эмоций. От эмоционального базиса зависит размерность этого пространства. Как было сказано ранее, можно определить эмоциональный базис по-разному. И, как следствие, существуют разные пространственные модели эмоций. Перечислим и кратко опишем несколько из них.

**Циркумплексная модель.** Разработана Джеймсом Расселом. Предлагает, что эмоции распределены в двухмерном пространстве, содержащем такие характеристики эмоций как валентность (тон) и интенсивность [2]. Интенсивность располагается по вертикальной оси, валентность по горизонтальной оси. В данной модели эмоциональные состояния могут быть представлены на любом уровне валентности и интенсивности. В основном данная модель используется для тестовых стимуляций эмоциональной окраски слов, лицевых выражений и аффектных состояний.

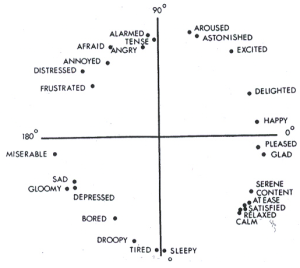


Рисунок 1.1. Циркумплексная модель эмоций

**Векторная модель.** Предполагает, что каждая эмоция представляет собой вектор, указывающий на две точки, и имеет форму бумеранга. К примеру, положительное значение валентности сдвинет эмоцию вверх, отрицательное вниз. В этой модели высокоинтенсивные эмоции отличаются по валентности, тогда как низкоинтенсивные - менее нейтральными, и представлены ближе к центру. Модель используется для тестирования эмоциональной окраски слов и изображений.

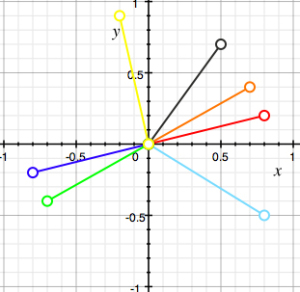


Рисунок 1.2. Векторная модель

**PANA model.** Модель положительная активация – отрицательная активация. Или согласованная модель эмоции, которая предполагает, что положительный аффект и отрицательный аффект – две отдельные системы [3]. Как и в векторной модели, высокоинтенсивные эмоции определяются своей валентности, тогда как низкоинтенсивные менее нейтральны в плане тональности. Вертикальная ось представляет значения положительного аффекта от низкого до высокого, а горизонтальная представляет значения отрицательного аффекта от низкого до высокого.

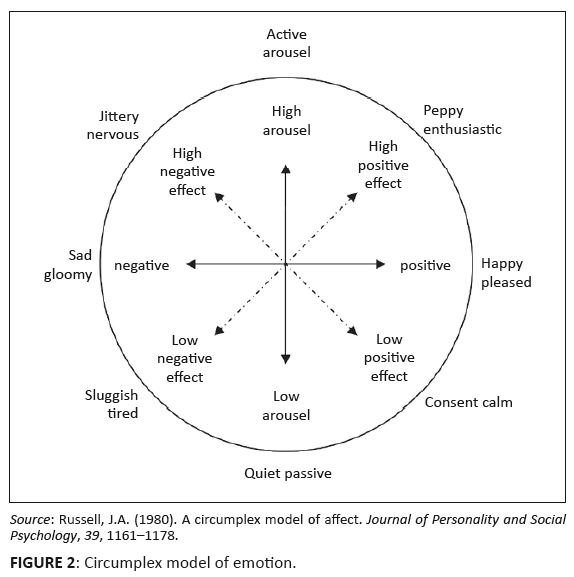


Рисунок 1.3. PANA model

**Модель Плутчика.** Предлагаемая трехмерная модель располагает эмоции концентрическими кругами, где внутренние круги представляют базовые эмоции, а внешние более сложные. При этом внешние круги сформированы смешением внутренних.

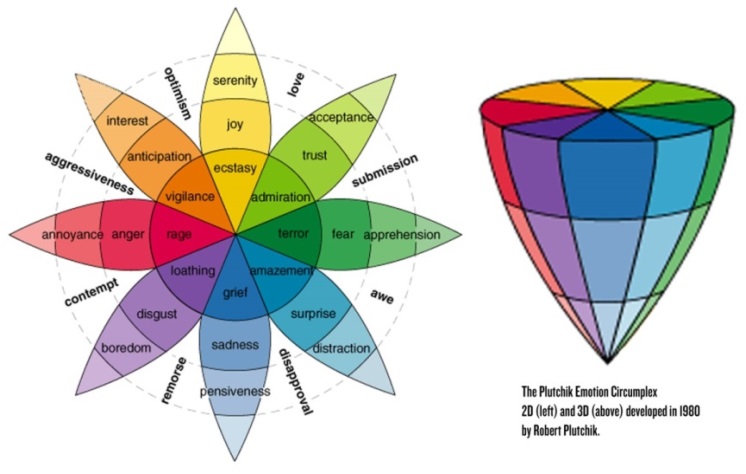


Рисунок 1.4. Модель Плутчика

**Куб эмоций Лёвхейма.** Лёвхейм предложил прямую зависимость между специфическими комбинациями уровней нейромедиаторов: допамина, норадреналина и серотонина и восемью базовыми эмоциями. Трехмерная модель представляет уровни нейромедиторов как стороны куба, а восемь основных эмоций распределены в углах этого куба.

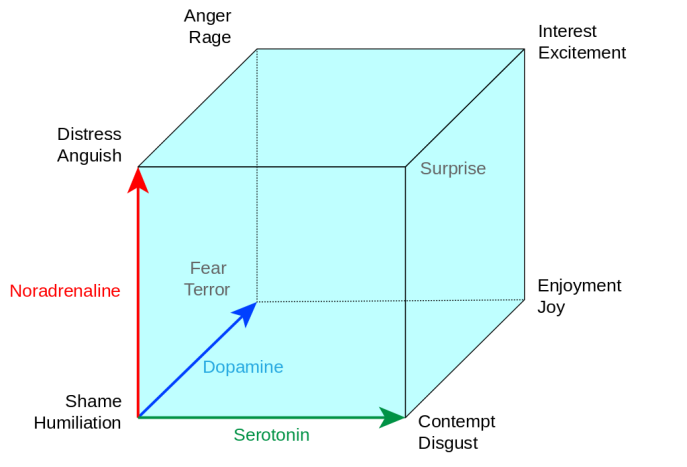


Рисунок 1.5. Куб Лёвхейма

Наиболее широко используемой на данный момент является указанная выше циркумплексная модель, в измерениях которой располагаются возбуждение и тональность. Возбуждение на шкале варьируется от отсутствия возбуждения до восторга, тональность – от негативной до позитивной. Данная модель является предпочтительной в распознавании эмоций из-за того, что можно расположить отдельные эмоции в пространстве, даже в случае, когда они не имеют конкретного названия для испытываемого ощущения.

Сравнивая циркумплексную модель с векторной моделью, можно выделить основное отличие: возможность расположения эмоций, характеризующихся высокой интенсивностью и нейтральной тональностью, такими как возбуждение, удивление, вовлечение или любые другие, которые предполагают сильную выраженность, но не могут быть помечены как негативные или позитивные [4]. Для определения таких эмоций требуется использование циркумплексной модели, потому как в векторной модели полагается разделение положительных и отрицательных высокоинтенсивных эмоций.

Таким образом, можно сказать, что наиболее подходящей моделью в данной работе будет циркумплексная модель, так как указанное выше преимущество в возможности размещения высокоинтенсивных состояний без яркой тональной оценки потребуется в реализации системы.

## Алгоритмы определения эмоций

Определив эмоции и существующие модели эмоций, необходимо сказать о том, как они выражаются с помощью тела человека. Реакции, вызываемые эмоциями, можно разделить на 3 группы: вегетативные реакции, мышечные реакции и импрессивно-экспрессивные реакции [5].

Эмоции характеризуются прежде всего нарушениями вегетативных функций. Последние являются, бесспорно, составной частью эмоций. Вегетативные проявления эмоций весьма разнообразны: изменение сопротивления кожи, частоты сердечных сокращений, кровяного давления, сужение и расширение сосудов, изменение скорости, амплитуды и ритма дыхания, температуры кожи, потоотделения, диаметра зрачка, секреции слюны; наблюдаются расстройства пищеварительной системы, сокращение и расслабление сфинктеров, меняются электрическая активность мозга, химический и гормональный состав крови, мочи, слюны, основной обмен. Однако нелишне подчеркнуть, что лишь некоторые изменения вегетативных функций могут рассматриваться как характерные проявления эмоций.

Исследования середины XX в. показали, что внешняя стимуляция (ощущение) трансформируется в тоническую, висцеральную и мышечную активность всего организма, представляющей аффективное жизни индивида. Эмоция начинается с гипертонуса, внешним проявлением которого является преобразование не использованной по назначению энергии в спазматические движения: смех, слезы, беспорядочные действия.

Из [5]: «Зафиксировано увеличение мышечного тонуса (электроды устанавливались во главе) при прослушивании детективной рассказы течение 10 мин. Одно только представление о движении вызывает увеличение электрической активности соответствующих групп мышц. За чрезмерной активацией наблюдается тем более мышечное напряжение, чем сильнее стремление субъекта подавить вызванные стимуляцией движения; это напряжение еще больше усиливается под влиянием социальных и моральных запретов (напряжение юноши, не решается обнять девушку которая нравится ему; напряжение ребенка, не желает подчиниться определенной требованию)».

Периферические изменения, охватывающие весь организм при эмоциях, распространяются и наружу. Захватывая систему мышц лица и всего тела, они оказываются в выразительных движениях - в мимике (выразительные движения лица), пантомимике (выпадающие движения всего тела) и в голосовых реакциях (интонации и тембре голоса). Эмоциональные переживания выражаются не только в сильных движениях, но и в микродвижениях (тремор, реакции зрачков).

Как вывод, эмоции проявляются множеством различных наблюдаемых реакций. Однако не все эти реакции удобно наблюдать в контексте вычислительной техники. А также выявление корреляции между детектируемыми реакциями является сложной задачей.

Кроме того, можно выделить основные способы определения эмоций: распознавание речи, распознавание лицевых выражений, распознавание жестов тела, физиологическое наблюдение. Область, которая занимается разработкой систем и устройств, предназначенных для распознавания, обработки и интерпретации эмоций называется «affective computing», что можно перевести как «аффективные вычисления».

Определение эмоций начинается с работы пассивных датчиков, которые захватывают данные о физическом состоянии пользователя или его поведении без интерпретации этих данных. К примеру, видеокамера может снимать лицевые выражения, жесты тела, тогда как микрофон захватывает речь. Другие датчики предоставляют данные, измеряя физиологические данные, такие как температура кожи, электрическая активность кожи и т.п.

Далее в полученных данных происходит распознавание некоторых паттернов с помощью техник машинного обучения, направленных на разные формы, такие как распознавание речи, обработка естественного языка или выявление лицевых выражений. На выходе получаются метки или координаты в одном из пространств валентность-интенсивность, о которых говорилось раньше.

Основную роль в распознавании эмоций играет выбранный классификатор. На данный момент наиболее часто используемые классификаторы это: линейный дискриминантный анализ, k-ый ближайший сосед, Гауссова смешанная модель, метод опорных векторов, искусственная нейронная сеть, дерево решений и скрытые сети Марковский моделей и т.д. Рассмотрим некоторые из них.

1. Линейный дискриминантный анализ – это метод статистики и машинного обучения, применяемый для нахождения линейных комбинаций признаков, наилучшим образом разделяющих два или более класса объектов или событий [7].   
   Линейный дискриминантный анализ используется для распознавания лиц как первый этап распознавания: ЛДА сокращает перед классификацией количество признаков до количества более удобного в работе. Новые размерности – это линейные комбинации значений пикселей. А новые размерности уже далее используются для классификации.

ЛДА в основном используется в случае, когда классы не перекрываются, однако изредка может работать и с частично перекрывающимися классами.

1. k-ый ближайший сосед – алгоритм для автоматической классификации объектов. Главным принципом данного метода является то, что классифицируемый объект присваивается тому классу, соседей которого больше среди соседей классифицируемого объекта.

Алгоритм может применяться для выборок с большим количеством атрибутов, но при этом необходимо определить функцию расстояния. Кроме того, необходимо нормализовать данные из-за разницы в диапазоне значений атрибутов.

При этом предполагается, что набор объектов, предварительно прошедших классификацию, уже есть.

1. Метод опорных векторов – набор алгоритмов обучения с учителем, использующихся в задачах классификации и регрессионного анализа. Основной идеей метода является перевод имеющихся векторов в пространство более высокой размерности и поиск разделяющей их гиперплоскости. При этом параллельно по обеим сторонам строятся гиперплоскости, а расстояние до двух параллельных гиперплоскостей (зазор) максимизируется для минимизации средней ошибки классификатора.

Данный метод широко применяется для классификации изображений, имеет высокую точность, для распознавания рукописного текста, используется также в биологии для распознавания белков по составу.

Минусами данной модели являются: требуется маркировка всех входных данных, напрямую применим только для задач с двумя классами (однако может быть расширен для многоклассовых задач), параметры получившейся модели сложно интерпретировать.

1. Скрытая модель Маркова – статистическая марковская модель, в которой моделируемая система принимается за марковский процесс со скрытыми состояниями. Вместо состояний наблюдаются выходные параметры. Каждое состояния марковской сети имеет вероятностное распределение среди возможных выходных значений. Таким образом, последовательность выходных значений сети дает представление о последовательности состояний внутри сети.

Скрытые модели Маркова хорошо зарекомендовали себя при автоматическом распознавании речи, так как способны описывать процессы и сигналы.

В данной работе не будет рассмотрено использование классификатора для распознавания эмоций по сигналам ЭЭГ, однако это является перспективным направлением при развитии проекта.

## Фрактальная размерность

В своей работе [8] исследователи из Кореи Kwang-Eun Ko, Hyun-Chang Yang, and Kwee-Bo Sim используют ЭЭГ сигналы, а точнее относительные степени сигналов ЭЭГ и Байесовскую сеть для определения эмоционального состояния человека.

В другой работе описывается, как используется фрактальная размерность для определения эмоционального состояния по данным ЭЭГ [9]. Исследователи из Сингапура проводят два эксперимента с прослушиванием музыкальных фрагментов несколькими людьми. В этот момент у них снимают данные ЭЭГ и просят отметить, какую из шести эмоций они испытывают при прослушивании. Затем, из значений ЭЭГ определяются значения фрактальной размерности, после чего устанавливается связь между значениями размерности и характеристиками модели: валентности и интенсивности. Таким образом, определив характеристики для эмоциональной модели, становится возможным определить конкретную эмоцию.

Фрактальная размерность – это один из способов определения размерности множества в метрическом пространстве. Фрактальная размерность была впервые введена как коэффициент, описывающий геометрически сложные формы, для которых детали являются более важными, чем полный рисунок [10]. Фрактальную размерность можно использовать как характеристику кривой. Кроме того, нелинейные системы, такие как данные ЭЭГ могут быть подвергнуты анализу при помощи фрактальной размерности [11]. В [12] также сказано, что фрактальная размерность отражает изменения в ЭЭГ сигнале, и более того, в [13] указано, что фрактальная размерность ЭЭГ сигнала мозга отличается при выполнении различных задач, связанных с умственной деятельностью.

Размерность Минковского — это один из способов задания фрактальной размерности ограниченного множества в метрическом пространстве, определяется следующим образом:

https://habrastorage.org/getpro/habr/post_images/52b/569/bde/52b569bdef3df13199731903cdcf9808.gif,

где N(ε) минимальное число множеств диаметра ε, которыми можно покрыть исходное множество.

Если предел не существует, то рассматривают верхний и нижний пределы и говорят соответственно о верхней и нижней размерности Минковского. Верхняя и нижняя размерности Минковского тесно связанны с размерностью Хаусдорфа, интуитивно это легко уловить по способу задания размерности. Обычно упомянутые три размерности совпадают, и только в очень специфичных случаях имеет смысл их различать, но это не наши случаи.

Размерность Минковского имеет так же другое название — box-counting dimension, из-за альтернативного способа ее определения, который, кстати, дает подсказку к способу вычисления этой самой размерности. Рассмотрим двумерный случай, хотя аналогичное определение распространяется и на n-мерный случай. Возьмем некоторое ограниченное множество в метрическом пространстве, например черно-белую картинку, нарисуем на ней равномерную сетку с шагом ε, и закрасим те ячейки сетки, которые содержат хотя бы один элемент искомого множества (Рисунок 1.6(a)). Далее начнем уменьшать размер ячеек, т.е. ε, тогда размерность Минковского будет вычисляться по вышеприведенной формуле, исследуя скорость изменения отношения логарифмов.

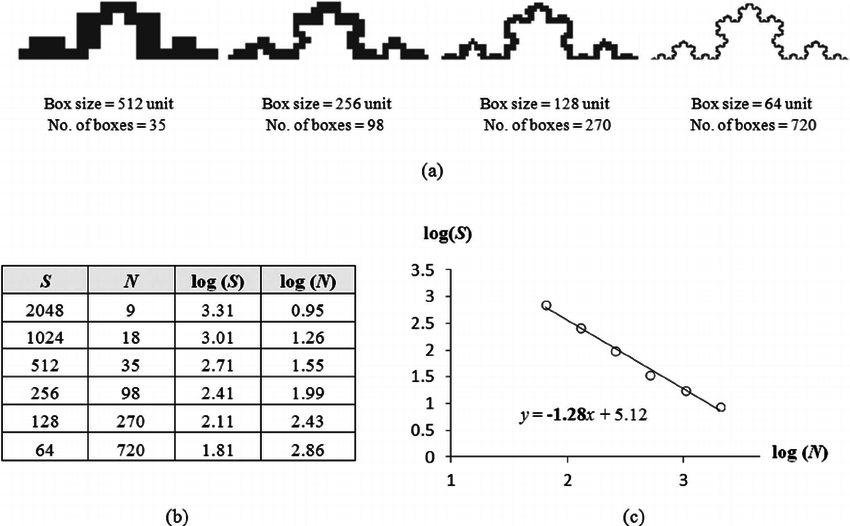


Рисунок 1.6. Пример применения размерности Минковского

Алгоритм выводится следующим образом: обозначим за Dbc приближенное значение размерности Минковского. Запишем определение этой размерности, убрав предел, его мы будем имитировать в итерациях, в которых будет изменяться размер ячеек. Если зафиксировать размеры ячеек ε и рассматривать Dbc как неизвестное, то легко заметить, что приведенное выражение является формулой линии. Можем запустить цикл по различным размерам ячеек ε и записывать результат. Нанесем эти результаты на график и построим линию регрессии для полученного множества данных, это значение и будет являться аппроксимацией фрактальной размерности Минковского.

## Существующие решения

В результате поисков, среди существующих систем полностью готовых решений для разрешения проблемы распознавания эмоций посредством сигнала ЭЭГ не было найдено. Среди близких систем по технологии и идеологии можно выделить следующие проекты.

**MyndPlay**

Компания MyndPlay выпустила систему управления сюжетами видео фильмов и игр при помощи данных от нейроинтерфейса с аналогичным названием.

Плюсы решения:

* Использование существующей платформы NeuroSky TGAM, что обуславливает низкую цену разработки;
* Решение очень интересно для пользователей в развлекательных целях;
* Решение позволяет создавать фильмы с контролируемым сюжетом для домашнего просмотра в развлекательных и развивающих, образовательных целях;

Минусы решения:

* Реализация проекта на платформе PC;
* Работа по контролю контента с использованием данных без верификации;
* Отсутствие многопользовательских решений;
* Невозможность из-за ограничений возможностей платформы работать с эмоциональными и иными однозначно определяемыми состояниями НС человека.

Резюме: Проект с высоким потенциалом на момент разработки, однако, использование платформы с ограниченным потенциалом сдерживает возможности роста и развития в соответствии с требованиями рынка 2016 года для создания контролируемых сред, игровых сценариев и т.д.

**Проект «SenseLabs VERSUS»**

Новое техническое решение - нейроинтерфейс 2015 года выпуска. Основной акцент разработки - на играх/тренингах БОС ЭЭГ. Основой устройства является чип TGAM, разработанный компанией NeuroSky, который выявляет состояния «концентрации» и «медитации» пользователя. Важный минус, что все сервисы вынесены онлайн с оплатой ежемесячной подписки от 20 USD в месяц.

Плюсы разрабатываемого решения по сравнению с имеющимися:

* реализация на мобильной платформе (ОС Android) в отличие от программы MyndPlay (PC, Mac);
* использование общедоступного формата видео файлов в отличие от программы MyndPlay, где требуются специализированные файлы;
* возможность расширения используемой библиотеки видео контента;
* наличие функционала учетной записи пользователя для сохранения данных и возможности дальнейшего анализа;
* возможность подключения администрирующего устройства с целью наблюдения и контроля третьей стороной (например, врачом);
* алгоритм работы, при котором пользователь не только развлекается, а учится управлять своим состоянием за счет создания биологической обратной связи.

Минусы разрабатываемого решения по сравнению с имеющимися:

* отсутствие интерактивности при просмотре видео, как это реализовано в MyndPlay (изменение сюжета в зависимости от состояния);
* необходимость предварительного введения пользователя в алгоритм работы системы.

Таблица 1.1. Сравнение существующих аналогов.

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | Muse | MyndPlay | SenseLabs VERSUS |
| **Данные** | Raw EEG at 220Hz, processed FFT, Absolute/relative band powers | Raw EEG at 512Hz,  processed power spectrum | Raw EEG data,  processed power spectrum |
| **API** | Android(Java),  Windows(C++ библиотека),  iOS (Objective-C) | Информация не найдена. Возможно, в разработке | Информация не найдена. Возможно, в разработке |
| **Платформа** | Windows, Android,  iOS | Windows, Mac,  Android, iOS | iOS, Android |
| **Цена** | $249.00 USD | £179.00 | Пока недоступно |

Таблица 1.1 Сравнение существующих аналогов. Окончание

Подводя итог анализа существующих систем, можно сказать, что на рынке имеется достаточное количество готовых технических решений для снятия показаний данных ЭЭГ и элементарного отображения этой информации на экранах персональных компьютеров и смартфонов. Однако среди этих систем нет решения для определения психоэмоционального состояния. Используя готовое техническое решение, становится возможным разработать систему, связав готовые аппаратные компоненты при помощи программного обеспечения в систему для определения психоэмоционального состояния по данным ЭЭГ.

## Биологическая обратная связь

Биологическая обратная связь (БОС) – метод лечения и реабилитации, основанный на развитии у пациента навыков самоконтроля и саморегуляции различных функций организма для улучшения общего состояния [14].

Для применения метода биологической обратной связи необходимо в ходе сеанса с помощью приборов или компьютерных комплексов регистрировать физиологические показатели какой-либо функциональной системы организма или органа пациента, а затем отобразить полученную информацию для создания обратной связи посредством зрительных или звуковых сигналов.

Так как в данной работе планируется применить характеристику фрактальной размерности для классификации психоэмоционального состояния, то ее же и стоит применить для отображения и создания биологической обратной связи.

Таким образом, система не только позволит пользователю отчасти увидеть, как функционирует мозг, но и, используя специальные методические приемы, самостоятельно корректировать работу функциональных систем собственного организма, развивать навыки самоконтроля и саморегуляции с помощью скрытых физиологических резервов. При этом необходимо подчеркнуть, что речь идет о приобретении навыков самоконтроля, при котором пациент обучается тому, как помогать себе самостоятельно [14].

## Цель и задачи

Подводя итог изложенного выше, утвердим целью данной работы следующее: разработать систему определения психоэмоционального состояния пользователя на основе данных ЭЭГ, получаемых с гарнитуры Muse, с использованием биологической обратной связи для ОС Android.

При этом в работе решаются следующие задачи:

* получение данных ЭЭГ с гарнитуры MUSE,
* разработка алгоритма определения состояния пользователя по данным ЭЭГ,
* организация вывода на экран текущего состояния пользователя и видео контента,
* организация взаимодействия двух устройств по каналу связи Bluetooth.

# ПРОЕКТИРОВАНИЕ СИСТЕМЫ ОПРЕДЕЛЕНИЯ ПСИХОЭМОЦИОНАЛЬНОГО СОСТОЯНИЯ

## 2.1. Требования

**Требования к аппаратной части системы**

Система должна обеспечивать:

* измерение и регистрацию подэлектродного сопротивления и межэлектродного потенциала непрерывно и синхронно с регистрацией биометрических параметров с частотой дискретизации не ниже 512 Гц;
* электропитание разрабатываемого АПК от встроенной аккумуляторной батареи с возможностью её подзарядки как через стандартный порт USB от ПК, так и от штатного, поставляемого в комплекте зарядного устройства с питанием от сети переменного тока 110­220 в 50/60 Гц;
* среднюю потребляемую мощность в рабочем режиме не более 0.5 ВТ;
* пиковую потребляемую мощность не более 1,5 ВТ;
* время автономной работы от аккумуляторной батареи не менее 2 часов.

Аппаратная часть должна также обеспечивать:

* Наличие возможности механического сопряжения 3D­очков и гарнитуры в единую конструкцию.
* Обеспечение плотного примыкания 3D­очков к области глаз.
* Сопряжение по Bluetooth гарнитуры и 3D­очков.
* Точечное приклеивание резинового профиля к 3D­очкам (с возможностью отсоединения при необходимости замены профиля).
* Возможность использовать карту памяти для библиотеки сюжетов (объем – 8­16 Гбайт).
* Наличие дополнительного конструктивного элемента для приема и передачи аудиосигнала ­ стандартный разъем для наушников 3.5mm jack.

**Требования к программной части системы**

Реализовать алгоритм работы основного программы, при котором она находится в одном из двух режимов работы: состоянии «покоя», в котором воспроизводится видео контент и состоянии «тревоги», в котором пользователю предлагаются советы по снижению уровня тревожности. Переключение между режимами должно происходить автоматически на основании информации о психоэмоциональном состоянии.

Отобразить следующую информацию в основной программе в режиме покоя:

* видео контент, то есть видео файл, выбранный автоматически основной программой, либо с помощью администрирующей программы;
* значение характеристики, по которой определяется психоэмоциональное состояние, в виде полоски, меняющей цвет и размер по мере приближения к состоянию тревоги;
* состояние подключения гарнитуры MUSE в виде надписи «ПОДКЛЮЧЕНО» или «ОТКЛЮЧЕНО»;
* уровень заряда батареи гарнитуры MUSE в виде процентов от 1 до 100;
* качество сигнала с каждого из 4-ех датчиков в виде разноцветных кругов, скрываемых или отображаемых в зависимости от качества сигнала с датчиков;
* состояние подключения администрирующего устройства в виде надписи «ПОДКЛЮЧЕНО» или «ОТКЛЮЧЕНО».

Информация для отображения основной программой в режиме тревоги:

* значение характеристики, по которой определяется психоэмоциональное состояние, в виде полоски, меняющей цвет и размер по мере приближения к состоянию тревоги;
* заранее выбранное художественное изображение в качестве фона;
* надпись с советом по снижению уровня тревожности.

Реализовать алгоритм работы администрирующей программы, когда при наличии подключения к основной программе отображается информация о сеансе, перечисленная ниже, а также можно выполнить управление воспроизводимым контентом.

Информация для отображения администрирующей программой:

* состояние подключения к основной программе в виде строки «ПОДКЛЮЧЕНО» или «ОТКЛЮЧЕНО»;
* состояние подключения гарнитуры MUSE к основной программе в виде строки «ПОДКЛЮЧЕНО» или «ОТКЛЮЧЕНО»;
* уровень заряда батареи основного устройства в виде процентов от 1 до 100;
* уровень заряда гарнитуры MUSE в виде процентов от 1 до 100;
* качество прилегания датчиков гарнитуры MUSE в виде 4ех кругов, скрываемых или показываемых в зависимости от качества прилегания датчиков;
* название воспроизводимого в данный момент видео файла;
* длительность воспроизводимого в данный момент видео файла;
* метка времени текущего состояния воспроизведения видео файла;
* график характеристики, по которой определяется психоэмоциональное состояние.

Требования к функциям администрирующей программы по контролю за воспроизведением видео файлов:

* приостановка и воспроизведение приостановленного видео файла;
* переключение на следующий или предыдущий видео файл;
* перемотка воспроизводимого видео файла на конкретный момент времени;
* отображение списка всех доступных для воспроизведения видео файлов;
* воспроизведение конкретного выбранного из предыдущего списка видео файла.

Требования к функциям администрирующей программы по хранимой информации:

* создание и удаление учетных записей пользователей, точное описание данных которых будет указано далее;
* просмотр списка существующих учетных записей;
* просмотр списка проведенных сеансов использования для выбранной учетной записи;
* просмотр информации о конкретном сеансе использования, точное описание данных которого будет указано далее;
* внесение комментариев врача и указание процедуры, которая проводилась для пользователя во время сеанса использования.

## Компоненты

Аппаратная часть представляет собой следующую систему:

* устройство на базе ОС Android 5.0 в форм-факторе очков;
* устройство Muse - гарнитура, считывающая сигналы ЭЭГ с мозга и передающее их по каналу Bluetooth;

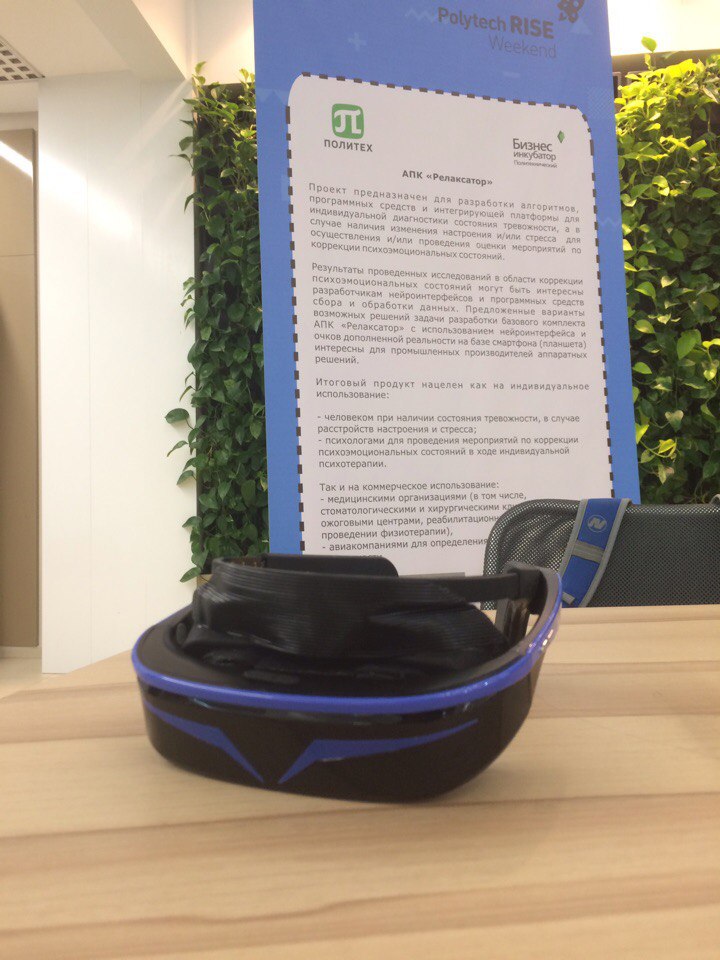
 

Рисунок 2.1. Аппаратная часть системы

* устройство на базе ОС Android 4.0 или выше с возможностью установления связи по Bluetooth каналу и наличием средств ввода-вывода.

Выбранные компоненты удовлетворяют всем перечисленным ранее требованиям к аппаратной части системы.

Программная часть разрабатываемой системы будет состоять из нескольких компонентов:

* основная программа для ОС Android, реализующая связь с гарнитурой Muse, в которую входят:
  + модуль определения психоэмоционального состояния по данным ЭЭГ;
  + модуль визуализации, ответственный за воспроизведение видео контента;
  + хранилище видео контента (раздел на физическом носителе с видео файлами, либо дополнительная надстройка над разделом в виде БД, размечающее видео контент по эмоциональной окраске, предполагаемым группам пользователей);
  + обработчик пакетов команд от администрирующей программы, который формирует ответ в виде пакета данных о текущем состоянии, либо передающий сигналы управления модулю визуализации;
* администрирующая программа для ОС Android, в которую входят:
  + модуль, обрабатывающий пакеты данных от основной программы;
  + модуль управления, ответственный за визуализацию данных из пакетов от основной программы, а также обрабатывающий команды, поступающие от пользователя с помощью графического интерфейса;
  + модуль управления учетными записями пользователей.

## Средства разработки

* Среда разработки Android Studio, Android SDK
* Язык Java
* СУБД SQLite
* Библиотеки:
  + ORMlite
  + Gson
  + MPAndroidChart

Выбор программных средств продиктован в первую очередь выбранной платформой для разработки: операционной системой Android. Среда разработки Android Studio является стандартным инструментом при разработке для указанной операционной системы, поставляемой компанией Google, разработчиком операционной системы. Со средой разработки поставляется и комплект средств разработки Android SDK. Язык программирования Java также продиктован официальным SDK, кроме того, он является одним из самых популярных языков программирования в мире, хорошо документирован и обладает огромной базой примеров использования. СУБД SQLite выбрана ввиду встроенной в ОС Android и реализованной в Android SDK поддержки данной СУБД.

При работе с базой данных из программы планируется использовать библиотеку ORMLite, реализующую технологию Object-Relation Mapping для языка Java. Библиотекой поддерживается ряд СУБД, среди которых SQLite [15], а также существует адаптированная версия библиотеки для ОС Android.

С помощью библиотеки MPAndroidChart в программе будут выводиться графики для данных. Библиотека выбрана из-за хорошей документации и наличия достаточного количества примеров использования.

## Взаимодействие

Описать взаимодействие компонентов систем можно с помощью схемы движения данных (см. Рисунок 2.2). На ней представлены перечисленные выше компоненты, информационные связи и данные, которые передаются по этим связям. Также указаны точки входа информации с устройств и выводимой на них информации.

Основной путь движения данных выглядит так:

1. с гарнитуры Muse по каналу Bluetooth передаются данные ЭЭГ;
2. программа принимает Bluetooth-пакеты с данными ЭЭГ, обрабатывает их и сохраняет в памяти;
3. компонент определения состояния по обработанным данным ЭЭГ вычисляет состояние человека и сохраняет его в памяти;
4. компонент визуализации выводит видео контент на основании текущего состояния.

Еще один путь движения данных:

1. Модуль управления в администрирующей программе отправляет пакет команды к основной программе.
2. Обработчик команд в основной программе выполняет требуемые действия, формирует пакет данных и отправляет ответ модулю управления.
3. Обработчик пакетов данных в администрирующей программе передает данные сеанса из пакета модулю управления

Модуль управления обновляет данные на графическом интерфейсе администрирующего устройства.

## Обработчик данных ЭЭГ

Гарнитура Muse является Bluetooth-устройством, по принципу работы схожим с датчиком. Во время своей работы устройство отправляет пакеты данных по каналу Bluetooth в формате OSC (Open Sound Protocol). О разборе пакетов в формате OSC нет необходимости заботиться: эта функция ложится на предоставляемый официальный SDK гарнитуры Muse.

Пакеты от гарнитуры бывают двух типов:

1. пакеты с данными о соединении и самом устройстве,
2. пакеты с данными о сигналах ЭЭГ и внутренних параметрах устройства.

Следовательно, кроме установления подключения с гарнитурой Muse, необходимо будет зарегистрировать обработчики для пакетов каждого из двух типов. После регистрации обработчиков и соединения с гарнитурой, для каждого принятого пакета данных будет выполнен соответствующий обратный вызов. Данные из пакета сохранятся в буфер для последующей обработки.



Рисунок 2.2. Схема движения данных

## Работа с администрирующим устройством

Взаимодействие с администрирующей программе происходит по модели сервер-клиент с использованием собственного протокола. Пакеты передаются с помощью канала связи Bluetooth. Клиент инициирует подключение к серверу, после установления соединения отправляет запросы в виде пакетов команд. Пакеты команд принимаются сервером, на них формируется ответ – пакет данных, и отправляется обратно клиенту. По протоколу передаются пакеты команд как запросы от клиента (администрирующей программы) и пакеты данных как ответы от сервера (основной программы).



Рисунок 2.3. Схема взаимодействия программ c помощью собственного протокола

В таблицах ниже описаны составляющие пакета команд (Таблица 2.1) и возможные команды (Таблица 2.2), пакета данных (  
Таблица 2.3).

Таблица 2.1. Описание пакета команд

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| command | Command | Команда управления |
| arguments | Object[] | Аргументы команды |

Таблица 2.2. Описание возможных команд пакета управления

|  |  |  |
| --- | --- | --- |
| **Команда** | **Аргументы** | **Описание** |
| GET | отсутствуют | получение актуального пакета данных |
| PLAY | целочисленный идентификатор | проиграть файл с указанным идентификатором |
| PAUSE | отсутствуют | приостановить или продолжить воспроизведение видео |
| LIST | отсутствуют | получение списка доступных для воспроизведения видео файлов |
| REWIND | новая позиция для воспроизведения | установить текущую позицию проигрывания видео |
| NEXT | отсутствуют | воспроизвести следующий видео файл |
| PREV | отсутствуют | воспроизвести предыдущий видео файл |

Таблица 2.3. Описание пакета данных

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| headsetBatteryPercent | Integer | Процент заряда батареи устройства, где запущено основная программа |
| museState | Boolean | Статус подключения гарнитуры Muse |
| museBatteryPercent | Integer | Процент заряда батареи гарнитуры Muse |
| museSensorsState | Boolean[] | Качество прилегания датчиков гарнитуры Muse |
| alphaPct | Integer | Вычисленный процент альфа-ритма |
| betaPct | Integer | Вычисленный процент бета-ритма |
| isPanic | boolean | Состояния человека |
| videoName | String | Название проигрываемого видео файла |
| videoState | Boolean | Проигрывается ли видео файл |
| duration | Integer | Длительность проигрываемого файла в секундах |
| currentPosition | Integer | Текущая позиция проигрываемого файла в секундах |
| videoList | ArrayList<VideoItem> | Список доступных для воспроизведения видео файлов |

Таблица 2.3 Описание пакета данных. Окончание

Таблица 2.4. Описание полей типа VideoItem

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| id | int | Идентификатор |
| name | String | Название |
| duration | int | Длительность в секундах |

## База данных

Администрирующая программа предоставляет доступ к управлению учетными записями пользователей, хранящихся в БД. Прежде чем сеанс использования устройства начинается, происходит выбор учетной записи пользователя. В учетной записи пользователя записаны основные данные о нем, а так же информация о проведенных сеансах. В информации о сеансе входит изменяющееся во времени психоэмоциональное состояние пользователя.

Сформулируем требования по хранению данных.

Пользователь:

* имя, фамилия,
* дата рождения,
* пол.

Сеанс использования:

* начало сеанса,
* окончание сеанса,
* данные о состоянии пользователя в каждый момент времени,
* проводимая процедура,
* комментарий врача,
* набор воспроизводимых видео файлов во время сеанса с временными метками.

Возрастные категории:

* нижний порог возрастной категории,
* верхний порог возрастной категории.

Тэги:

* название.

Данные о видео:

* название,
* длительность,
* имя файла,
* набор подходящих возрастных категорий,
* набор тэгов, характеризующих видео.

Получившиеся требования к хранимым данным можно формализовать следующим образом (см. Рисунок 2.4. Схема связей в БД).



Рисунок 2.4. Схема связей в БД

В Таблице 2.5 описаны колонки и соответствующие типы данных, которые содержит таблица пользователей. Первичный ключ таблицы – колонка id, целое число. Имя и фамилия – колонки firstname и lastname – строкового типа (text). Колонка, хранящая дату рождения, тоже строкового типа. Данные будут преобразовываться с помощью средств языка программирования.

Таблица 2.5. Описание полей таблицы User

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| id | integer | N |
| firstname | text | N |
| lastname | text | N |
| birthdate | text | N |
| sex | text | N |

В Таблице 2.6 хранятся данные о сеансе использования. Целочисленный идентификатор – колонка id – первичный ключ таблицы. Колонка user\_id – внешний ключ - ссылка таблицу Users, идентифицирующая пользователя, который проводил сеанс использования. Время начала и окончания - колонки start и finish соответственно – строкового типа. Колонка data – двоичного типа. В ней будет храниться сериализованный массив данных сеанса. Колонки action и comment предназначены для описания проводимой при сеансе использования процедуры и комментария, могут отсутствовать.

Таблица 2.6. Описание столбцов таблицы Seance

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| id | integer | N |
| user\_id | integer | N |
| start | text | N |
| finish | text | N |
| data | blob | N |
| action | text | Y |
| comment | text | Y |

Таблица 2.7 содержит данные об имеющихся возрастных категориях. Возрастная категория имеет идентификатор (целочисленная колонка id, являющейся первичным ключом таблицы).

Таблица 2.7. Описание столбцов таблицы AgeCategory

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| id | integer | N |
| top | integer | N |
| bottom | integer | N |

Таблица 2.8 содержит список используемых в системе тэгов для характеристики видео контента, используемого системой. Колонка id – целочисленный первичный ключ таблицы.

Таблица 2.8. Описание столбцов таблицы Tag

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| id | integer | N |
| name | text | N |

Используемый системой видео контент описан в таблице 2.9. Целочисленный первичный ключ – колонка id. Название видео в произвольном формате содержится в колонке name. Путь к файлу содержится в колонке filename. Длительность видео контента в секундах хранится в колонке duration.

Таблица 2.9. Описание столбцов таблицы Video

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| id | integer | N |
| name | text | N |
| filename | text | N |
| duration | integer | N |

Таблица 2.10 является реализацией отношения многие ко многим между тэгами и видео контентом.

Таблица 2.10. Описание столбцов таблицы VideoTag

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| video\_id | integer | N |
| tag\_id | integer | N |

Таблица 2.11 аналогично предыдущей таблице является реализацией отношения многие ко многим между видео контентом и возрастными категориями.

Таблица 2.11. Описание столбцов таблицы VideoAgeCategory

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| video\_id | integer | N |
| agecategory\_id | integer | N |

Таблица 2.12 тоже является реализацией отношения многие ко многим между сеансами и видео. Однако в ней так же содержится дополнительная информация о том, в какой временной промежуток воспроизводилось указанное видео.

Таблица 2.12. Описание столбцов таблицы SeanceVideo

|  |  |  |
| --- | --- | --- |
| Название | Тип | Может быть нулем |
| seance\_id | integer | N |
| video\_id | integer | N |
| seance\_timestart | integer | N |
| seance\_timeend | integer | N |

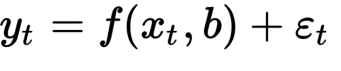
## 2.7. Модуль определения состояния

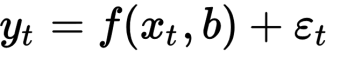
Модуль определения состояния представляет собой реализацию алгоритма, на вход которому подаются обработанные данные ЭЭГ с соответствующего модуля. Ранее было сказано, что в данной работе будет использоваться такая характеристика кривой как фрактальная размерность.

Схема алгоритма вычисления фрактальной размерности представлена ниже (Рисунок 2.5). Для вычисления необходимы будут две процедуры, начнем с линейной регрессии. Вообще решить задачу линейной регрессии можно различными способами, чаще всего для этого используется метод градиентного спуска и метод наименьших квадратов (Normal equations). Воспользуемся методом наименьших квадратов.

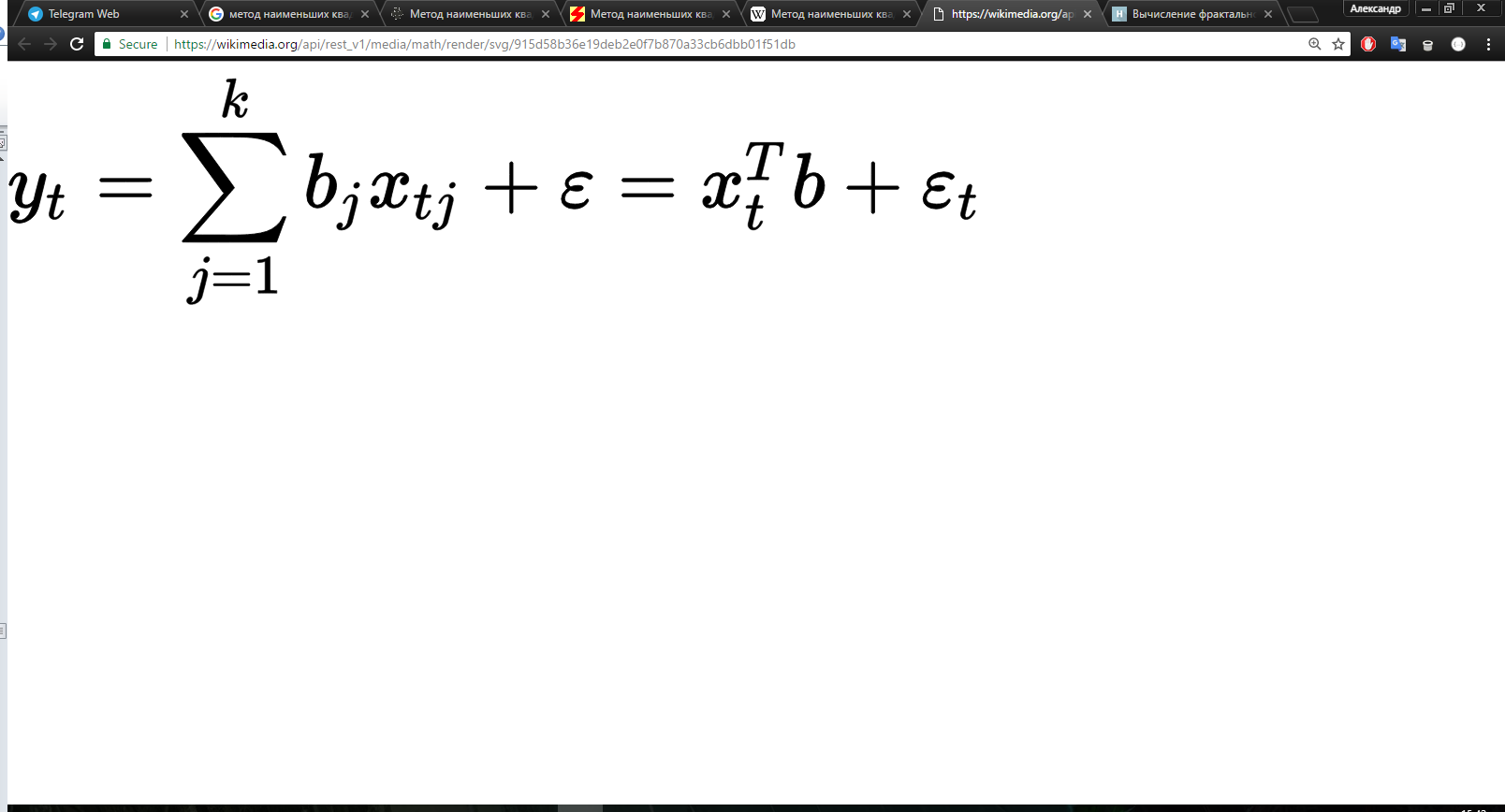
Пусть имеется n значений некоторой переменной y (это могут быть результаты наблюдений, экспериментов и т. д.) и соответствующих переменных x. Задача заключается в том, чтобы взаимосвязь между y и x аппроксимировать некоторой функцией f(x,b), известной с точностью до некоторых неизвестных параметров b, то есть фактически найти наилучшие значения параметров b, максимально приближающие значения f(x,b) к фактическим значениям y.

В регрессионном анализе используются вероятностные модели зависимости между переменными:

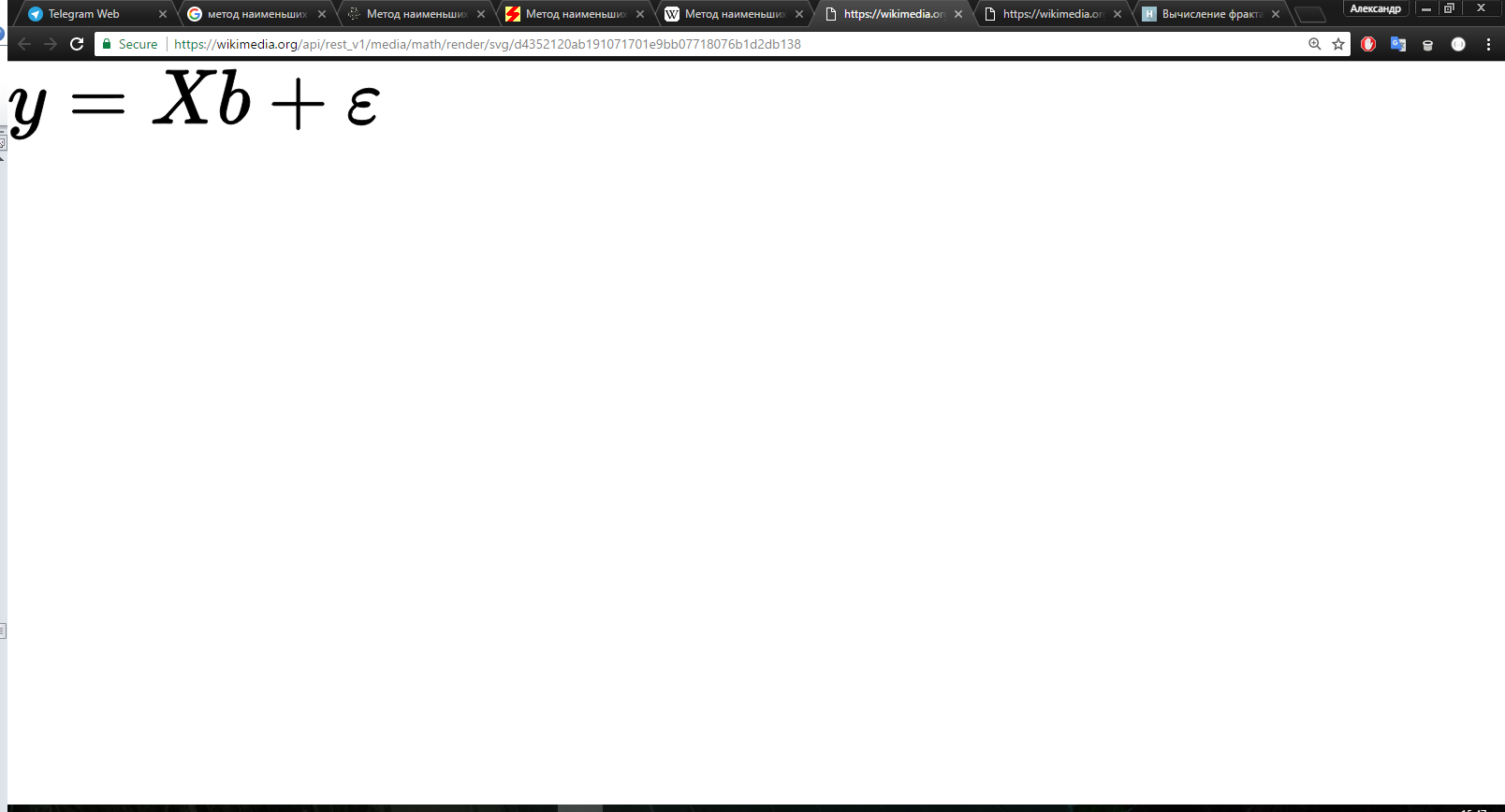
,

где  — случайные ошибки модели.

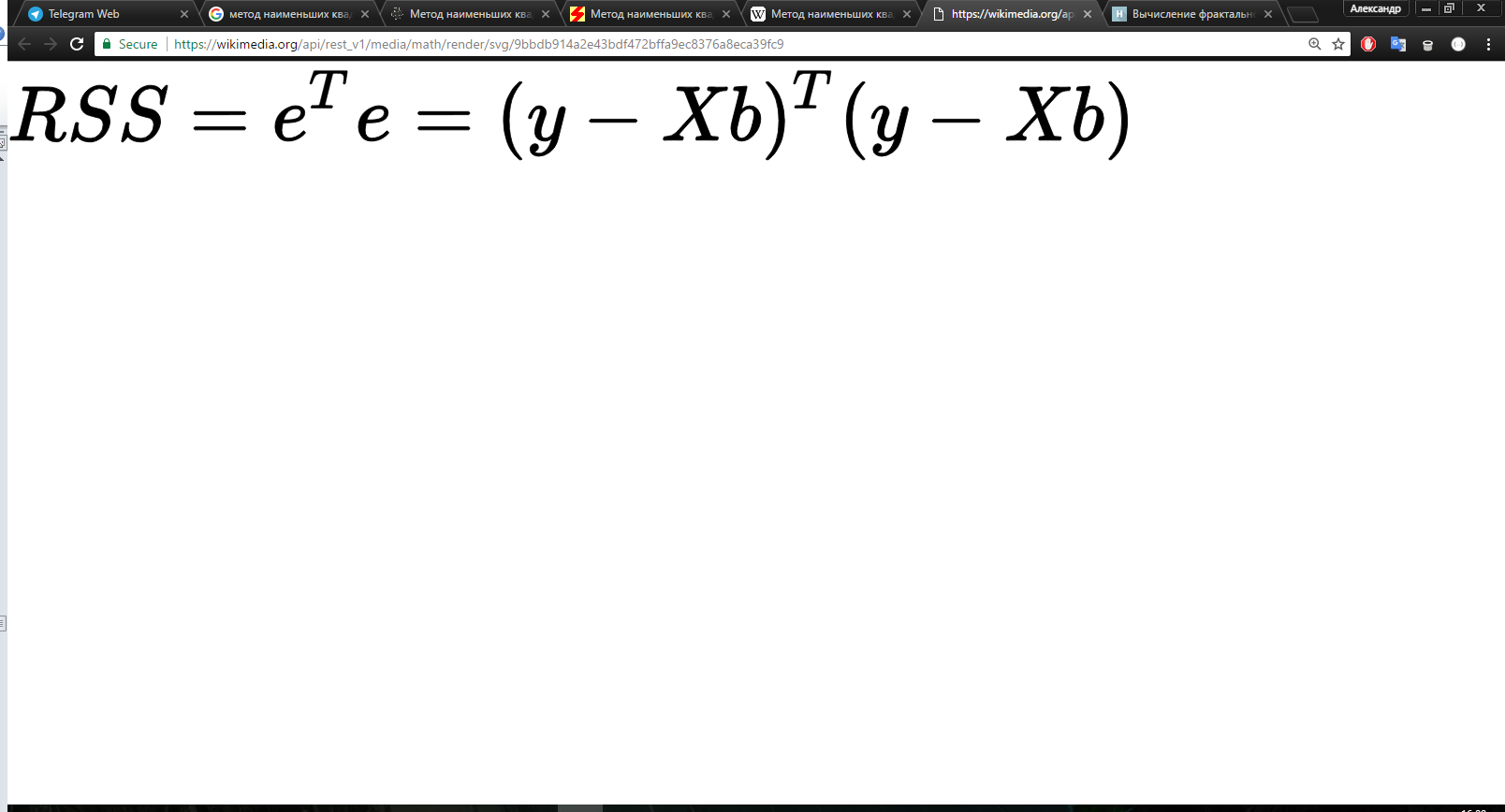
Пусть регрессионная зависимость является линейной:



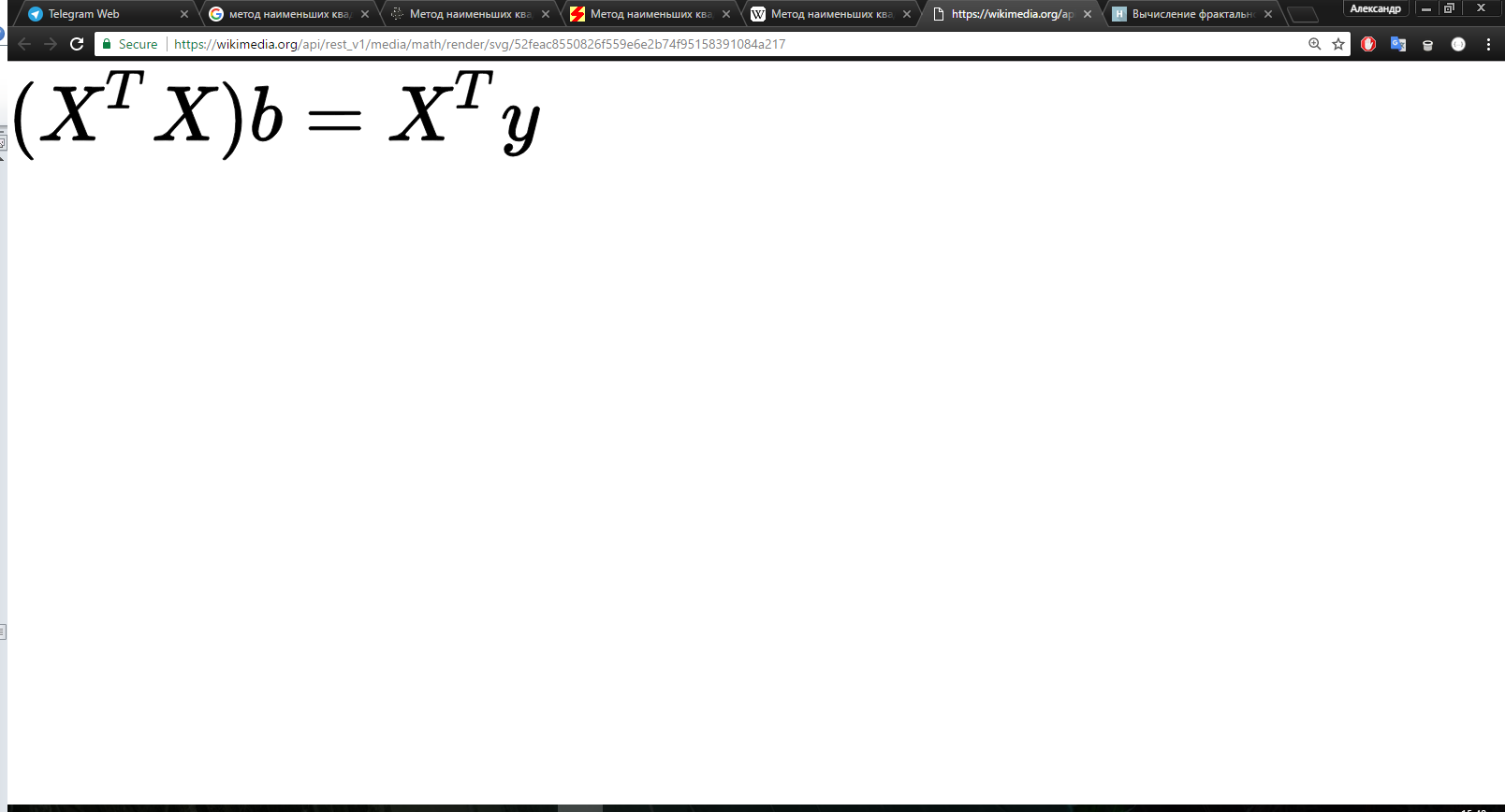
Пусть *y* — вектор-столбец наблюдений объясняемой переменной, а X{\displaystyle X} — это {\displaystyle ({n\times k})}матрица наблюдений факторов (строки матрицы — векторы значений факторов в данном наблюдении, по столбцам — вектор значений данного фактора во всех наблюдениях). Матричное представление линейной модели имеет вид:



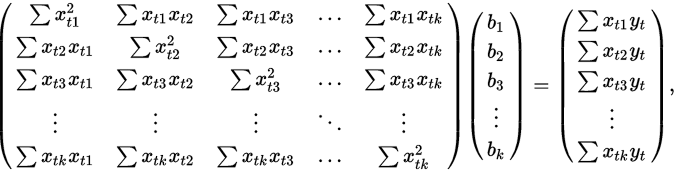
Сумма квадратов остатков регрессии будет равна:



Дифференцируя эту функцию по вектору параметров b и приравняв производные к нулю, получим систему уравнений (в матричной форме):



В расшифрованной матричной форме эта система уравнений выглядит следующим образом:



где все суммы берутся по всем допустимым значениям t.

Если в модель включена константа, то в левом верхнем углу матрицы системы уравнений находится количество наблюдений, а в остальных элементах первой строки и первого столбца — просто суммы значений переменных, и первый элемент правой части системы — сумма всех y.

В векторизованном виде решение линейной регрессии записывается следующим образом:

https://habrastorage.org/getpro/habr/post_images/ba0/494/a48/ba0494a48f1f5526a4fa5d942f915596.gif,

где тета – искомый вектор параметров, матрица X размером 2 на 2 описана ранее, а вектор y – вектор значений экспериментов.

Обратную матрицу будем искать по следующей формуле:

https://habrastorage.org/getpro/habr/post_images/6d7/d80/211/6d7d802118df6e04102af9359b1a36e5.png



Рисунок 2.5. Схема алгоритма вычисления фрактальной размерности

# РАЗРАБОТКА ПРОГРАММНОЙ ЧАСТИ СИСТЕМЫ

## Система классов

Проще всего описать разработанную систему классов, посмотрев на указанную ранее схему компонентов системы (Рисунок 3.1). Укажем, какие классы были разработаны, какие поля и методы содержат, к какому модулю относятся, а также краткое описание их предназначения.



Рисунок 3.1. Схема компонентов системы

## Обработка данных ЭЭГ

Проектирование системы классов начинается с точки входа данных с устройства. Такой точкой является гарнитура Muse с каналом связи Bluetooth. К данной гарнитуре на официальном сайте предлагается SDK для языка Java. Основным классом для работы в ОС Android является класс Activity. Для выделения функциональности, связанной с соединением гарнитуры Muse и устройства на ОС Android, опишем абстрактного наследника класса Activity под названием MuseActivity.

Класс MuseManager позволяет получать список обнаруженных устройств (гарнитур Muse). Используя статические методы класса MuseManager, можем получить ссылку на экземпляр класса Muse, который представляет собой устройство Muse. После того как устройство для подключения будет выбрано, ссылка на него сохраняется в поле класса MainActivity.



Рисунок 3.2. Схема классов для соединения с гарнитурой Muse

Для установления соединения с гарнитурой Muse необходимо выполнить следующие действия:

1. Установить контекст для работы классу MuseManager, чтобы указать, из какой активности в данный момент происходит работа.
2. Зарегистрировать обработчики пакетов от устройства для двух типов пакетов соответственно: conectionListener, dataListener.
3. Получить ссылку на экземпляр класса muse, сохранив ее в соответствующее поле класса MuseActivity.
4. Запустить обмен пакетами с выбранной гарнитурой, инициировав тем самым подключение к гарнитуре.

Данные действия производятся при создании экземпляра класса MuseActivity в методе setupMuseManager, который вызывается в методе onCreate.

Листинг 3.1. Метод setupMuseManager

|  |
| --- |
| private void setupMuseManager() {  MuseManager.setContext(this);  MuseManager.registerMuseListeners(  connectionListener,   dataListener);  muse = MuseManager.getMuse();  muse.runAsynchronously();  } |

После запуска обмена пакетами, в зарегистрированном ранее экземпляре класса ConnectionListener будет вызван переопределенный метод receiveMuseConnectionPacket, в котором будет вызван метод updateConectionStatus для обновления поля connectionState класса MuseActivity. В этом поле хранится текущее состояние подключения к гарнитуре.

Листинг 3.2. Класс ConnectionListener

|  |
| --- |
| public ConnectionListener(final WeakReference<MuseActivity> activityRef) {  this.activityRef = activityRef;  }   @Override  public void receiveMuseConnectionPacket(  final MuseConnectionPacket p,   final Muse muse)  {  final ConnectionState current = p.getCurrentConnectionState();  MuseActivity museActivity = activityRef.get();  museActivity.updateConnectionStatus(current);  } } |

После того как статус подключения изменится на CONNECTED, в ранее зарегистрированном экземпляре класса DataListener будет вызыван метод receiveMuseDataPacket. В этом методе, в зависимости от типа пришедшего пакета с данными от гарнитуры, будет вызван соответствующий метод класса MuseActivity: processMuseDataRelative для пакета типа ALPHA\_RELATIVE или BETA\_RELATIVE, processMuseDataBattery для пакета типа BATTERY, processMuseDataSensors для пакета типа IS\_GOOD.

Листинг 3.3. Методы класса MuseActivity для обработки данных от гарнитуры Muse

|  |
| --- |
| public void processMuseDataSensors(ArrayList<Double> packetValues) {  for (int i = 0; i < CHANNEL\_COUNT; i++) {  sensorsStateBuffer[i] = packetValues.get(i) > 0.5;  }  sensorsStale = true;  }  public void processMuseDataBattery(MuseDataPacket p) {  batteryValue = p.getBatteryValue(Battery.CHARGE\_PERCENTAGE\_REMAINING);  batteryStale = true;  }  public void processMuseDataRelative(ArrayList<Double> packetValues, int relativeIndex) {  fillRelativeBufferWith(relativeIndex, packetValues);  relativeStale = true;  } |

## Работа с администрирующим устройством

Для работы по каналу Bluetooth был написан класс BluetoothService, берущий на себя всю работу по управлению подключением к устройству и передаче пакетов данных. Класс берет на себя как роль сервера, так и роль клиента в подключении по каналу Bluetooth.

Для использования класса вне зависимости от роли необходимо инициализировать экземпляр класса с помощью вызова конструктора, имеющего следующую сигнатуру:

Листинг 3.4. Сигнатура конструктора класса BluetoothService

|  |
| --- |
| public BluetoothService(Handler handler, BluetoothAdapter btAdapter); |

Ссылка на класс типа Handler требуется для обеспечения механизма обратного вызова при изменении состояния соединения, либо обработки полученного пакета. Ссылка на BluetothAdapter требуется для непосредственно работы с аппаратной частью Bluetooh-компонента устройства.

Для использования инициализированного экземпляра класса BluetoothService в роли сервера, вызывается метод startServer(), внутри которого производится запуск потока (вложенный класс AcceptThread) для прослушивания запросов на подключение. Работа потока заключается в вызове блокирующего метода accept для получения экземпляра сокета подключенного устройства или обработке ошибок, возникших при этом. При успешном подключении и получении экземпляра сокета, производится запуск потока для принятия сообщений от устройства (вложенный класс ConnectedThread).

Листинг 3.5. Сигнатура метода start класса BluetoothService

|  |
| --- |
| public synchronized void startServer(); |

Класс ConnectedThread получает из сокета ссылки на входной и выходной потоки для связи с устройством, а затем, пока состояние подключения позволяет, считывает из входного потока данные и реализует механизм обратного вызова с помощью ссылки на экземпляр класса Handler, переданного ранее в конструкторе класса BluetoothService. Кроме того класс ConnectedThread предоставляет метод write для записи данных в выходной поток сокета подключенного устройства.

Для использования инициализированного экземпляра класса BluetoothService в роли клиента необходимо вызвать метод connectToServer, имеющий следующую сигнатуру:

Листинг 3.6. Сигнатура метода connectToServer

|  |
| --- |
| public synchronized void connectToServer(BluetoothDevice device); |

Ссылка на устройство для установления подключения затем передается в конструктор вложенного класса ConnectThread, представляющего собой поток для установления подключения. При успешном установлении подключения, далее инициализируется класс экземпляр класса ConnectedThread. Работа этого класса описана выше и аналогична случаю с выполнением экземпляра класса BluetoothService роли сервера: пакеты данных считываются через входной поток, полученный из сокета подключенного устройства, при успешном получении информации из потока, происходит уведомление об этом с помощью переданной ранее в конструкторе класса BluetoothService ссылки на экземпляр класса Handler.



Рисунок 3.3. Схема работы класса BluetoothService

## Работа с базой данных

Библиотека ORMLite предполагает активное использование аннотации при объявлении классов, хранимых в базе данных. Через аннотации можно задать и тип данных поля, что очень удобно и не размазывает код, связанный с моделью по проекту. Проект поддерживает множество типов данных и вариантов их хранения. Например, для java.util.Date предусмотрен как числовой, так и строковый вариант. К недостаткам можно отнести необходимость реализовывать OrmLiteSqliteOpenHelper, через который вы сможете получить DAO объект и взаимодействовать с ORM. Использование отдельных DAO объектов избавляет от необходимости наследовать классы ваших сущностей от объектов сторонних библиотек и позволяет гибко управлять кэшем.

Благодаря использованию библиотеки ORMlite, работа с базой данных сильно упрощается. Для интеграции библиотеки в систему необходимо аннотировать классы, которые будут храниться в базе данных. Каждый класс будет представлять собой таблицу, строки в которой хранят сериализованную информацию о данных конкретного экземпляра класса.

Аннотация DatabaseTable используется для указания на то, что класс будет храниться в БД, а также позволяет задать имя таблицы. Чтобы указать, какие поля класса требуется хранить, используется аннотация DatabaseField. Для данной аннотации предусмотрен ряд параметров, которые позволяют более гибко сконфигурировать хранимые поля. В разрабатываемо системе используются следующие параметры:

* generatedId

булевское значение для спецификации поля как автоматически генерируемого идентификатора. Значение по умолчанию – false. Только одно поле в классе может иметь значение true. Этот параметр сообщает базе данных, что необходимо автоматически сгенерировать значение для данного поля, когда происходит операция вставки.

* canBeNull

булевское значение для спецификации возможности записи значения NULL. По умолчанию true.

* foreign

булевское значение для спецификации поля как внешнего ключа. По умолчанию is false. Поле не должно быть примитивного типа. Тип поля должен иметь идентификатор, который хранится в данной таблице.

* foreignAutoRefresh

булевское значение для автоматического получения всего связанного объекта, а не только идентификатора.

Приведем пример использования аннотаций на примере класса User. Класс помечены аннотацией DatabaseTable с указанием имени таблицы «users». Поле id, выполняющее роль идентификатора и первичного ключа для таблицы, аннотировано как DatabaseField(generatedId = true), остальные строковые поля, такие как firstName, lastName, sex, birthDateStr аннотированы как DatabaseField(canBeNull = false), так как предполагается их наличие в любом хранимом экземпляре класса User. Поле seances помечено как ForeignCollectionField, что позволит извлечь связанные данные, относящиеся к конкретной записи.

Листинг 3.7. Класс User

|  |
| --- |
| @DatabaseTable(tableName = "users") public class User {   @DatabaseField(generatedId = true)  public int id;   @DatabaseField(canBeNull = false)  public String firstName;   @DatabaseField(canBeNull = false)  public String lastName;   @DatabaseField(canBeNull = false)  public String sex;   @DatabaseField(canBeNull = false)  public String birthDateStr;   @ForeignCollectionField()  public ForeignCollection<Seance> seances; } |

Когда все классы аннотированы, необходимо описать класс VideoMoodDbHelper, который будет контролировать создание самой базы данных. Он является наследником класса OrmLiteSqliteOpenHelper. В нем переопределен метод onCreate, который вызывается для БД каждый раз при создании. В методе необходимо явно создать все требуемые таблицы для всех требуемых классов с помощью статического метода createTable класса TableUtils из библиотеки ORMlite. Все типы, которые будут созданы, хранятся в виде массива значений в поле entityClasses.

Листинг 3.8. Метод onCreate класса VideoMoodDbHelper

|  |
| --- |
| private static final Class[] entityClasses = new Class[]{  AgeCategory.class,  Tag.class,  User.class,  Video.class,  VideoTag.class,  VideoAgeCategory.class,  Seance.class,  SeanceVideo.class };  @Override public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {  try {  for (int i = 0; i < entityClasses.length; i++) {  Class entityClass = entityClasses[i];  TableUtils.createTable(connectionSource, entityClass);  }  } catch (SQLException e) {  Log.e(VideoMoodDbHelper.class.getName(), "Can't create database", e);  throw new RuntimeException(e);  } } |

В классе присутствует поле DATABASE\_NAME для указания имени файла хранимой базы данных, а также поле DATABASE\_VERSION для указания текущей версии схемы базы данных. Версия схемы базы данных требуется в случае, если схема был изменена (добавлялись или удалялись новые таблицы, менялись связи между имеющимися таблицами, менялся состав хранимых полей какой-либо таблицы). В таком случае различие версий будет выявлено, и с помощью переопределения метода onUpgrade будут выполнены все требуемые операции по обновлению схемы.

Описанный класс VideoMoobDbHelper затем будет использоваться при объявлении любого класса активности, работающего с базой данных. Такой класс активности будет наследоваться от обобщенного класса OrmLiteBaseActivity<THelper>. В качестве типа THelper будет указан VideoMoodDbHelper. Это позволит активности точно знать, с какой базой данных она работает, и предоставлять объект этого типа при вызове метода getHelper.

Пример инициализации DAO-объекта приведен в листинге ниже. В методе onCreate вызывается метод getHelper класса OrmLiteBaseActivity, возвращающий ссылку на класс VideoMoodDbHelper. Вызвав затем метод getUserDao определенный в классе VideoMoodDbHelper, получим DAO-объект для управления записями типа User в базе данных.

Листинг 3.9. Пример инициализации DAO-объекта в классе UserActivity

|  |
| --- |
| public class UsersActivity extends OrmLiteBaseActivity<VideoMoodDbHelper> {  private Dao<User, Integer> userDao;  @Override  protected void onCreate(@Nullable Bundle savedInstanceState) {  super.onCreate(savedInstanceState);   try {  userDao = getHelper().getUserDao();  } catch (SQLException e) {  e.printStackTrace();  }  ...  } |

Имея экземпляр класса-наследника OrmLiteSqliteOpenHelper, можем получить DAO (Data Access Object) для конкретного типа, хранимого в базе данных. Например обобщенный тип Dao<TEntity, TKey> выглядит как Dao<User, Integer> для случая, когда необходимо получить объекты типа User, идентификатором которых является поле с целочисленным типом. DAO-объекты позволяют производить CRUD-операции (Create, Read, Update, Delete) с объектами указанного типа.

Например, класс UsersActivity использует DAO-объект типа Dao<User, Integer>, ссылка на который сохранена в поле usersDao, следующим образом:

* userDao.create(userToCreate) – для создания нового пользователя в БД;
* userDao.queryForAll() – для получения всех существующих пользователей из БД;
* userDao.remove(userToRemove) – для удаления указанного пользователя из БД.

Пример использования объекта userDao приведен в листинге ниже.

Листинг 3.10. Метод confirmCreateUser

|  |
| --- |
| private void confirmCreateUser() {  User userToCreate = new User();  ...  try {  if (userDao.create(userToCreate) == 0)  throw new Exception("No user created");  if (userToCreate.id != -1)  userAdapter.add(userToCreate);  else  throw new Exception("No user created");  } catch (Exception e) {  e.printStackTrace();  }  } |

## Алгоритм определения состояния

Все вычисления, необходимые для определения значения, производятся в методах класса MoodResolver. Его реализация приведена ниже. Класс содержит метод boxCountingDimension, который для заданного массива значений в цикле для всего диапазона значений размеров, наносит сетку, маркирует ячейки этой сетки, в которых находятся значения массива, а также вычисляет значения, по которым считается предел. В результате метод вернет хэш-таблицу значений логарифма обратного квадрата площади ячейки сетки на значения количества занятых ячеек.

Метод normalEquations2d берет значения хэш-таблицы, интерпретируя их как координаты точек и решает задачу линейной регрессии посредством использования метода наименьших квадратов. В результате метод вернет вектор из двух значений: значение коэффициента прямой линейной регрессии и смещение. Первое значение и является искомым значением размерности Минковского.

Листинг 3.11. Класс MoodResolver

|  |
| --- |
| public class MoodResolver {   public static int minY = 0;  public static int maxY = 1700;   public static double[] normalEquations2d(double[] y, double[] x) {  // x^t \* x  double[][] xtx = new double[2][2];  for (int i = 0; (i < x.length); i++) {  xtx[0][1] = (xtx[0][ 1] + x[i]);  xtx[0][0] = (xtx[0][ 0] + (x[i] \* x[i]));  }   xtx[1][0] = xtx[0][1];  xtx[1][1] = x.length;   // inverse  double[][] xtxInv = new double[2][2];  double d = (1 / ((xtx[0][0] \* xtx[1][1]) - (xtx[1][0] \* xtx[0][1])));  xtxInv[0][0] = (xtx[1][1] \* d);  xtxInv[0][1] = ((xtx[0][1] \* d) \* -1);  xtxInv[1][0] = ((xtx[1][0] \* d) \* -1);  xtxInv[1][1] = (xtx[0][0] \* d);   // times x^t  double[][] xtxInvxt = new double[2][x.length];  for (int i = 0; (i < 2); i++) {  for (int j = 0; (j < x.length); j++) {  xtxInvxt[i][j] = ((xtxInv[i][0] \* x[j]) + xtxInv[i][1]);  }  }   // times y  double[] theta = new double[2];  for (int i = 0; (i < 2); i++) {  for (int j = 0; (j < x.length); j++) {  theta[i] = (theta[i] + (xtxInvxt[i][j] \* y[j]));  }  }   return theta;  }   private static boolean isInRange(float value, int left, int right) {  return value > left && value <= right;  }   /\*\*  \* Box-counting algorithm  \* @param plot - timeline of values  \* @param startSize - initial size of square of grid  \* @param finishSize - final size of square of grid  \* @param step - step of changing of the grid  \* @return map Math.Log(1/b) to Math.Log(a) where b is square length size, a is the number of intersection of image with grid squares  \*/  public static HashMap<Double, Double> boxCountingDimension(float[] plot, int startSize, int finishSize, int step)  {  //length size - number of boxes  HashMap<Double, Double> baList = new HashMap<>();   int bwHeight = maxY - minY;   for (int boxSize = startSize; boxSize <= finishSize; boxSize += step)  {  boolean[][] filledBoxes = fillBoxes(plot, boxSize, bwHeight);   int a = 0;  for (int i = 0; i < filledBoxes.length; i++)  for (int j = 0; j < filledBoxes[0].length; j++)  if (filledBoxes[i][j])  a++;   baList.put(Math.log(1d/boxSize), Math.log(a));  }   return baList;  }   /\*\*  \* create array of boxes depend on timeline plot size and box size, mark boxes where values of plot are presented  \* @param plot - array of timeline plot values  \* @param boxSize - size of box to divide plot  \* @param plotHeight - plot height  \* @return array of marked boxes  \*/  private static boolean[][] fillBoxes(float[] plot, int boxSize, int plotHeight) {  int plotWidth = plot.length;   int hCount = plotHeight/boxSize;  int wCount = plotWidth/boxSize;   if (plotWidth > wCount\*boxSize)  wCount += 1;  if (plotHeight > hCount\*boxSize)  hCount += 1;   boolean[][] filledBoxes = new boolean[wCount][hCount];  for (int i = 0; i < plot.length; i++) {  float value = plot[i];  int yBox = (int) (value/boxSize);  int left = yBox\*boxSize;  int right = left + boxSize;  if (isInRange(value, left, right)) {  int xBox = i/boxSize;  filledBoxes[xBox][yBox] = true;  }  }  return filledBoxes;  }   public static double[] getThetaValues(float[] curve, int startSize, int endSize, int step) {  HashMap<Double, Double> baList = boxCountingDimension(curve, startSize, endSize, step);  double[] y = new double[baList.size()];  double[] x = new double[baList.size()];  int c = 0;  for (double key : baList.keySet())  {  y[c] = baList.get(key);  x[c] = key;  c++;  }  double[] theta = normalEquations2d(y, x);  return theta;  } } |

# ТЕСТИРОВАНИЕ

В данной работе под тестированием подразумевается проверка реализации требований, поставленных в разделе 2.1.

## 4.1 Тестирование реализованного алгоритма работы

Согласно заданию, основная программа должна автоматически переключаться между двумя режимами работы: «спокойствие» и «тревога». Переключение происходит на основании информации от модуля определения психоэмоционального состояния.

При включении программа автоматически осуществляет поиск доступных для подключения гарнитур Muse (Рисунок 4.1). В случае если гарнитура одна, к ней осуществляется подключение. Одновременно с этим начинается воспроизведение случайного видео файла (Рисунок 4.2).

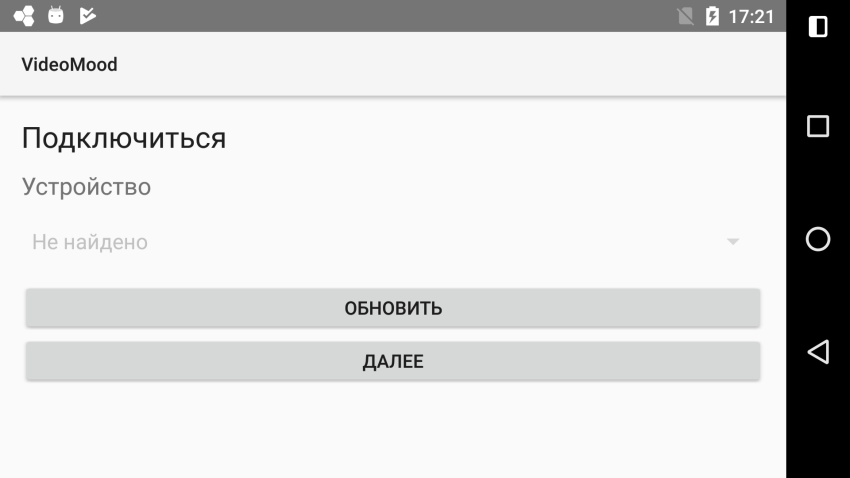


Рисунок 4.1. Интерфейс программы при попытке подключения к гарнитуре Muse

При успешном подключении к гарнитуре Muse, начинается анализ сигнала ЭЭГ, а вместе с ним и отслеживание психоэмоционального состояния (Рисунок 4.3). При включении программа находится в режиме «спокойствие».

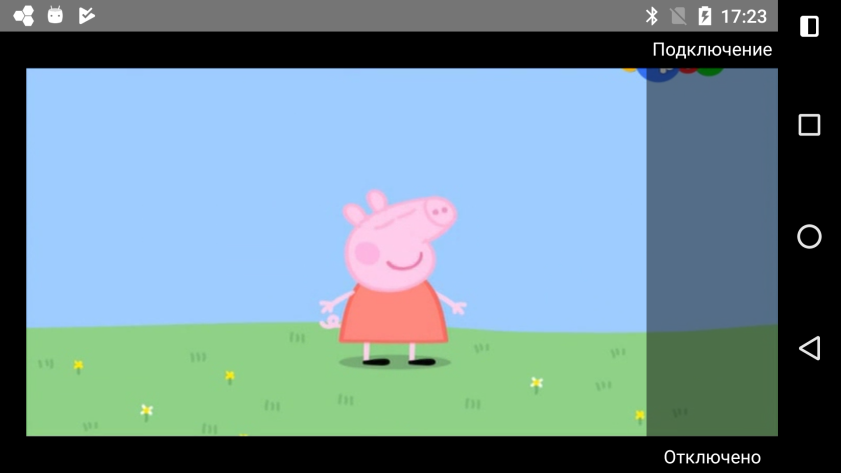


Рисунок 4.2. Интерфейс программы до подключения к гарнитуре Muse

Далее раз в секунду происходит анализ переменной, отвечающей за текущее состояние. При ее изменении в значение «тревога», происходит переключение графического интерфейса в соответствующий режим (Рисунок 4.4).



Рисунок 4.3. Интерфейс программы при подключенной гарнитуре Muse  
 и режиме "спокойствие"

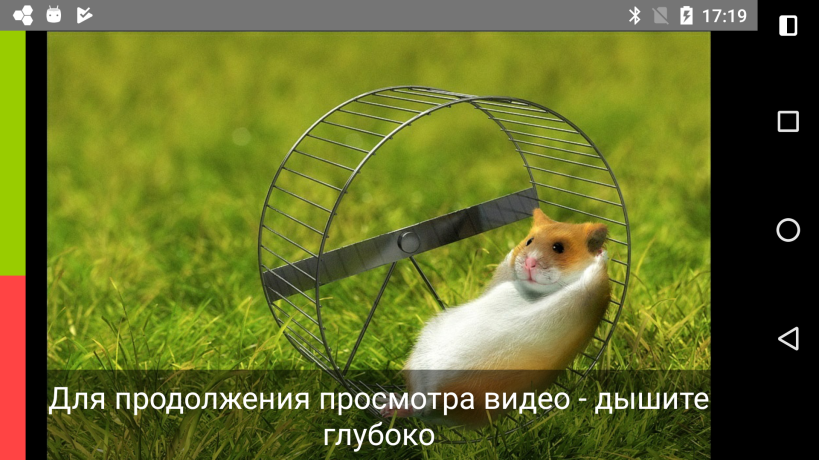


Рисунок 4.4. Интерфейс программы в режиме тревоги

На рисунках выше можно наблюдать, что требования из технического задания по отображаемой информации выполнены. В режиме «спокойствие» присутствуют:

* видео контент;
* значение характеристики, по которой определяется психоэмоциональное состояние (полоски красного и зеленого цвета слева экрана);
* состояние подключения гарнитуры MUSE в виде надписи «ПОДКЛЮЧЕНО» или «ОТКЛЮЧЕНО» (в правом верхнем углу экрана);
* уровень заряда батареи гарнитуры MUSE в виде процентов от 1 до 100 (в правой части экрана);
* качество сигнала с каждого из 4-ех датчиков в виде разноцветных кругов, скрываемых или отображаемых в зависимости от качества сигнала с датчиков (в правой части экрана);
* состояние подключения администрирующего устройства в виде надписи «ПОДКЛЮЧЕНО» или «ОТКЛЮЧЕНО» (в правом нижнем углу экрана).

Кроме того, выполнены условия из технического задания по отображению информации в режиме «тревога». На экране отображаются:

* значение характеристики, по которой определяется психоэмоциональное состояние (полоски красного и зеленого цвета слева экрана);
* заранее выбранное художественное изображение в качестве фона;
* надпись с советом по снижению уровня тревожности (в нижней части экрана).

## 4.2 Тестирование взаимодействия основной и администрирующей программы

При запуске администрирующая программа выводит экран, на котором содержится список имеющихся в системе учетных записей (Рисунок 4.5), а также имеется возможность создать новую (Рисунок 4.6) или удалить имеющуюся (Рисунок 4.7).

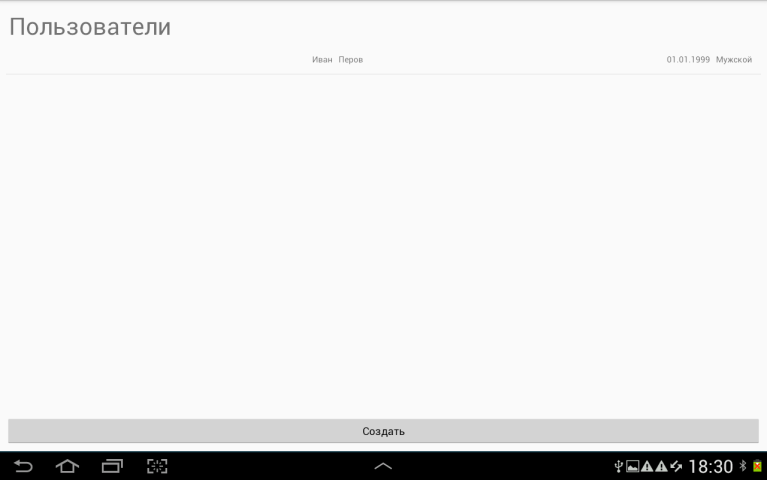


Рисунок 4.5. Интерфейс программы при отображении списка учетных записей

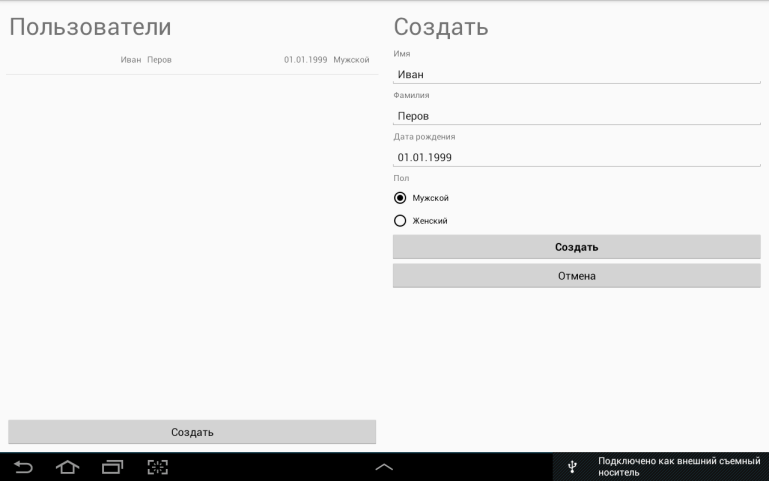


Рисунок 4.6. Интерфейс программы при создании новой учетной записи

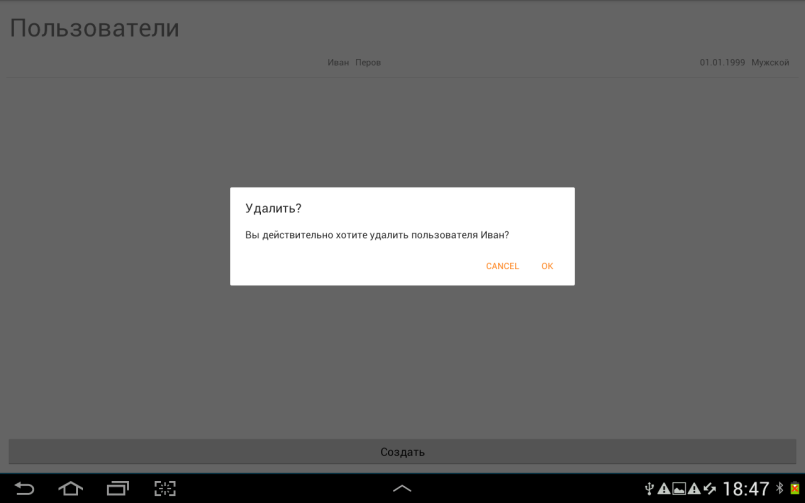


Рисунок 4.7. Интерфейс программы при удалении учетной записи

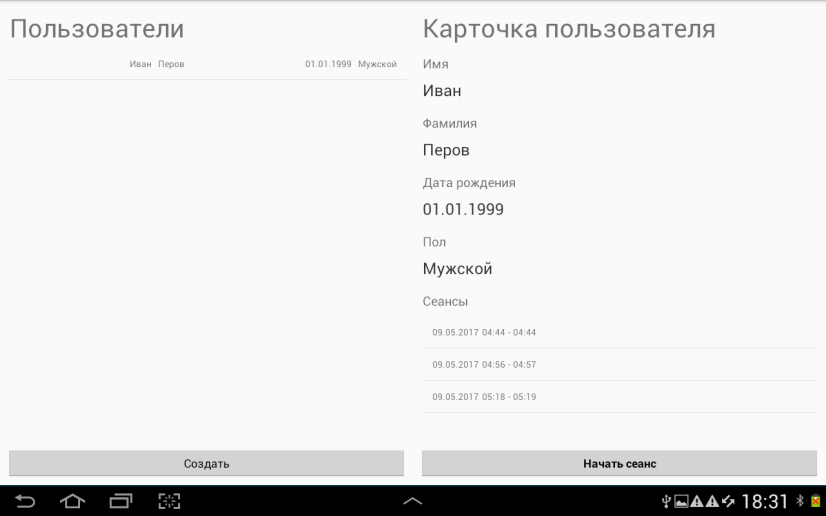


Рисунок 4.8. Интерфейс программы при отображении информации об учетной записи

После выбора учетной записи, для которой будет проводиться сеанс (Рисунок 4.8), появляется экран выбора устройства для подключения (Рисунок 4.9).

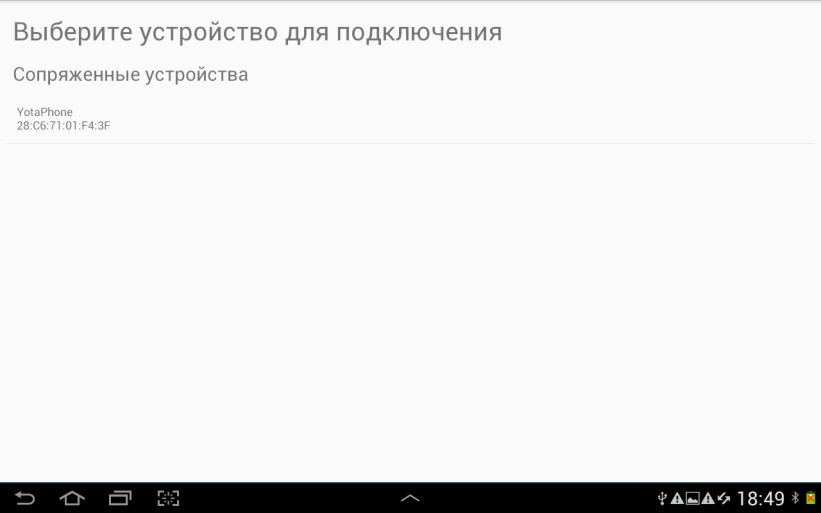


Рисунок 4.9. Интерфейс программы при выборе устройства   
для подключения к основной программе

После выбора устройства для подключения открывается основной экран администрирующего приложения, на котором будет содержаться вся информация о сеансе (Рисунок 4.10).

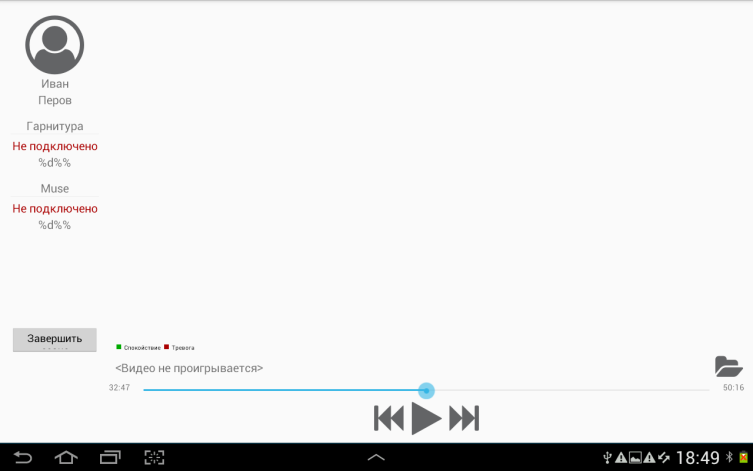


Рисунок 4.10. Интерфейс программы до подключения к основной программе

При успешном подключении к основной программе, статус подключения меняется на «ПОДКЛЮЧЕНО» (верхний левый угол, под именем и фамилией пользователя, под надписью «ГАРНИТУРА»). Там же отображается процент заряда батареи устройства, на котором выполняется основная программа. Ниже отображено состояние подключения гарнитуры Muse (под надписью «MUSE»), а также процент заряда батареи гарнитуры Muse. Состояние подключения дублируется цветом для ускорения считывания информации.

В левом верхнем углу над именем и фамилией пользователя выводится изображение улыбающегося или печального лица для обозначения текущего психоэмоционального состояния пользователя: «спокойствие» или «тревога» соответственно.

В нижней части экрана присутствуют кнопки управления воспроизведением видео файлов (предыдущий, проиграть/остановить/ следующий), ползунок для отображения процесса проигрывания текущего видео файла, значения текущего положения проигрывания (слева от ползунка) и общей длительности файла (справа от ползунка). Кроме того отображается название проигрываемого видео файла и кнопка для выбора другого файла.

Большую часть экрана занимает график со столбцами значений характеристики, по которой происходит определение психоэмоционального состояния. Столбцы окрашиваются в зеленый или красный цвет в зависимости от текущего психоэмоционального состояния (Рисунок 4.12).

По окончании сеанса становится доступна страница с информацией о проведенном сеансе (Рисунок 4.13), в которой отображается вся информация, указанная в соответствующем пункте требований.

В результате тестирования было установлено, что требования к функциональности программ выполнены.

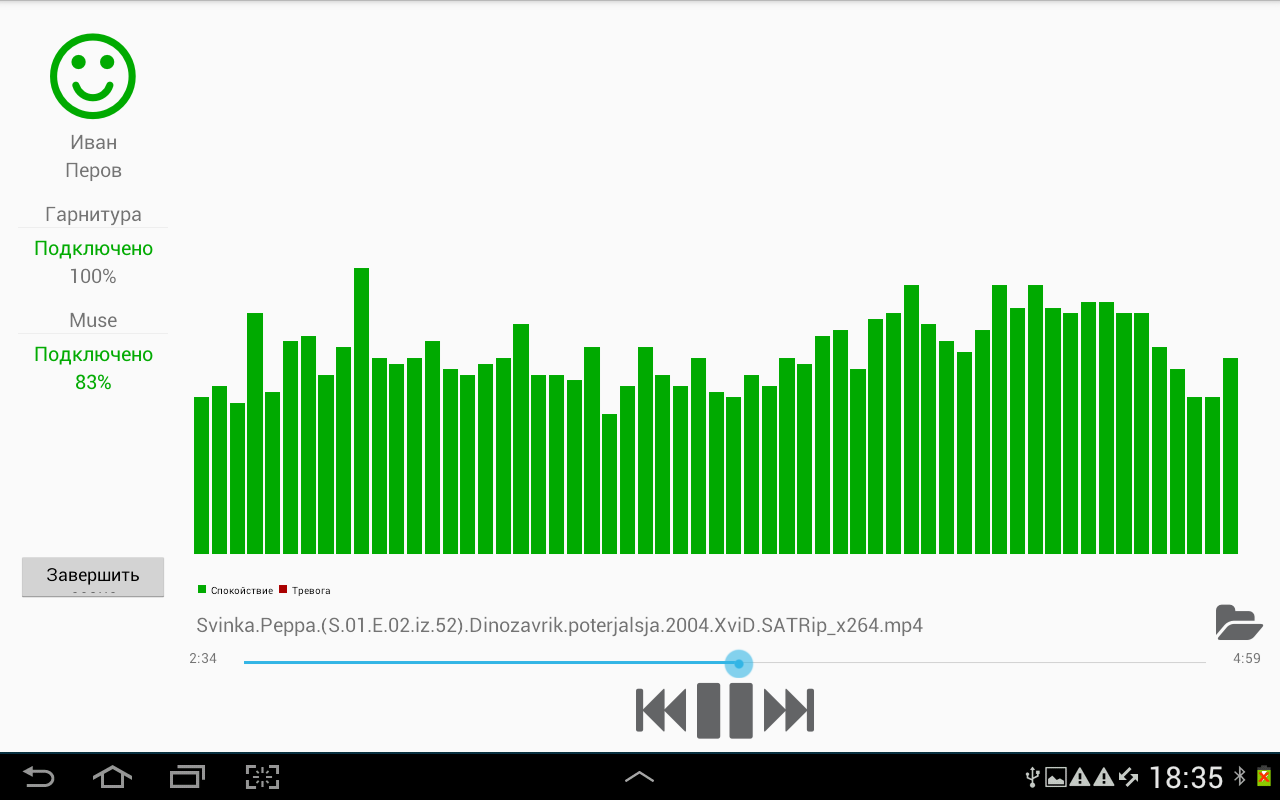


Рисунок 4.11. Интерфейс программы при проведении сеанса

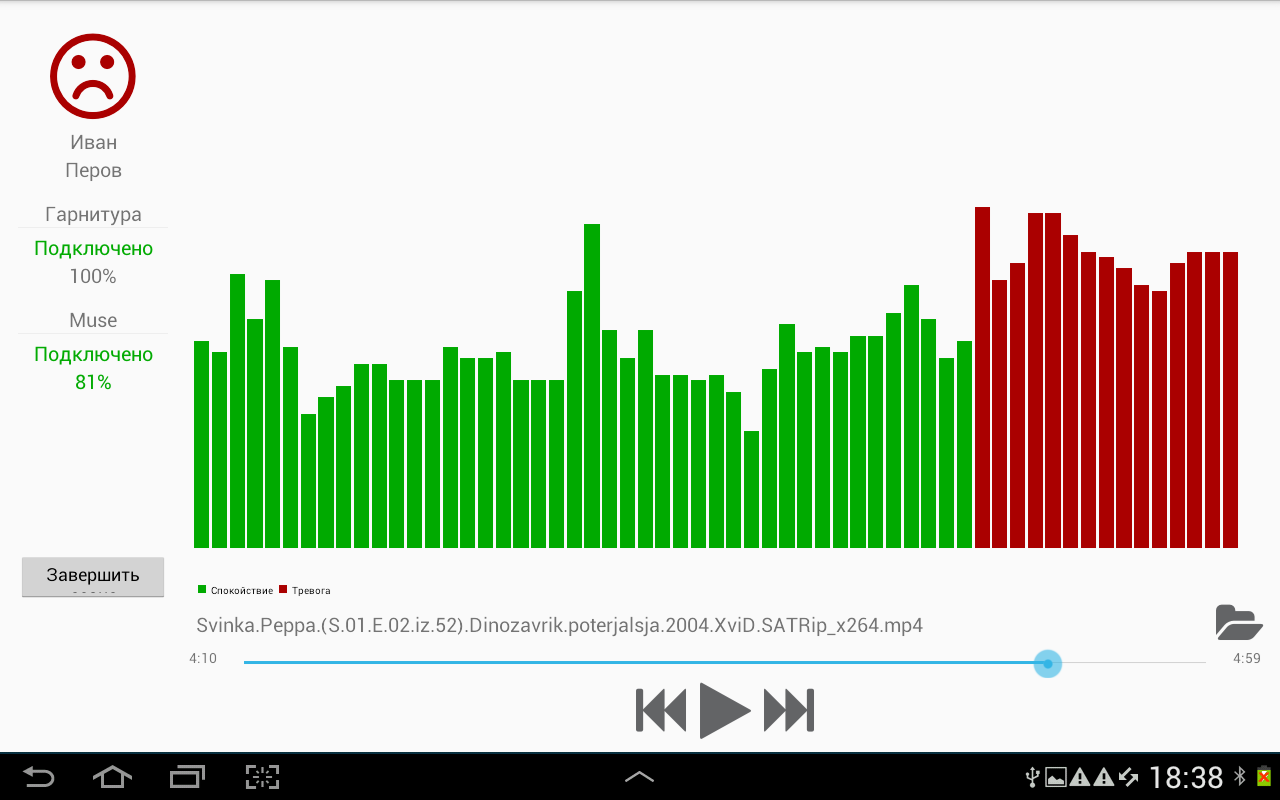


Рисунок 4.12. Интерфейс программы при переключении основной программы   
в режим тревоги

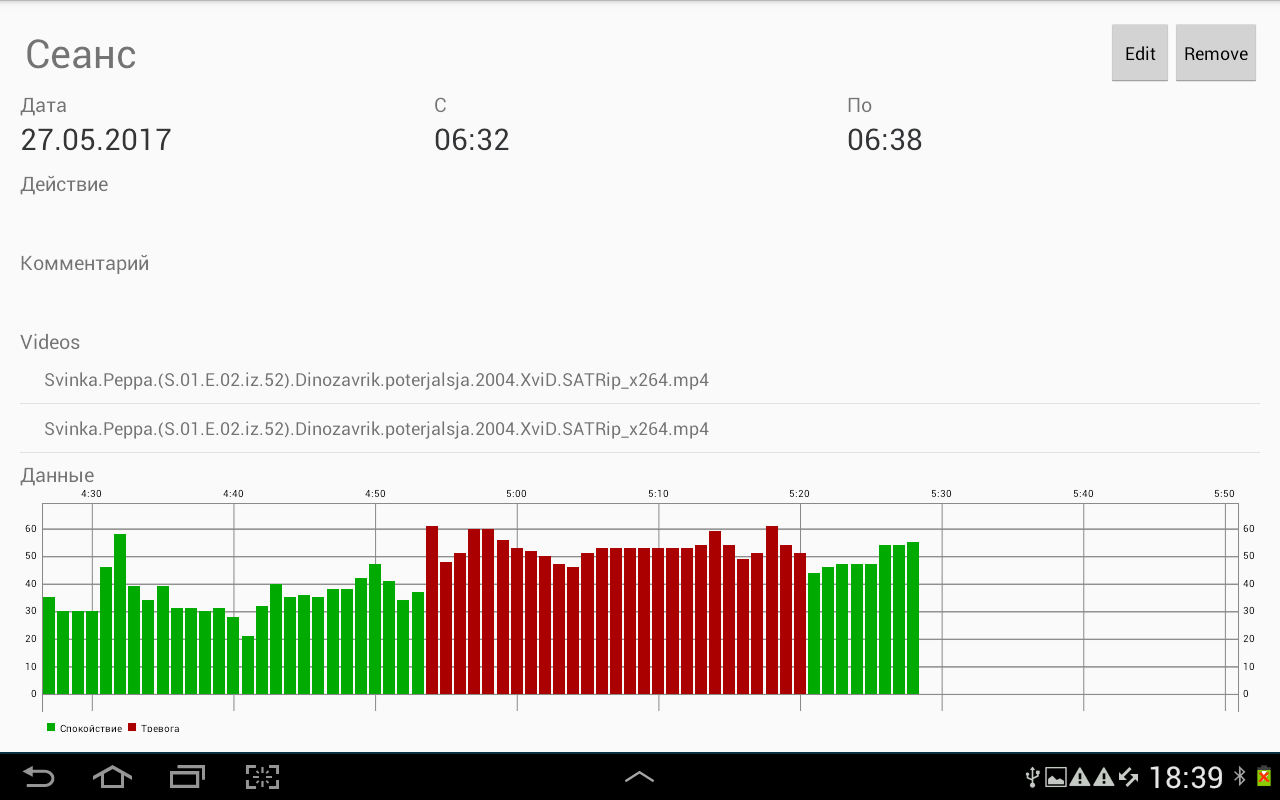


Рисунок 4.13. Интерфейс программы при отображении информации   
о завершенном сеансе

# ЗАКЛЮЧЕНИЕ

В ходе диссертационного исследования был выполнен следующий объем работы: обзор формальных моделей эмоций, обзор алгоритмов для определения эмоций. Также был произведен обзор существующих решений для определения психоэмоционального состояния по данным ЭЭГ. Далее была поставлена цель по реализации системы для этого. Для достижения поставленной цели были сформулированы задачи и требования, предъявляемые к системе. Согласно поставленным требованиям была сконфигурирована аппаратная платформа. Среди алгоритмов определения эмоций был выбран алгоритм с использованием фрактальной размерности.

Следующим этапом стало проектирование программной части системы. Были выделены функциональные компоненты и связи по данным между ними. Кроме того были выбраны средства разработки программной части, исходя из выбранной аппаратной платформы (операционной системы Android). Далее основные функциональные компоненты были спроектированы более детально, с учетом требований к функциональности системы. Была представлена детальная схема алгоритма вычисления фрактальной размерности.

Разработка производилась параллельно во всех компонентах, постепенно реализовывалась требуемая функциональность. В результате была создана система классов, абстрактно разделяемая на функциональные компоненты, согласно спроектированной ранее схеме. Исходя из требований, предъявленных к системе ранее, было проведено тестирование функциональности разработанного программного обеспечения. По результатам тестирования было установлено, что программное обеспечение выполняет свои функции согласно техническому заданию.

Кроме тестирования программного обеспечения согласно техническому заданию, планируется написание автоматизированных модульных тестов для функциональных компонентов системы. Такой подход позволит в будущем развивать части программного обеспечения в функциональном плане и в плане производительности, будучи при этом уверенным в правильной работе уже имеющихся частей. Также предстоит разработать тестирующую систему для алгоритма определения эмоционального состояния.

Целью дальнейшего исследования и разработки также является совершенствование алгоритма определения психоэмоционального состояния. Планируется внедрение дополнительных характеристик кривой, внедрение классификаторов и алгоритмов машинного обучения для улучшения качества определения психоэмоционального состояния.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статья всемирной организации здравоохранения о депрессии.   
   URL: <http://www.who.int/mediacentre/factsheets/fs369/ru/>
2. Posner, Jonathan, James A. Russell, and Bradley S. Peterson. “The Circumplex Model of Affect: An Integrative Approach to Affective Neuroscience, Cognitive Development, and Psychopathology.” Development and psychopathology 17.3 (2005): 715–734. PMC. Web. 1 June 2017.
3. Toward a consensual structure of mood. Watson, David; Tellegen, Auke Psychological Bulletin, Vol 98(2), Sep 1985, 219-235.
4. David C. Rubin, Jennifer M. Talarico «A Comparison of Dimensional Models of Emotion: Evidence from Emotions, Prototypical Events, Autobiographical Memories, and Words», 2009,   
   URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2784275/>
5. Экспериментальная психология. / Ред.-сост. П. Фресс, Ж. Пиаже. М.: Прогресс, 1975. С.133–142
6. Сайт библиотеки для модульного тестирования ПО на языке Java – JUnit. URL: <https://github.com/junit-team/junit4/wiki/Getting-started>, на 19.04.2017 г.
7. Статья «Линейный дискриминантный анализ». URL: <http://www.machinelearning.ru/wiki/index.php?title=Линейный_дискриминантный_анализ>
8. Kwang-Eun Ko, Hyun-Chang Yang, and Kwee-Bo Sim, «Emotion Recognition using EEG Signals with Relative Power Values and

Bayesian Network», International Journal of Control, Automation, and Systems (2009).

1. Yisi Liu, Olga Sourina, and Minh Khoa Nguyen, «Real-time EEG-based Human Emotion Recognition and Visualization», International Conference on Cyberworlds (2010).
2. Alexanderson Albers , Alexanderson Gerald L. Benoit Mandelbrot: In his own words // Mathematical people : profiles and interviews. — Wellesley, Mass: AK Peters, 2008. — 214 p.
3. A. Accardo, M. Affinito, M. Carrozzi, and F. Bouquet, «Use of the fractal dimension for the analysis of electroencephalographic time series» Biological Cybernetics, vol. 77, 1997, pp. 339-350.
4. N. Pradhan and D. Narayana Dutt, «Use of running fractal dimension for the analysis of changing patterns in electroencephalograms» Computers in Biology and Medicine, vol. 23, 1993, pp. 381-388
5. W. Lutzenberger, T. Elbert, N. Birbaumer, W. J. Ray, and H. Schupp, «The scalp distribution of the fractal dimension of the EEG and its variation with mental tasks» Brain Topography, vol. 5, 1992, pp. 27-34
6. Статья «Биологическая обратная связь» на сайте СПб НИПНИ им. В.М. Бехтерева. URL: <http://bekhterev.ru/clinika/fizioterapiya/bos/index.php>
7. Сайт библиотеки ORMlite URL: <http://ormlite.com/>

# ПРИЛОЖЕНИЕ 1

Программное обеспечение системы

для определения психоэмоционального состояния по данным ЭЭГ

с применением биологической обратной связи

**СПЕЦИФИКАЦИЯ**

(Р.П.35126-01)

Листов 1

Санкт-Петербург

2017

|  |  |  |
| --- | --- | --- |
| **Обозначение** | **Наименование** | **Примечание** |
|  | Документация |  |
| Р.П.35126-01 | Техническое задание на разработку | Приложение 4 |
| Р.П.35126-01.12.01 | Тексты программы | Приложение 7 |
| Р.П.35126-01.13.01 | Описание программы | Приложение 5 |
| Р.П.35126-01.51.01 | Программа и методика испытаний | Приложение 6 |
|  | Входящие программы |  |
| VideoMood | VideoMood | Программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи |

# ПРИЛОЖЕНИЕ 2

Программное обеспечение системы

для определения психоэмоционального состояния по данным ЭЭГ

с применением биологической обратной связи

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

(Р.П.35126-01)

Листов 9

Санкт-Петербург

2017

АННОТАЦИЯ

В данном документе представлено техническое задание на программный продукт «Программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи». Приведены основные требования к программному изделию и программной документации, а также описаны основные стадии и этапы разработки.

1. **Введение**
   1. **Наименование программы**

Наименование – «Программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи».

* 1. **Краткая характеристика области применения**

Программа предназначена к применению в стоматологических клиниках для снижения уровня тревожности при выполнении медицинских операций.

1. **Основания для разработки**
   1. **Основание для проведения разработки**

Основанием для проведения разработки является задание на дипломное проектирование, утвержденное 04 февраля 2017г. на кафедре Компьютерных Систем и Программных Технологий СПбПУ (далее – Заказчик). Задание на дипломное проектирование согласовано с руководителем дипломного проекта М.В. Болсуновской и принято к исполнению 04 февраля 2017г.

* 1. **Наименование темы разработки**

Наименование темы разработки – «Разработка системы определения психоэмоционального состояния пользователя на основе данных ЭЭГ с использованием биологической обратной связи».

1. **Назначение разработки**
   1. **Функциональное назначение**

Функциональным назначением программного обеспечения является определение психоэмоционального состояния по данным ЭЭГ с использованием биологической обратной связи.

* 1. **Эксплуатационное назначение**

Программа предполагается к использованию в стоматологических клиниках для обучению пользователей контролю за психоэмоциональным состоянием посредством биологической обратной связи.

1. **Требования к программному модулю**
   1. **Требования к функциональным характеристикам**
      1. **Требования к составу выполняемых функций**

Программное обеспечение должно обеспечивать возможность выполнения перечисленных ниже функций:

* Программное соединение компонентов системы (устройств на ОС Android) по каналу связи Bluetooth;
* Воспроизведение видео файлов с файловой системы основного устройства на ОС Android;
* Обработка пакетов данных о сигнале ЭЭГ, поступающем от гарнитуры MUSE;
* Определение психоэмоционального состояния на основании данных ЭЭГ;
* Вывод на экран основного устройства индикатора психоэмоционального состояния;
* Отслеживание изменения психоэмоционального состояния и отображение советов по снижению тревоги при возникновении соответствующего психоэмоционального состояния;
* Обмен данными между основным и администрирующим устройством по каналу Bluetooth;
* Отображение данных, полученных с основного устройства на экране администрирующего устройства;
* Обработка и выполнение команд от администрирующего устройства основным устройством:
  + Приостановка/воспроизведение текущего видео файла,
  + Переключение на следующий/предыдущий видео файл,
  + Получение списка доступных видео файлов,
  + Воспроизведение выбранного из списка доступных видео файлов,
* Хранение данных об учетных записях пользователей и проводимых сеансах
  + 1. **Требования к организации входных данных**

Входными данными являются видео файлы с файловой системы основного устройства, а также пакеты данных сигнала ЭЭГ от гарнитуры MUSE.

* + 1. **Требования к организации выходных данных**

Выходными данными являются графический интерфейс с отображаемыми данными о сеансе, либо видео файл с индикатором психоэмоционального состояния. Кроме того выходными данными может считаться база данных с учетными записями пользователей и содержащимися в ней данными о сеансах использования системы.

* + 1. **Требования к временным характеристикам**

Требования к временным характеристикам программы не предъявляются.

* 1. **Требования к надежности**
     1. **Требования к обеспечению надежного (устойчивого) функционирования программы**

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. организацией бесперебойного питания технических средств;
2. регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
3. регулярным выполнением требований ГОСТ 51188 – 98 «Испытания программных средств на наличие компьютерных вирусов».
   * 1. **Время восстановления после отказа**

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем операционной системы, не должно превышать пятнадцати минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

* + 1. **Отказы из-за некорректных действий пользователя**

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой.

* 1. **Условия эксплуатации**
     1. **Климатические условия эксплуатации**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

* + 1. **Требования к видам обслуживания**

См. требования к обеспечению надежного (устойчивого) функционирования программы.

* + 1. **Требования к численности и квалификации персонала**

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – оператор системы, при задействовании администрирующего устройства, и конечный пользователь программы.

Оператор системы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы Android, а именно: включением устройства, включением передачи сигнала по каналу Bluetooth, запуском программ, управлением файлами на файловой системе устройства.

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы Android.

* 1. **Требования к составу и параметрам технических средств**

Состав технических средств и их параметры определяются аппаратной частью системы.

* 1. **Требования к информационной и программной совместимости**
     1. **Требования к исходным кодам и языкам программирования**

Исходные коды программы должны быть реализованы на языке Java. В качестве интегрированной среды разработки программы должна быть использована среда Android Studio.

* 1. **Требования к маркировке и упаковке**

Программа поставляется как программная часть программно-аппаратной системы, не поставляется отдельно.

* + 1. **Требование к маркировке**

Требований к маркировке не предъявляется.

* + 1. **Требования к упаковке**

Требований по упаковке не предъявляется

* 1. **Требования к транспортированию и хранению**

Допускается транспортирование программного изделия в транспортной таре всеми видами транспорта. При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания пыли и атмосферных осадков.

1. **Требования к программной документации**
   1. **Предварительный состав программной документации**

Состав программной документации должен включать в себя:

* техническое задание;
* спецификацию;
* описание применения;
* программу и методику испытаний;
* текст программы.

1. **Стадии и этапы разработки**
   1. **Стадии разработки**

Разработка должна быть проведена в три стадии:

* разработка технического задания;
* исследования и рабочее проектирование;
* внедрение.
  1. **Этапы разработки**

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии исследования и рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

* Исследование алгоритмов для определения психоэмоционального состояния пользователя по данным ЭЭГ;
* Проектирование основной программы для получения и обработки данных ЭЭГ;
* Проектирование программы для администрирования сеанса использования основной программы;
* Разработка указанных программ;
* Разработка технической документации на разработанные программы;
* Тестирование разработанных программ.

На стадии внедрения должен быть выполнен этап подготовки и передачи приложения в эксплуатацию.

* 1. **Содержание работ по этапам**

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

* постановка задачи;
* определение общих требований к техническим средствам;
* определение общих требований к приложению;
* согласование и утверждение технического задания.

На этапе разработки и исследования концептуальной и логической модели программного модуля должны быть выполнены следующие работы:

* выбор инструментальных средств разработки программы;
* разработка структуры требований к разрабатываемому модулю;
* анализ и выбор методов определения психоэмоционального состояния;

На этапе разработки программ должны быть выполнены следующие работы:

* реализация связи с гарнитурой MUSE;
* реализация обработки данных ЭЭГ;
* реализация алгоритмов для определения психоэмоционального состояния;
* организация пользовательского интерфейса.

На этапе разработки программной документации на приложение должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.

На этапе тестирования разработанного приложения должны быть выполнены перечисленные ниже виды работ:

* разработка программы и методики тестовых испытаний программ;
* тестирование и отладка разработанных программ;

На этапе подготовки и передачи программного обеспечения должна быть выполнена работа по подготовке и передаче программ и программной документации в эксплуатацию.

1. **Порядок контроля и приемки**
   1. **Виды испытаний**

Приемо-сдаточные испытания программы должны проводиться согласно разработанной программе и методике испытаний.

Ход проведения приемо-сдаточных испытаний должен быть документирован в протоколе проведения испытаний.

* 1. **Общие требования к приемке работы**

На основании Протокола проведения испытаний подписывается акт приемки-сдачи программы в эксплуатацию.

# ПРИЛОЖЕНИЕ 3

Программное обеспечение системы

для определения психоэмоционального состояния по данным ЭЭГ

с применением биологической обратной связи

**ОПИСАНИЕ ПРОГРАММЫ**

(Р.П.35126-01.13.01)

Листов 11

Санкт-Петербург

2017

АННОТАЦИЯ

В данном документе представлено описание применения программного модуля продукта «Программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи». Приведены общие сведения о модуле, его функциональное назначение, описание логической структуры, описаны используемые технические средства, способ вызова и загрузки, входные и выходные данные.

1. **Назначение программы**
   1. **Функциональное назначение**

Функциональным назначением программного обеспечения является определение психоэмоционального состояния по данным ЭЭГ с использованием биологической обратной связи.

* 1. **Эксплуатационное назначение**

Программа предполагается к использованию в стоматологических клиниках для обучению пользователей контролю за психоэмоциональным состоянием посредством биологической обратной связи.

1. **Условия применения**
   1. **Условия, необходимые для функционирования программы**

Программное обеспечение функционирует на операционной системе Android.

* 1. **Требования к программному обеспечению**

Системные программные средства, используемые программой, должны быть представлены операционной системой Android.

* 1. **Требования к техническим средствам**

Технические средства системы:

- Устройство на ОС Android не ниже версии 5.0 для работы основного приложения;

- Устройство на ОС Android не ниже версии 4.0 для работы администрирующего приложения;

- Гарнитура Muse любой версии.

1. **Описание задачи**
   1. **Используемые методы**

Исходные коды программного продукта реализованы на языке программирования Java. В качестве интегрированной среды разработки использовалась среда Android Studio 2.0, выпускаемая компанией Google. Обе разработанные программы имеют графический пользовательский интерфейс.

Основная программа состоит из двух экранов: экрана выбора гарнитуры Muse для подключения и основного экрана. Экран выбора гарнитуры для подключения содержит список обнаруженных устройств с возможностью выбора конкретного элемента списка. Основной экран содержит воспроизводимый в данный момент видео файл, индикатор эмоционального состояния для обеспечения биологической обратной связи, а также информацию о подключенных устройствах.

В программе используются стандартные классы библиотеки визуальных компонентов Android SDK, включающие в себя стандартные объекты построения пользовательского интерфейса, простые кнопки, поля ввода, информационные поля, что необходимо для организации взаимодействия с пользователем. Для реализации отображения графиков подключается библиотека MPAndroidChart. Библиотека предоставляет классы для отображения различных типов графиков.

* 1. **Вызов и загрузка**
     1. **Основная программа**

Для загрузки программы необходимо запустить его из главного меню ОС, нажав на значок.

При включении программа автоматически осуществляет поиск доступных для подключения гарнитур Muse (Рисунок 4.1). В случае если гарнитура одна, к ней осуществляется подключение. Одновременно с этим начинается воспроизведение случайного видео файла (Рисунок 4.2).

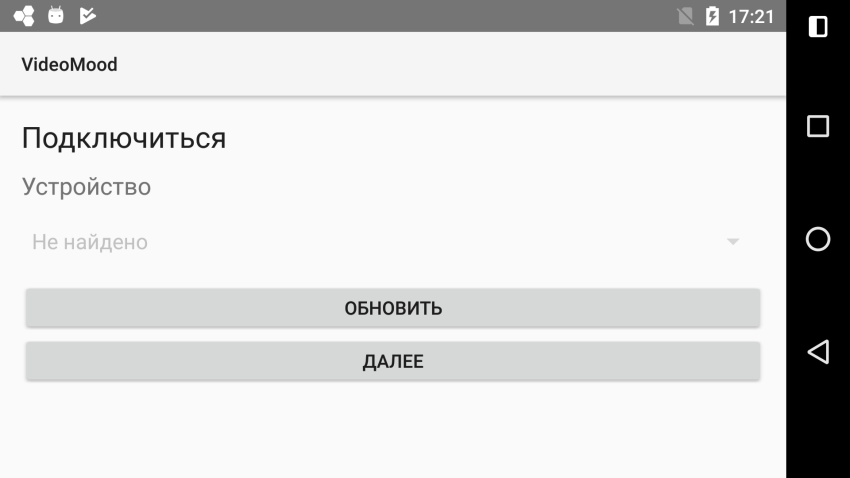


Рисунок П3.1. Интерфейс программы при попытке подключения к гарнитуре Muse

При успешном подключении к гарнитуре Muse, начинается анализ сигнала ЭЭГ, а вместе с ним и отслеживание психоэмоционального состояния (Рисунок 4.3). При включении программа находится в режиме «спокойствие».

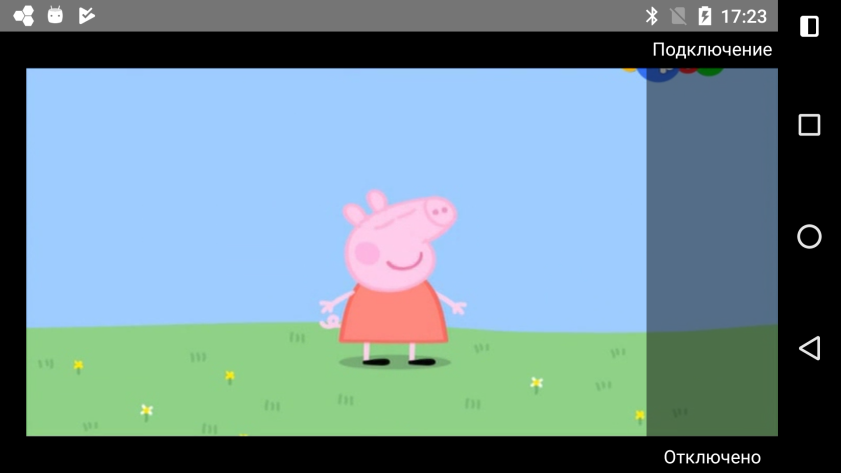


Рисунок П3.2. Интерфейс программы до подключения к гарнитуре Muse

Далее раз в секунду происходит анализ переменной, отвечающей за текущее состояние. При ее изменении в значение «тревога», происходит переключение графического интерфейса в соответствующий режим (Рисунок 4.4).



Рисунок П3.3. Интерфейс программы при подключенной гарнитуре Muse  
 и режиме "спокойствие"

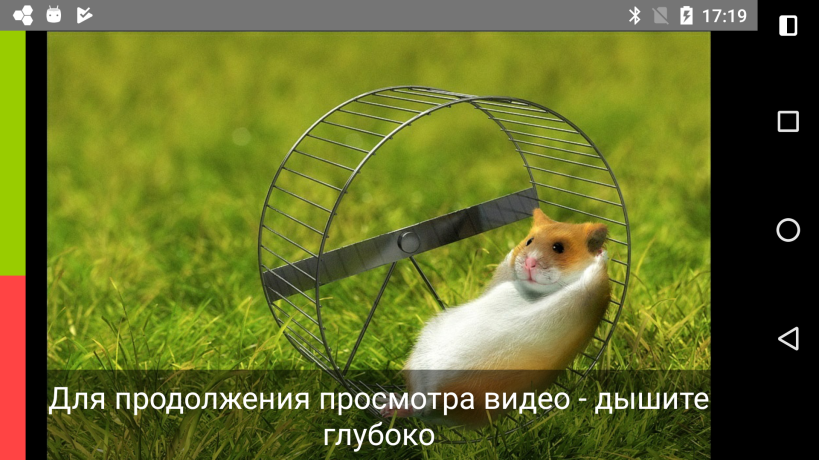


Рисунок П3.4. Интерфейс программы в режиме тревоги

* + 1. **Администрирующая программа**

При запуске администрирующая программа выводит экран, на котором содержится список имеющихся в системе учетных записей (Рисунок П3.5), а также имеется возможность создать новую (Рисунок П3.6) или удалить имеющуюся (Рисунок П3.7).

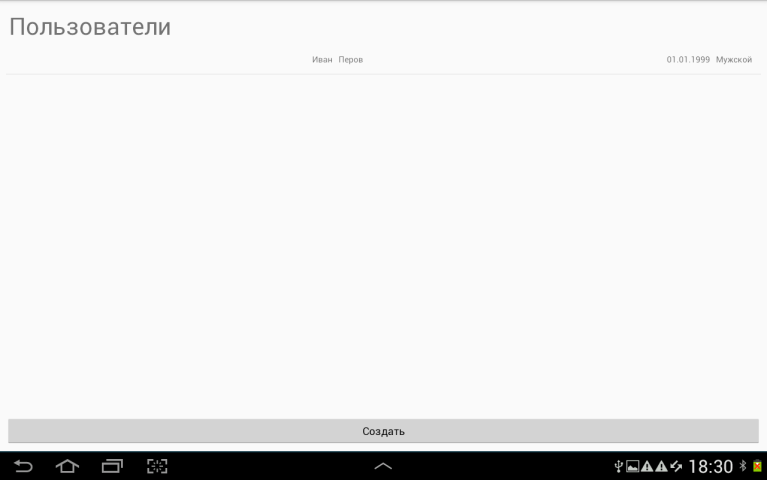


Рисунок П3.5

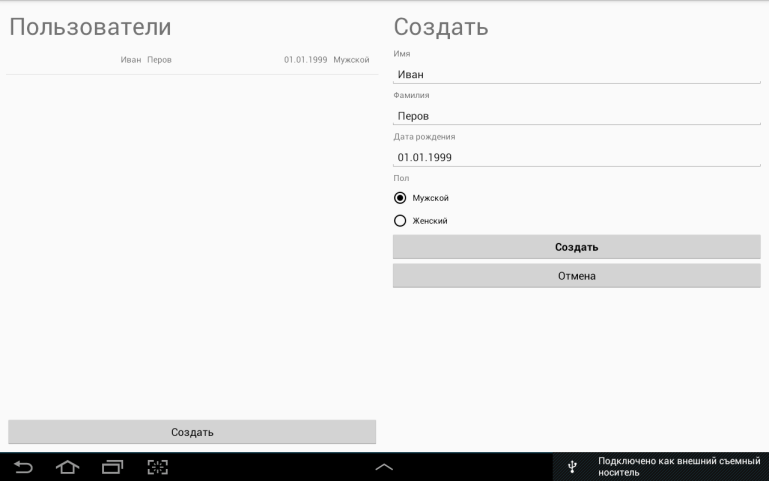


Рисунок П3.6

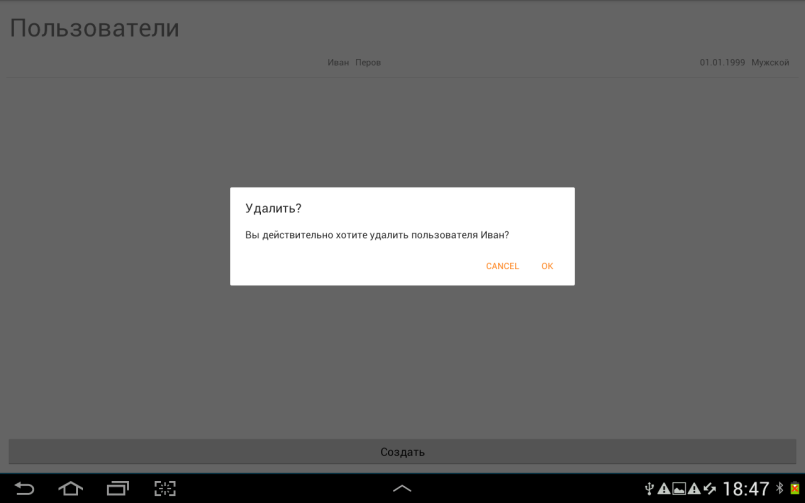


Рисунок П3.7

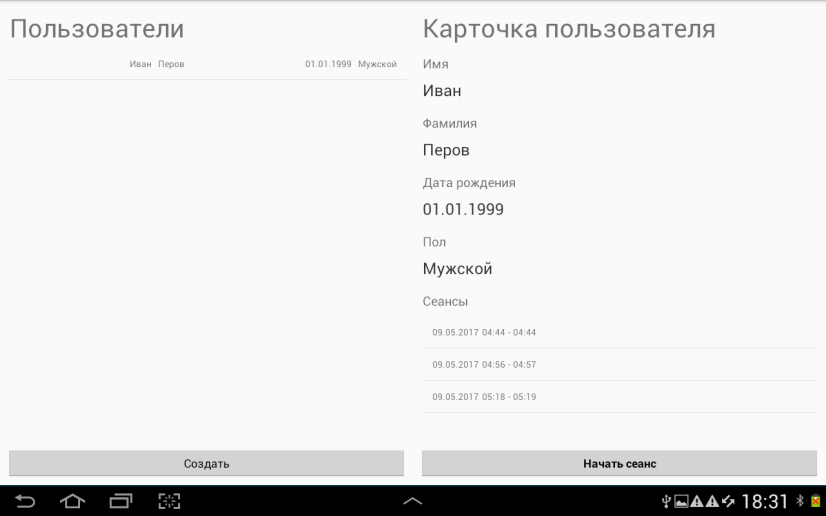


Рисунок П3.8

После выбора учетной записи, для которой будет проводиться сеанс (Рисунок 4.8), появляется экран выбора устройства для подключения (Рисунок 4.9).

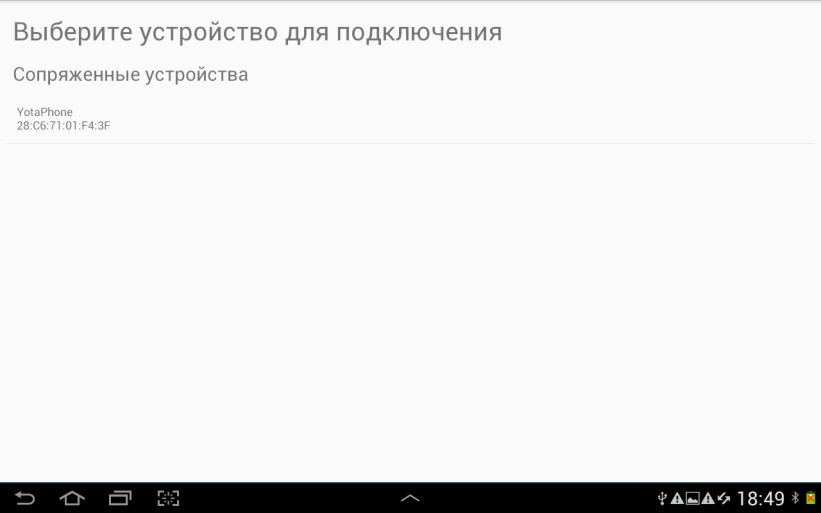


Рисунок П3.9

После выбора устройства для подключения открывается основной экран администрирующего приложения, на котором будет содержаться вся информация о сеансе (Рисунок 4.10).

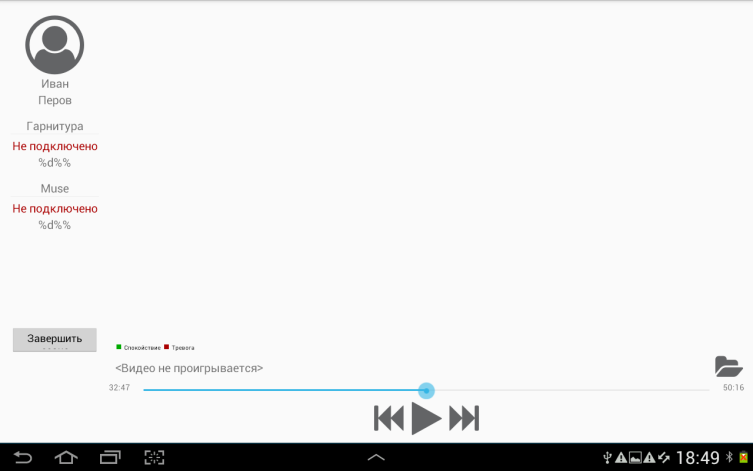


Рисунок П3.10

При успешном подключении к основной программе, статус подключения меняется на «ПОДКЛЮЧЕНО» (верхний левый угол, под именем и фамилией пользователя, под надписью «ГАРНИТУРА»). Там же отображается процент заряда батареи устройства, на котором выполняется основная программа. Ниже отображено состояние подключения гарнитуры Muse (под надписью «MUSE»), а также процент заряда батареи гарнитуры Muse. Состояние подключения дублируется цветом для ускорения считывания информации.

В левом верхнем углу над именем и фамилией пользователя выводится изображение улыбающегося или печального лица для обозначения текущего психоэмоционального состояния пользователя: «спокойствие» или «тревога» соответственно.

В нижней части экрана присутствуют кнопки управления воспроизведением видео файлов (предыдущий, проиграть/остановить/ следующий), ползунок для отображения процесса проигрывания текущего видео файла, значения текущего положения проигрывания (слева от ползунка) и общей длительности файла (справа от ползунка). Кроме того отображается название проигрываемого видео файла и кнопка для выбора другого файла.

Большую часть экрана занимает график со столбцами значений характеристики, по которой происходит определение психоэмоционального состояния. Столбцы окрашиваются в зеленый или красный цвет в зависимости от текущего психоэмоционального состояния (Рисунок П3.12).

По окончании сеанса становится доступна страница с информацией о проведенном сеансе (Рисунок П3.13), в которой отображается вся информация, указанная в соответствующем пункте требований.

В результате тестирования было установлено, что требования к функциональности программ выполнены.

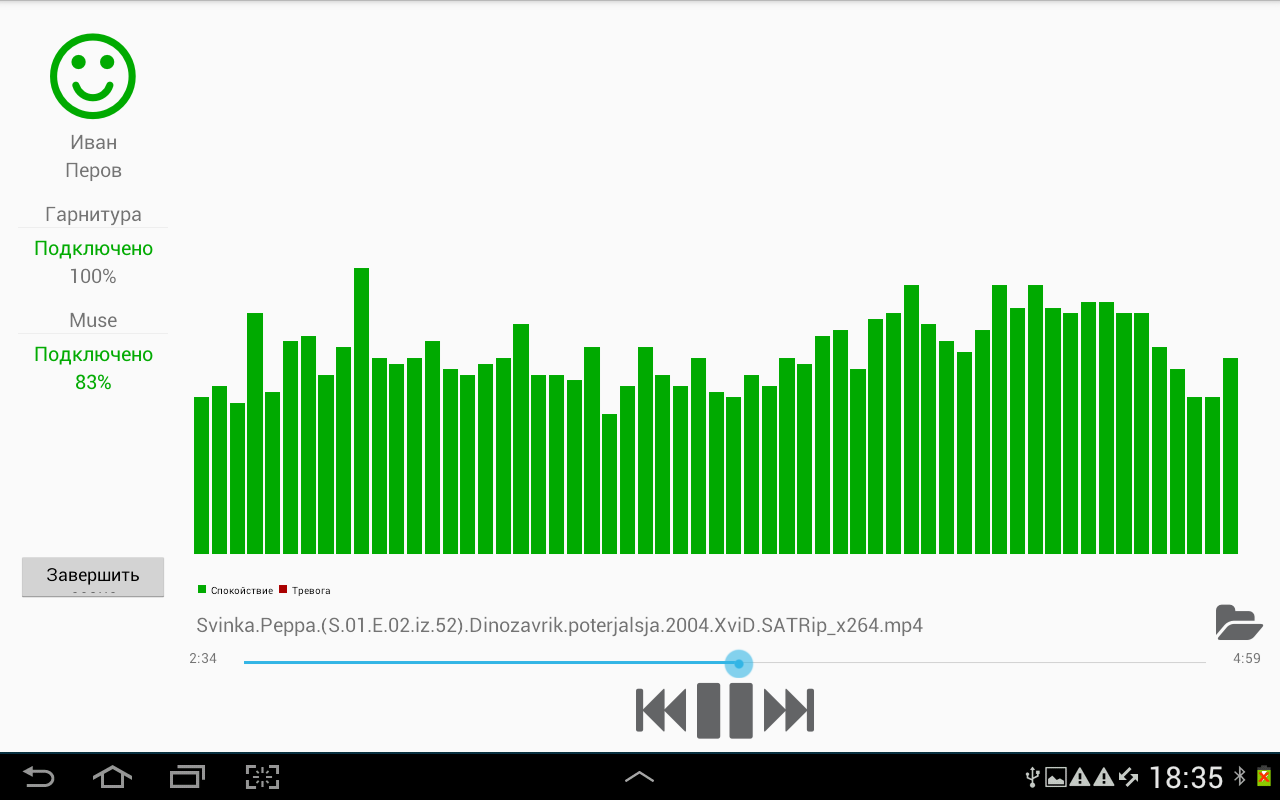


Рисунок П3.11

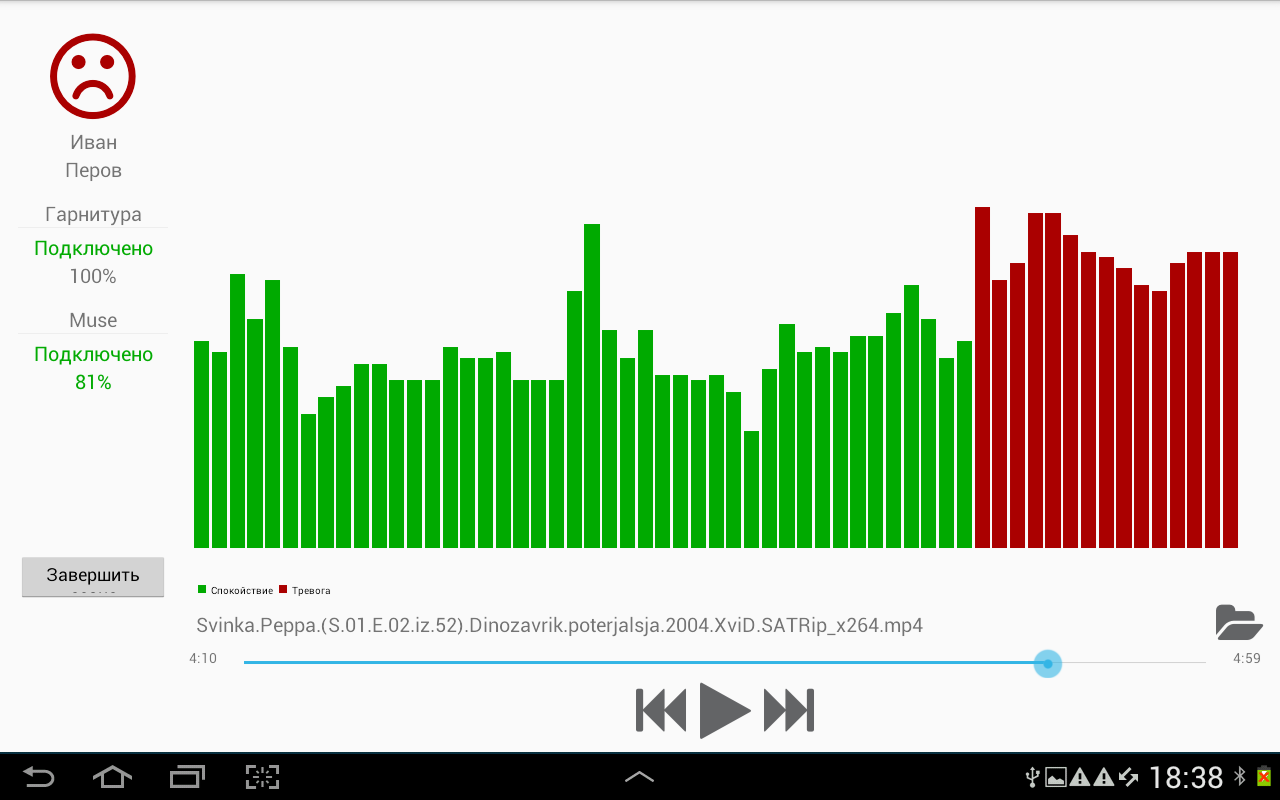


Рисунок П3.12

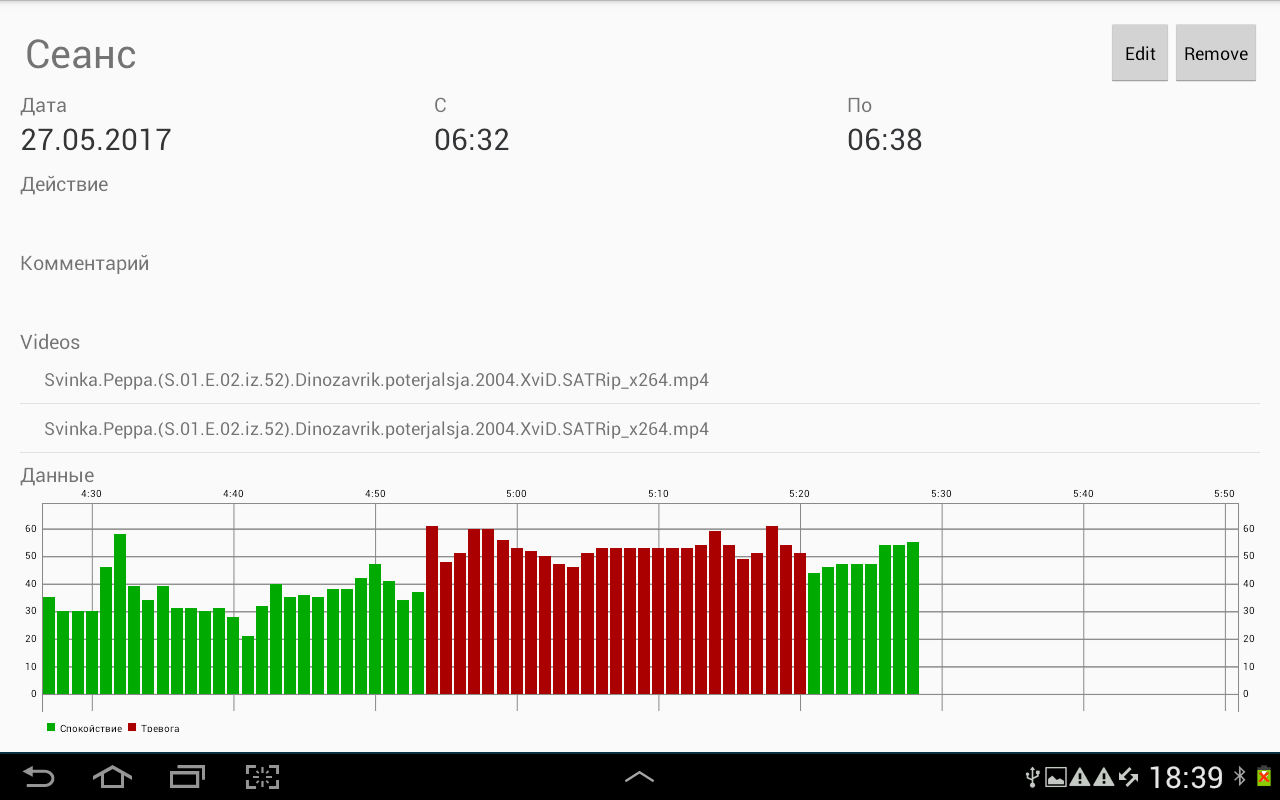


Рисунок П3.13

1. **Входные и выходные данные**

Входными данными являются видео файлы с файловой системы основного устройства, а также пакеты данных сигнала ЭЭГ от гарнитуры MUSE.

Выходными данными являются графический интерфейс с отображаемыми данными о сеансе, либо видео файл с индикатором психоэмоционального состояния. Кроме того выходными данными может считаться база данных с учетными записями пользователей и содержащимися в ней данными о сеансах использования системы.

# ПРИЛОЖЕНИЕ 4

Программное обеспечение системы

для определения психоэмоционального состояния по данным ЭЭГ

с применением биологической обратной связи

**ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ**

(Р.П.35126-01.51.01)

Листов 11

Санкт-Петербург

2017

**АННОТАЦИЯ**

В данном документе представлена программа и методика испытаний на программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи. Приведены основные требования к программному изделию и программной документации, а также описаны средства и порядок проведения испытаний.

1. **Объект испытаний**
   1. **Наименование программы**

Наименование – «Программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи».

* 1. **Область применения**

Программа предполагается к использованию в стоматологических клиниках для обучению пользователей контролю за психоэмоциональным состоянием посредством биологической обратной связи.

* 1. **Обозначение программы**

Наименование темы разработки – «Программное обеспечение системы для определения психоэмоционального состояния по данным ЭЭГ с применением биологической обратной связи».

1. **Цель испытаний**

Цель проведения испытаний – проверка соответствия характеристик разработанной программы (программного изделия) функциональным и отдельным иным видам требований, изложенных в документе «Техническое задание».

1. **Требования к программе**
   1. **Требования к функциональным характеристикам**

Программное обеспечение должно обеспечивать возможность выполнения перечисленных ниже функций:

* Программное соединение компонентов системы (устройств на ОС Android) по каналу связи Bluetooth;
* Воспроизведение видео файлов с файловой системы основного устройства на ОС Android;
* Обработка пакетов данных о сигнале ЭЭГ, поступающем от гарнитуры MUSE;
* Определение психоэмоционального состояния на основании данных ЭЭГ;
* Вывод на экран основного устройства индикатора психоэмоционального состояния;
* Отслеживание изменения психоэмоционального состояния и отображение советов по снижению тревоги при возникновении соответствующего психоэмоционального состояния;
* Обмен данными между основным и администрирующим устройством по каналу Bluetooth;
* Отображение данных, полученных с основного устройства на экране администрирующего устройства;
* Обработка и выполнение команд от администрирующего устройства основным устройством:
  + Приостановка/воспроизведение текущего видео файла,
  + Переключение на следующий/предыдущий видео файл,
  + Получение списка доступных видео файлов,
  + Воспроизведение выбранного из списка доступных видео файлов,
* Хранение данных об учетных записях пользователей и проводимых сеансах
  + 1. **Требования к организации входных данных**

Входными данными являются видео файлы с файловой системы основного устройства, а также пакеты данных сигнала ЭЭГ от гарнитуры MUSE.

* + 1. **Требования к организации выходных данных**

Выходными данными являются графический интерфейс с отображаемыми данными о сеансе, либо видео файл с индикатором психоэмоционального состояния. Кроме того выходными данными может считаться база данных с учетными записями пользователей и содержащимися в ней данными о сеансах использования системы.

* + 1. **Требования к временным характеристикам**

Требования к временным характеристикам программы не предъявляются.

* 1. **Требования к надежности**
     1. Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. Организацией бесперебойного питания технических средств;
2. Регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
3. Регулярным выполнением требований ГОСТ 51188 – 98 «Испытания программных средств на наличие компьютерных вирусов».
   * 1. **Время восстановления после отказа**

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем операционной системы, не должно превышать пятнадцати минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

* + 1. **Отказы из-за некорректных действий оператора**

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой.

* 1. **Условия эксплуатации**
     1. Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

* + 1. **Требования к видам обслуживания**

См. требования к обеспечению надежного (устойчивого) функционирования программы.

* + 1. **Требования к численности и квалификации персонала**

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – оператор системы, при задействовании администрирующего устройства, и конечный пользователь программы.

Оператор системы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы Android, а именно: включением устройства, включением передачи сигнала по каналу Bluetooth, запуском программ, управлением файлами на файловой системе устройства.

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы Android.

* 1. **Требования к составу и параметрам технических средств**

Состав технических средств и их параметры определяются аппаратной частью системы.

* 1. **Требования к информационной и программной совместимости**
     1. **Требования к исходным кодам и языкам программирования**

Исходные коды программы должны быть реализованы на языке Java. В качестве интегрированной среды разработки программы должна быть использована среда Android Studio.

* + 1. **Требования к программным средствам, используемым программой.**

Системные программные средства, используемые программой, должны быть представлены операционной системой Android.

* 1. **Требования к маркировке и упаковке**

Программа поставляется как программная часть программно-аппаратной системы, не поставляется отдельно.

* + 1. **Требование к маркировке**

Требований к маркировке не предъявляется.

* + 1. **Требования к упаковке**

Требований по упаковке не предъявляется

* 1. **Требования к транспортированию и хранению**

Допускается транспортирование программного изделия в транспортной таре всеми видами транспорта. При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания пыли и атмосферных осадков.

1. **Требования к программной документации**

Состав программной документации должен включать в себя:

1. Техническое задание;
2. Спецификацию;
3. Описание применения;
4. Программу и методику испытаний.
5. **Средства и порядок испытаний**
   1. **Технические средства, используемые во время испытаний**

Испытания проводятся на аппаратуре, предназначенной для функционирования системы:

* Устройство на ОС Android не ниже версии 5.0 для работы основного приложения;
* Устройство на ОС Android не ниже версии 4.0 для работы администрирующего приложения;
* Гарнитура Muse любой версии.
  1. **Программные средства, используемые во время испытаний**

Используется только разработанное для системы программное обеспечение.

* 1. **Порядок проведения испытаний**

По завершении этапа исследования и рабочего проектирования Исполнитель переходит к функциональному тестированию разработанного программного обеспечения.

В ходе проведения испытания осуществляется проверка корректности выполнения всех функций модуля приложения, перечисленных в п.3.1.1 настоящего документа, проверка корректности работы программного обеспечения.

Испытания в соответствии с настоящей Программой и методиками. Результат выполнения каждого испытания фиксируется.

1. **Методы испытаний**
   1. **Методика проведения проверки комплектности программной документации**

Проверка комплектности программной документации на программное изделие производится визуально представителями Заказчика. В ходе проверки сопоставляется состав и комплектность программной документации, представленной Исполнителем, с перечнем программной документации, приведенным в п.4.

Проверка считается завершенной в случае соответствия состава и комплектности программной документации, представленной Исполнителем, перечню программной документации, приведенному в указанном выше пункте.

По результатам проведения проверки представитель Заказчика выносит решение о том, что «Комплектность программной документации соответствует (не соответствует) требованиям п.4».

* 1. **Методика проведения проверки комплектности и состава технических и программных средств**

Проверка комплектности и состава технических и программных средств производится визуально представителем Заказчика. В ходе проверки сопоставляется состав и комплектность технических и программных средств, представленных Исполнителем, с перечнем технических и программных средств, приведенным в п.5 настоящего документа.

Комплектность программных средств проводится также визуально.

Проверка считается завершенной в случае соответствия состава и комплектности технических и программных средств, представленных Исполнителем, с перечнем технических и программных средств, приведенных в п.5

По результатам проведения проверки представитель Заказчика выносит решение о том, что «Комплектность технических и программных средств соответствует (не соответствует) требованиям п.5».

* 1. **Метод проверки запуска программы**

Проверка запуска программы выполняется загрузкой исполняемого кода на исполнение в операционной системе.

Проверка считается завершенной в случае успешной загрузки графического пользовательского интерфейса и перехода к одному из экранов основного приложения при запуске основного приложения или одному из экранов администрирующего приложения при запуске администрирующего приложения.

* 1. **Метод проверки соединения с гарнитурой Muse**

Проверка соединения с гарнитурой Muse выполняется запуском приложения при одновременно включенной гарнитуре Muse.

Проверка считается завершенной в случае смены надписи в правом верхнем углу экрана основного приложения на «ПОДКЛЮЧЕНО».

* 1. **Метод проверки соединения с администрирующим устройством**

Проверка соединения с администрирующим устройством выполняется на администрирующем устройстве при выборе устройства для подключения во время работы основного приложения. Проверка считается завершенной в случае смены надписи в правом нижнем углу экрана основного приложения на «ПОДКЛЮЧЕНО» и смены надписи на «ПОДКЛЮЧЕНО» в администрирующем приложении в левой части главного экрана.

* 1. **Метод проверки воспроизведения файлов в основном приложении**

Проверка воспроизведения файлов выполняется при запуске основного приложения. Проверка считается завершенной в случае воспроизведения видео файла при запуске приложения и переходе на главный экран.

* 1. **Метод проверки обработки пакетов данных о сигнале ЭЭГ**

Проверка обработки пакетов данных о сигнале ЭЭГ производится после установления соединения с гарнитурой Muse. Проверка считается завершенной в случае отображения красной и зеленой полосок в левой части главного экрана основного приложения.

* 1. **Метод проверки определения психоэмоционального состояния на основании данных ЭЭГ**

Проверка определения психоэмоционального состояния на основании данных ЭЭГ производится после проверки обработки данных о сигнале ЭЭГ. Проверка считается завершенной в случае переключения главного экрана основного приложения в режим тревоги или выходе из него.

* 1. **Метод проверки отображения данных, полученных с основного устройства на экране администрирующего устройства**

Проверка производится после проверки соединения с администрирующим устройством.

Проверка считается завершенной в случае отображения надписи «ПОДКЛЮЧЕНО» в соответствующей зоне экрана администрирующего приложения.

# ПРИЛОЖЕНИЕ 5

Программное обеспечение системы

для определения психоэмоционального состояния по данным ЭЭГ

с применением биологической обратной связи

**ТЕКСТ ПРОГРАММЫ**

(Р.П.35126-01.12.01)

Листов 1

Санкт-Петербург

2017

Текст программы приведен на прилагаемом компакт-диске в папке VideoMood. Для просмотра может быть использован любой текстовый редактор.