

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э.Баумана)

ФАКУЛЬТЕТ «Факультет международных образовательных программ»

КАФЕДРА «Компьютерные системы и сети»

***РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
НА ТЕМУ:***

**ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА ДЛЯ
УПРАВЛЕНИЯ ЭЛЕКТРОННЫМИ
УСТРОЙСТВАМИ С ПОМОЩЬЮ ИМК**

Студент ИУ6 – 47И
(Группа)

(Подпись, дата)

З. Л. Нгуен
(И.О.Фамилия)

Руководитель ВКР

(Подпись, дата)

к.т.н., доц. А. Ю. Попов
(И.О.Фамилия)

Консультант

(Подпись, дата)

(И.О.Фамилия)

Консультант

(Подпись, дата)

(И.О.Фамилия)

Нормоконтролер

(Подпись, дата)

Ю. И. Бауман
(И.О.Фамилия)

Москва, 2018 г.

АННОТАЦИЯ

Основной целью работы является разработка программно – аппаратного комплекса для реализации архитектуры распределённой вычислительной системы для управления электронными устройствами с помощью ИМК. Данный комплекс реализован как платформа для совместной работы между электроэнцефалографом, микропроцессорными устройствами, искусственной нейронной сетью и облачной платформой.

В результате анализа различных типов интерфейса «мозг – компьютер», а также разных облачных платформ выбран интерфейс на основе вызванной волны P300 для совместной работы с облачной платформой Bluemix от компании IBM. Кроме этого необходимой частью данного проекта является выбор платформы и инструмента для реализации. Для реализации программной части проекта используются такие платформы как Node JS, MySQL и Cloud Object Storage. Для реализации аппаратной части разрабатываемой системы изучена среда разработки основанная на языке Python, которая встроена в ОС Raspbian.

Предложена реализация клиент – серверной структуры системы. Спроектирован пользовательский интерфейс, разработаны алгоритмы работы всех программных модулей, отлажены и протестированы все программные модули для их совместной работы, собрана аппаратная часть, а также указаны перспективы дальнейшего развития проекта.

ANNOTATION

The main goal of the work is the development of a software and hardware complex with the aim of implementing the architecture of a distributed computer system for controlling electronic devices using IMC. This complex is realized as a platform for joint work between an electroencephalograph, microprocessor devices, an artificial neural network and a cloud platform.

As a result of the analysis of various types of brain-computer interface, as well as various cloud platforms, an interface based on the called P300 wave was chosen to work together with IBM's cloud-based Bluemix platform. In addition, a necessary part of this project is the choice of a platform and a tool for implementation. To implement the program part of the project, platforms such as Node JS, mySQL and Cloud Object Storage are used. To implement the hardware part of the system being developed, the development environment was developed based on the Python language, which is built into the Raspbian OS.

The client - server structure of the system is proposed. The user interface is designed, the algorithms of all program modules are worked out, all program modules for their joint work are debugged and tested, the hardware part is collected, and prospects for further development of the project are indicated.

РЕФЕРАТ

РПЗ 108 с., 57 рис., 15 таб., 30 источников.

ИНТЕРНЕТ ВЕЩЕЙ, ОБЛАЧНЫЙ СЕРВИС, СЕРВИС ПРИЛОЖЕНИЯ,
ЭЛЕКТРОЭНЦЕФАЛОГРАФ, ИНТЕРФЕЙС МОЗГ – КОМПЬЮТЕР,
ПАРАЛЛЕЛЬНЫЕ ПОТОКИ, СЕТЬ НЕЙРОННАЯ

Основной целью данной работы является разработка унифицированной архитектуры программно – аппаратной системы на базе IoT, которая используют интерфейс мозг – компьютер для управления устройствами.

В результате анализа различных решений в области человеко – машинных систем, известной как интерфейс мозг – компьютер, возникла идея обеспечить больным людям возможность взаимодействия с внешним миром. Особенно для людей лишенных естественных каналов коммуникации (в частности, для полностью парализованных лиц), эта технология в настоящее время стремительно развивается и по прогнозам ведущих специалистов по влиянию на развитие цивилизации может оказаться сопоставима с появлением письменности. Возможность управления внешними устройствами и информационными каналами посредством электрической активности собственного мозга может не только качественно преобразить взаимодействие человека с внешним миром, но и оказать существенное влияние на развитие его внутреннего мира.

Система интересна тем, что она является программно – аппаратным продуктом, что позволяет разработчику проектировать не только программную, но и аппаратную часть. Для реализации аппаратной части системы использованы: одноплатный компьютер Raspberry pi – он работает как вычислительный хаб, позволяя получать данных с ЭЭГ и отправлять данные на облако. При разработке программной части системы использованы самые современные технологии, такие как NodeJS – среда разработки программных продуктов, система управления базами данных MySQL Workbench 6.3 CE,

Websocket - для транзакции реальных данных в онлайн режиме без задержки, HTTP - запросы - для получения данных с сервера и базы данных, MQTT – протокол, JSChart – библиотека для динамической визуализации данных, а также Bootstrap – для верстки сайта.

Актуальность данной разработки заключается в том, что у нее нет аналогов. А также она позволяет с помощью мозговой активности управлять электронными устройствами.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	11
ВВЕДЕНИЕ.....	13
1 Исследование методик и средств построения устройств ИМК с использованием облачных технологий.....	14
1.1 Анализ использования облачных технологий для реализации задач Интернета вещей.	14
1.1.1 Тенденция развития инфраструктуры IoT.....	14
1.1.2 Инфраструктура типового решения IoT	15
1.1.3 Анализ облачной платформы Bluemix.....	16
1.1.4 Краткое описание концепций Bluemix.....	19
1.1.5 Развертывание и управление приложением	21
1.2 Интерфейс "мозг-компьютер" на основе вызванной волны P300.....	22
1.2.1 Интерфейсы на основе мю-ритма.....	24
1.2.2 Интерфейсы на основе альфа-ритма	25
1.2.3 Интерфейсы на основе SSVEP и айтрекинг	26
1.2.4 Интерфейсы, использующие P300	27
1.3 Принцип действия ЭГГ и основы ЭГГ головного мозга человека	29
1.4 Анализ требования к выбору аппаратных составляющих для проекта	30
1.4.1 Одноплатный компьютер Raspberry Pi 3.....	31
1.4.2 Восьмиканальный электроэнцефалограф ИНЭУМ им И.С.Брука.....	32
1.5 Постановка задачи на проектирование ВСУЭУ	34
Вывод к исследовательской части.....	35
2 Разработка программная и аппаратная часть проекта. Описание архитектуры и алгоритма работы отдельных подсистем (конструкторская часть)	36
2.1 Выбор архитектуры системы	36
2.2 Выбор языка и технологии программирования	43
2.3 Описание работы подсистем и программные модули.....	45
2.3.1 Главная подсистема	45

2.3.2 Подсистема выбора режимов.....	48
2.3.3 Подсистема отвечающая за режим просмотра профиля.....	50
2.3.4 Подсистема, отвечающая за режим обучения И. Н. С.	52
2.3.5 Модуль отвечающий за обучения И. Н. С.....	57
2.3.5.1 Выбор топологии нейронной сети.....	59
2.3.5.2 Библиотека PyBrain.....	61
2.3.6 Модуль работы с ЭЭГ.....	63
2.3.6.1 Описание процесса взаимодействия с проборм.....	65
2.3.6.2 Описание процесса парсинга данных	67
2.3.7 Модуль отображения интерфейса	70
2.3.8 Модуль работы с файлом	71
2.3.9 Подсистема отвечающая за режим работы	73
2.3.10 Модуль отвечающий за выбор бинарного файла	78
2.3.11 Модуль, отвечающий за работу механизированной руки	80
2.3.12 Подсистема отображения графиков	83
2.3.13 Описание процесса передачи данных по Websocket.....	84
Вывод конструкторской части.....	87
3. Разработка технологии использования ВСУЭУ	88
3.1 Настройка удаленного доступа к raspberry pi.....	88
3.2 Установка необходимых библиотек для запуска и отлаживания программного кода	91
3.3 Публикация веб приложение на облачной платформе Bluemix.....	92
3.3.1 Установка Bluemix CLI.....	92
3.3.2 Публикации проекта в облаке Bluemix	93
3.4 Доступ к облачному платформу Bluemix	94
3.5 Доступ в БД.....	95
3.6 Доступ к облачной хранилище IBM Cloud Object Storage.....	96
3.7 Выбор стратегии тестирования и разработка тестов.....	97
Вывод технологической части.....	104
ЗАКЛЮЧЕНИЕ	105

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	106
ПРИЛОЖЕНИЕ А. Техническое задание.....	109
ПРИЛОЖЕНИЕ Б. Руководство пользователя	118
ПРИЛОЖЕНИЕ В. Графические материалы	125

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

IoT – это концепция вычислительной сети физических объектов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключаящее из части действий и операций необходимость участия человека.

MQTT (*Message Queue Telemetry Transport*) - упрощенный протокол сетевого уровня для обмена сообщениями между устройствами. Этот протокол работает поверх стека TCP/IP и разработан для преодоления проблем, связанных с подключением быстро растущего числа датчиков, микрокомпьютеров, приводов, телефонов, планшетов. В настоящее время MQTT является наиболее распространенным протоколом для организации IoT инфраструктуры.

CRC - алгоритм нахождения контрольной суммы, предназначенный для проверки целостности данных.

FIFO (акроним First In, First Out — «первым пришёл — первым ушёл») — способ организации и манипулирования данными относительно времени и приоритетов.

ВСУЭУ – Вычислительная система для управления электронными устройствами с помощью ИМК

IoT - Internet of Things (Интернет вещей).

ПО – Программное обеспечение.

API – Application programming interface (интерфейс программирования приложений, интерфейс прикладного программирования).

ОС – Операционная система.

AJAX – asynchronous Javascript and XML.

API – application programming interface.

HTML – hypertext markup language.

URL – uniform resource locator.

БД – база данных.

ИМК - Интерфейс мозг-компьютер.

ЭЭГ – Электроэнцефалография.

Motor imagery, МІ – мысленное воображение движений.

И. Н. С. – Искусственная нейронная сеть.

ВВЕДЕНИЕ

Основанием для разработки данного проекта является острая потребность в системе, которая позволила бы использовать реакцию головного мозга для управления электронными устройствами.

Основной целью ВСУЭУ является разработка унифицированной архитектуры программно – аппаратной системы на базе технологии IoT, включающей облачные технологии, сетевые технологии, беспроводные сети, микропроцессоры, базы данных, современные языки программирования для реализации интерфейса мозг – компьютер.

В рамках данной магистерской работы построен аппаратно – программный комплекс, позволяющий с помощью мозговой активности управлять электронными устройствами, а также реализован сервер на базе облачной технологии от IBM Bluemix, где хранятся основные подсистемы хранения и визуализации данных.

Основными задачами данной работы являются:

1 Исследование применения облачных технологий для аппаратных разработок.

2 Исследование предметной области связанной с ИМК.

3 Создание тестового набора, достаточного для проверки правильности и слаженности работы системы в целом.

4 Разработка ВСУЭУ, состоящей из следующих компонентов: подсистема базы данных, подсистема сбора данных с ЭЭГ, подсистема отправки данных в облачное хранилище, подсистема декодирования данных, подсистема работы с запросами для получения необходимых данных из БД, подсистема сбора и анализа результатов, подсистема визуализации данных.

Подводя итоги всему вышесказанному, можно отметить, что ВСУЭУ является шаблоном для развертывания более крупных проектов на базе совместной работы ИМК и облачной платформы Bluemix.

1 Исследование методик и средств построения устройств ИМК с использованием облачных технологий

1.1 Анализ использования облачных технологий для реализации задач Интернета вещей.

1.1.1 Тенденция развития инфраструктуры IoT

Воплощение потенциала идей, заложенных в концепцию Интернета вещей, способно существенно изменить уклад современной экономики. Благодаря внедрению масштабируемых облачных решений, использованию большого количества датчиков и распределенных микропроцессорных систем уже в ближайшее время могут быть созданы прорывные решения в таких областях, как: транспорт, сельское хозяйство, промышленное производство, здравоохранение, социальная сфера, быт и других. Все большее количество компаний обращает внимание на применение идей и технологий Интернета вещей для внедрения аналитики их деятельности и поиска новых возможностей для продуктов и услуг .

Перечислим некоторые интересные факты: - По оценкам специалистов, к 2020 году к сети Интернет будет подключено до 50 млрд устройств, 20 млрд из них будут задействованы в инфраструктуре IoT. На рисунке 1 показан график, показывающий тенденции развития и внедрения IoT в промышленности [1][2].

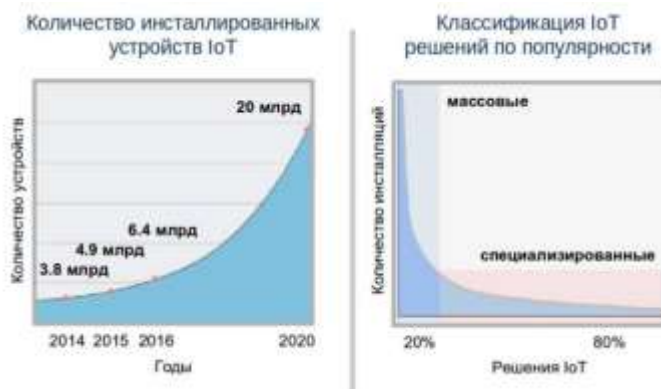


Рисунок 1 – Тенденция внедрения IoT в специализированных отраслях промышленности

- До 90% данных, анализируемых устройствами Интернета вещей ранее не подвергались обработке.
- До 60% данных, получаемых устройствами Интернета вещей, остаются актуальными лишь несколько миллисекунд.
- В настоящее время только 0,1% устройств, способных выполнять полезную вычислительную нагрузку, подключены к сети.

1.1.2 Инфраструктура типового решения IoT

Под типовым решением Интернета вещей в данном контексте понимается распределенная кибер-физическая система, интегрирующая вычислительные ресурсы в физические процессы. В такой системе должны быть реализованы следующие основные функции:

- Сбор первичных данных с помощью сенсоров, расположенных в непосредственной близости от реальных объектов.
- Управление объектами через актуаторы, подключенные к микрокомпьютерам.
- Передача первичных данных от микрокомпьютеров в вычислительный хаб и в обратном направлении.
- Первичная обработка данных в вычислительном хабе, формирование пакетов данных для передачи их в облако.
- Получение и хранение данных в облаке.
- Аналитическая обработка в облаке и формирование ключевых показателей эффективности (KPI) на основе данных об объектах, данных от сторонних источников, исторических данных.
- Визуализация данных и результатов анализа на различных платформах: мобильных устройствах, носимой электронике, планшетах, компьютерах, мониторах и пр.
- Прием команд от внешних управляющих консолей.

– Принятие решений на основе команд, выработка управляющих и информационных сообщений для актуаторов.

– Передача управляющих сообщений в вычислительные хабы.

На рисунке 2 показан простой пример реализации IoT [3].

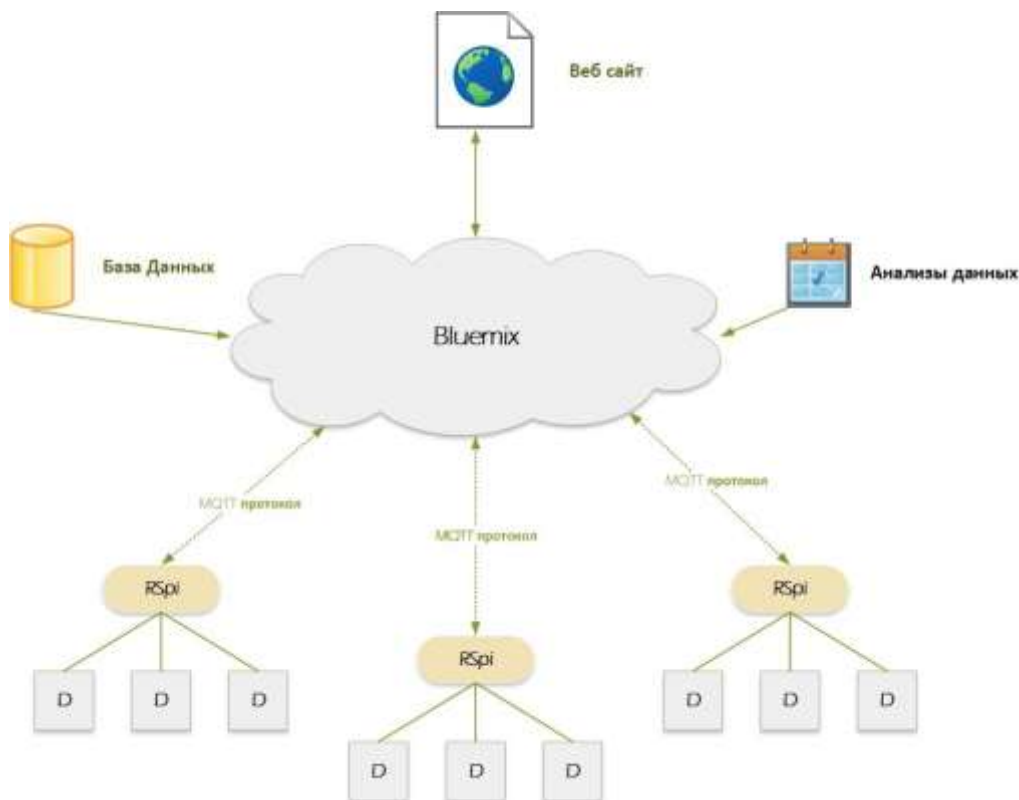


Рисунок 2 – Пример решения Интернета вещей

D – Датчики.

RSpi – raspberry pi

1.1.3 Анализ облачной платформы Bluemix

Bluemix — это открытое облачное предложение типа PaaS (Platform-as-a-Service) на базе проекта с открытым исходным кодом Cloud Foundry. Эта платформа предназначена для разработки и хостинга приложений, а также упрощения задач по управлению инфраструктурой. Она позволяет быстро создавать и разворачивать приложения, а также управлять ими [4][5][6].

Bluemix обеспечивает следующие возможности:

– быстрое и инкрементное составление приложений из сервисов;

- непрерывное внесение изменений в приложения и обеспечение постоянной доступности;
- поддержка высокоспециализированных моделей программирования и сервисов для конкретных рабочих нагрузок;
- встраивание высокой степени управляемости в сервисы и приложения;
- оптимизация и эластичная адаптация к рабочей нагрузке.



Рисунок 3 – Каталог компонентов BlueMix

Каталог BlueMix (Рисунок 3) содержит большую часть из того, что необходимо для быстрого начала работы, большое количество шаблонов, заранее сконфигурированных наборов сервисов, сред исполнения и примеров кода, готовых к использованию:

- Сред исполнения, в том числе: Liberty for Java, Node.js, Ruby on Rails.
- Веб-сервисов и сервисов приложений, в том числе: Data/Session Cache, ElasticMQ, Decision, SSO, Log Analysis, Redis, RabbitMQ, Twilio.
- Мобильных сервисов, в том числе: push-уведомлений, Cloud Code, Mobile Application Management, Mobile Quality Assurance.
- Сервисов управления данными, в том числе: MongoDB, реляционной базы данных от IBM, JSON – базы данных от IBM, MySQL, PostgreSQL, MobileData, Mobile Sync, BLU Data Warehouse, MapReduce.
- Сервисов мониторинга и анализа.

– Сервисов DevOps Services (прежнее название: JazzHub).

Bluemix обеспечивает доступ к широкому спектру служб, которые можно встраивать в приложения. Некоторые из этих служб происходят из Cloud Foundry, другие поставляются IBM и сторонними поставщиками. В каталог регулярно добавляются новые и усовершенствованные службы.



Рисунок 4 – Принцип работы BlueMix

Платформа BlueMix достигает этих целей посредством абстрагирования и скрытия большинства сложностей (на рисунке 4), традиционно сопутствующих хостингу приложений в облаке и управлению ими в облачной среде. Bluemix может быть использована разработчиками для создания и применения самых разных приложений, включая веб-приложения, мобильные приложения, приложения для работы с большими данными, приложения для разумных устройств и т.д [3]. Bluemix поддерживает разработку на популярных языках программирования и средах разработки. Java-технологии, средства

создания серверных частей для мобильных приложений, мониторинг приложений, технологии с открытым исходным кодом и т. д. — все эти возможности доступны в облаке как сервисы.

1.1.4 Краткое описание концепций Bluemix

В терминологии Bluemix приложение (application) — это созданный вами артефакт, т. е. весь программный код (исходный код или исполняемые двоичные файлы), который необходимо запустить или на который необходимо сослаться в процессе исполнения. Мобильные приложения выполняются за пределами среды Bluemix и используют сервисы Bluemix, представленные приложениями. В случае веб-приложений приложение — это код, загруженный на платформу Bluemix с целью хостинга. Кроме того, платформа Bluemix способна осуществлять хостинг программного кода приложения, который вы хотите выполнять на внутреннем сервере в среде на базе контейнера [7].

На рисунке 5 показаны принципы взаимодействия Bluemix с клиентскими приложениями.

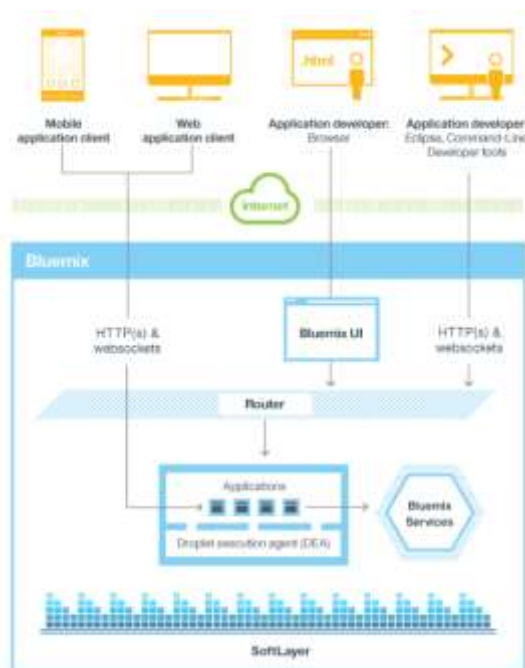


Рисунок 5 – Принципы взаимодействия Bluemix с клиентскими приложениями

Сервис (service) — это код, работающий на платформе Bluemix и предлагающий некоторую функциональность, которую могут использовать приложения. Это может быть готовый сервис, используемый непосредственно — например, push-уведомления для мобильных приложений или эластичное кэширование для веб-приложения. Вы также можете создавать собственные сервисы в диапазоне от простых служебных функций до сложной бизнес-логики.

Организация (organization) и пространство (space) — это организационные единицы инфраструктуры, способные хранить и отслеживать ресурсы приложения. Организация содержит домены (domain), пространства и пользователей. Пространство содержит приложения и сервисы. По умолчанию используется три пространства: Development (разработка), Production (производство) и Staging (подготовка). Для приложений, которым требуется среда типа PaaS, предоставляются buildpack-пакеты, каждый из которых представляет собой набор скриптов для подготовки кода к исполнению на целевой PaaS-платформе. Buildpack-пакеты, которые включают необходимую вашим приложениям среду исполнения и могут также содержать специализированные инфраструктуры, упрощают развертывание приложения в облаке по сравнению с самостоятельной установкой и конфигурированием среды исполнения.

Использование сервисов в Bluemix включает три этапа: 1. Сообщите платформе Bluemix, что вам требуется новый экземпляр сервиса и какое конкретное приложение будет использовать этот новый экземпляр. 2. Bluemix автоматически инициализирует новый экземпляр этого сервиса и свяжет его с приложением. 3. Приложение взаимодействует с сервисом.

Пакеты сервисов (Service bundles) — это коллекции API-интерфейсов, используемых в конкретных областях. Например, пакет Mobile Services включает сервисы MobileData, Cloud Code, Push и Mobile Application Management. Доступные сервисы и среды исполнения представлены в каталоге Bluemix. Кроме того, вы можете зарегистрировать собственные сервисы.

1.1.5 Развертывание и управлением приложением

Чтобы развернуть свое приложение, необходимо загрузить его в среду *Bluemix* и указать, сколько экземпляров этого приложения должно исполняться, а затем сконфигурировать *Bluemix*, введя необходимую информацию для поддержки этого приложения [8].

В случае мобильного приложения среда *Bluemix* содержит артефакт, который представляет серверную часть мобильного приложения — набор сервисов, который использует приложение для взаимодействия с сервером. *Bluemix* поддерживает серверные компоненты мобильного приложения, взаимодействующие с сервисами PushWorks, Cloud Code и Mobile Data, непосредственно из пользовательского интерфейса *Bluemix*.

В случае веб-приложения необходимо предоставить в *Bluemix* соответствующую информацию о среде исполнения и среде разработки, чтобы платформа смогла сформировать надлежащую инфраструктуру для исполнения этого приложения.

При развертывании приложений и управлении ими можно использовать инструмент командной строки cf, веб-интерфейс *Bluemix* или сервисы DevOps Services [9].

Браузерные и мобильные клиенты — а также другие приложения, развернутые на платформе *Bluemix* и выполняющиеся за ее пределами — взаимодействуют с приложениями, работающими на платформе *Bluemix*, через API-интерфейсы типа REST/HTTP. Каждый клиентский запрос маршрутизируется к одному из экземпляров приложения или составляющих его сервисов. Среды исполнения приложений в *Bluemix* изолированы друг от друга даже тогда, когда они находятся на одной и той же физической машине[10].

В ходе управления приложениями можно запускать, останавливать, перезапускать экземпляры приложения (или, в случае веб-приложения, изменять их количество), а также изменять объем памяти, используемый приложением. Ключевая конструктивная особенность *Bluemix* — отличные показатели при

хостинге масштабируемых приложений и артефактов приложений. На данный момент эта платформа не масштабирует приложение автоматически в соответствии с нагрузкой, поэтому этим процессом необходимо управлять самостоятельно посредством создания или удаления экземпляров при изменении рабочей нагрузки. По этой причине ваши приложения должны сохранять все персистентные данные за пределами приложения в одном из сервисов хранения данных, предоставляемых платформой *Bluemix*. При повторном развертывании приложения после обновления используется тот же процесс, что и при начальном развертывании. *Bluemix* останавливает все исполняющиеся экземпляры и переводит новые экземпляры в рабочее состояние автоматически.

1.2 Интерфейс "мозг-компьютер" на основе вызванной волны P300

Основной целью применения ИМК является получение компьютерной системой однозначно интерпретируемых команд непосредственно от головного мозга без использования мышечной активности. Для разработки ИМК следует рассматривать три основные парадигмы[11][12]:

- Неинвазивный ИМК, основанный на распознавании ментальных состояний, вызванных воображаемым выполнением движений. Он обеспечивает формирование дискретных управляющих команд и требует минимального времени обучения оператора при достаточно высокой производительности.

- Неинвазивный ИМК, использующий принцип непрерывного управления. После выработки навыка управления, такой ИМК позволяет управлять внешним устройством как собственным (виртуальным) исполнительным органом, не требуя ментального кодирования дискретного набора команд.

- Инвазивный ИМК, основанный на двусторонней связи мозг-компьютер посредством имплантируемых электродов и позволяющий полностью инкорпорировать внешние технические устройства во внутреннюю нейронную

модель схемы тела и, соответственно, оперировать с ними так же, как и с естественными исполнительными органами.

В магистерской работе будут применяться неинвазивные ИМК первого типа, построенные на основе многоканальной электроэнцефалограммы головного мозга (ЭЭГ). В основе такой системы лежит регистрация электрического потенциала на поверхности головы и его компьютерная обработка[13].

В ЭЭГ человека прослеживается определенная ритмическая активность, которая делится на несколько групп в зависимости от частоты волн (альфа-ритм, бета-ритм, гамма-ритм, дельта-ритм, мю-ритм). Для состояния бодрствования характерны бета- и мю-ритмы. Гамма- и дельта-ритмы появляются при засыпании и во сне. Альфа-ритм появляется при закрывании глаз, а также в медитативных состояниях. В связи с этими особенностями ритмов в ИМК наиболее часто используются мю-, бета- и альфа-ритмы. (рисунок 6 и 7).

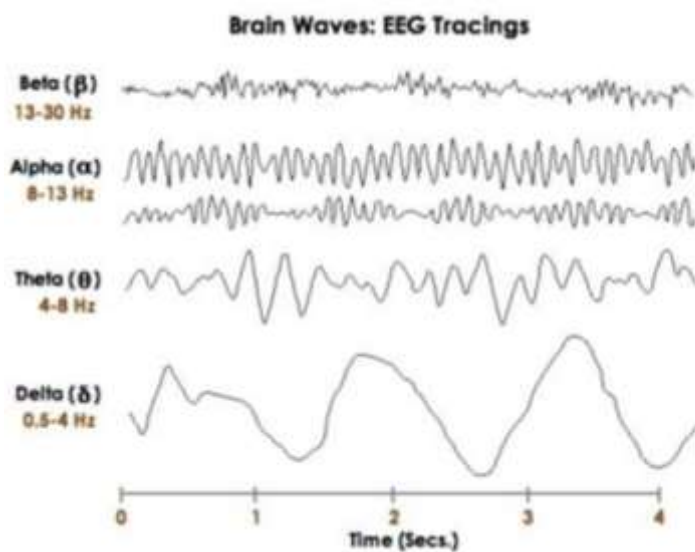


Рисунок 6 - Основные ритмы ЭЭГ человека

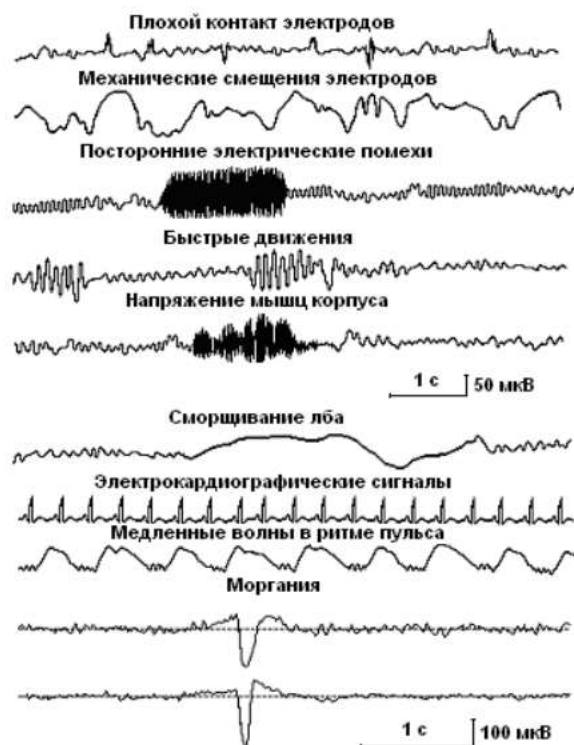


Рисунок – 7 Регистрируемые на ЭЭГ потенциалы, возникающие не в головном мозге – артефакты

1.2.1 Интерфейсы на основе мю-ритма

Интерфейсы на основе мю-ритма, как правило, используют моторное воображение (motor imagery, MI) – мысленное воображение движений (например, поднятия левой или правой руки), при котором человек не совершает реальных движений. При представлении движений происходит подавление характерного для состояния бодрствования мю-ритма (частотой 7-11 Гц), регистрируемого электродами, расположенными на коже головы в центральных и центрально-височных областях. Пользователя подобным интерфейсом инструктируют, что в момент, когда он хочет отдать команду интерфейсу, он должен представить то или иное движение. Классификатор обучают различать два типа электрической активности – наличие мю-ритма (синхронизация) и отсутствие мю-ритма (десинхронизация, замещение мю-ритма в ЭЭГ менее синхронными низкоамплитудными колебаниями). Таким образом, после

обучения классификатора, можно установить, что в момент, когда произошла десинхронизация мю-ритма, пользователь хотел отдать команду, и запрограммировать компьютер на определенные действия, совершаемые в этом случае [14].

Так, в одном из исследований с применением подобного интерфейса, айтрекинг использовался для отслеживания взгляда пользователя, выбирающего одну из нескольких целей на мониторе компьютера, а моторное воображение – для подтверждения выбора (O'Doherty et al., 2014). В другом исследовании к компьютеру была подключена система управления экзоскелетом руки, который был разработан для больных гемипарезом. С помощью айтрекинга происходило отслеживание намерения пользователя взять тот или иной объект в реальной среде (выбор цели), а моторное воображение использовалось для контроля различных параметров движения руки (скорость, ускорение) и для конечного захвата объекта (Frisoli et al., 2012)[15]. Существуют и другие варианты интерфейсов. Интерфейсы, использующие моторное воображение, удобны в использовании, так как обеспечивают интуитивное управление (представил движение – произошло движение, при условии, что интерфейс подключен, например, к роботизированной руке), но требуют достаточно длительного обучения.

1.2.2 Интерфейсы на основе альфа-ритма

Интерфейсы, использующие альфа-ритм, устроены таким образом, что для отдачи определенной команды нужно закрыть глаза. Пример подобного гибридного интерфейса – система управления роботизированной рукой, в которой саккады использовались для перемещения руки в одном из четырех направлений, а для сжатия руки пользователю необходимо было закрыть глаза, при этом интерфейс улавливал изменение в электроэнцефалограмме (появление альфа-ритма) (Postelnicu et al., 2011). Интерфейсы также вполне успешно работают, но минус достаточно очевиден – при отдаче команды ненадолго

теряется зрительная связь с окружающей средой, и, кроме того, подобное управление несколько утомительно [16]. Кроме того, у некоторых людей альфа-ритм слабо выражен, что затрудняет широкое применение интерфейсов, использующих его для управления.

1.2.3 Интерфейсы на основе SSVEP и айтрекинг

В других гибридных интерфейсах используются зрительные вызванные потенциалы стабильного состояния (SSVEP) и айтрекинг. SSVEP - устойчивые зрительные вызванные потенциалы, которые возникают при стимуляции на частоте от 3,5 до 75 Гц (Beverina et al, 2003), при этом частота потенциалов повторяет частоту предъявления зрительных стимулов. Пользователей данных интерфейсов инструктируют, что для отдачи команды необходимо сосредоточить внимание на зрительном стимуле. Классификатор обучается различать изменения в ЭЭГ при появлении SSVEP на стимул, мигающий с определенной частотой. Один из примеров подобной гибридной системы – система для набора текста, совмещающая в себе SSVEP и айтрекинг (Lee et al., 2013)[17]. В этой системе 30 клавиш (буквы английского алфавита и другие кнопки, необходимые для ввода текста) постоянно мигали, каждая на своей частоте, для выбора определенной буквы нужно было сконцентрировать внимание на ней, при этом контролировалось положение взгляда. В случае, если оно сильно не соответствовало положению клавиши, команда напечатать букву не отдавалась.

Другой пример гибридного интерфейса, совмещающий в себе ИМК-SSVEP и айтрекинг – система управления простой игрой, в которой нужно собрать пазл (Kos'Myna, 2013)[18].

Интерфейсы на основе SSVEP и айтрекинга, в которых используется большое количество стимулов, в ряде случаев могут вызывать достаточно сильное утомление, так как пользователю приходится постоянно смотреть на

экран, где мигает большое число стимулов на разной частоте. Кроме того, при некоторых условиях они потенциально эпилептогенны.

1.2.4 Интерфейсы, использующие P300

Компонент P300 возникает в ответ на неожиданный редко предъявляемый (например, предъявляемый с вероятностью 0.2) значимый стимул, когда он появляется среди часто предъявляемых незначимых стимулов.

P300 возникает примерно на 300 мс после предъявления значимого стимула, имеет длительность около 300–400 мс и положительную амплитуду 5–15 мкВ. Максимальная амплитуда P300 наблюдается под центральным электродом. Чем реже предъявляется значимый стимул, тем больше амплитуда P300. Как правило, требуется несколько усреднений для его выделения из фоновой активности. P300 зависит от внимания испытуемого, но не от физических параметров стимула.

P300 — часть сложного потенциала, отражающего процессы переработки информации в мозге, связанные с направленным вниманием при выполнении когнитивной задачи. Физические параметры стимула отражаются в параметрах ранних компонентов вызванных потенциалов. Процессы опознания и классификации стимулов отражаются в компонентах с латентностью 96-250 мс после начала стимула, которые принято обозначать как волну N200. Непосредственно с потенциалом P300 связаны завершающие этапы обработки информации - окончательная классификация стимула и принятие решения о действии, связанном со стимулом (Picton, 1992). Существуют некоторые особенности эндогенных вызванных потенциалов на различные зрительные стимулы. Так, некоторых работах показано, что ЗВП на изображения лиц имеют характерные именно для такой стимуляции компоненты (Zhang et al., 2012)[19].

Для выделения P300 используется «оддболл» парадигма или ее модификации — человеку предъявляются целевые и нецелевые стимулы (зрительные или слуховые), причем целевых стимулов на порядок меньше,

дается задание считать редкие стимулы про себя. Электроэнцефалограмму разбивают на эпохи относительно подачи редкого стимула, полученные отрезки суммируют, таким образом выделяются вызванные потенциалы, в которых с латентностью около 300 мс появляется волна Р300 (цифра «300» в названии как раз и указывает на латентность этой волны [9]).

Классический интерфейс на основе волны Р300 – система для печати, которая представляет собой матрицу из букв, в которой последовательно подсвечиваются строки и столбцы (Farwell, Donchin, 1998). Пользователь отмечает про себя (как правило, счетом) появление строки или столбца, содержащего нужную ему букву. Каждая строчка и столбец подсвечивается по нескольку раз, в быстром темпе, что позволяет усреднить реакции на стимулы и выделить, в каких из них наблюдается Р300. Таким образом, найдя строчку и столбец, на которые удалось выделить такую реакцию, определяют букву, которую хотел напечатать пользователь (рисунок 8).

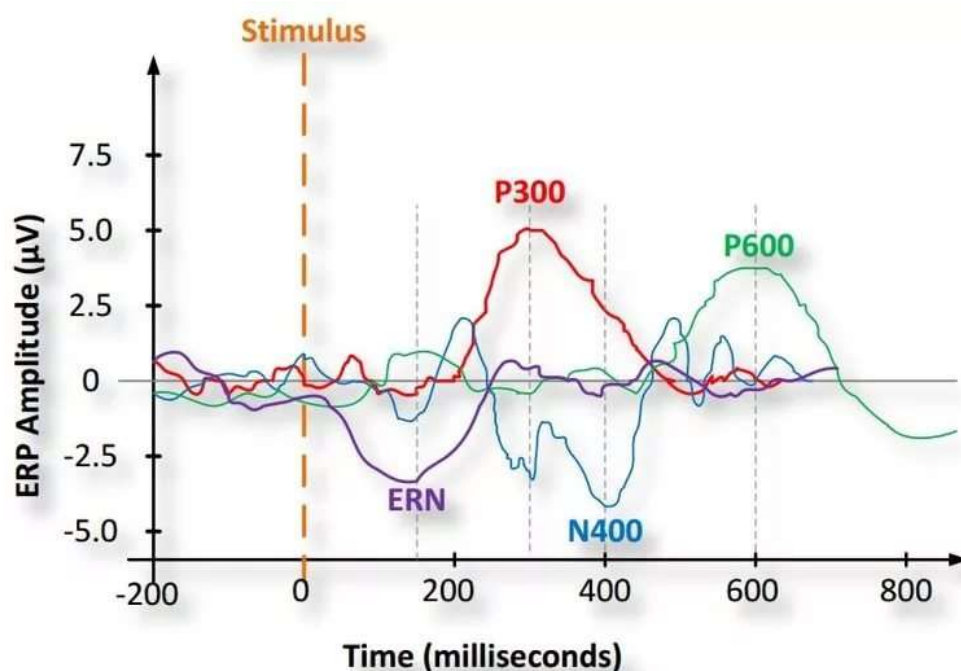


Рисунок 8 – Позитивный компонент зрительного вызванного потенциала с латентностью около 300 мс (Вызванная волна Р300)

Интерфейсы, использующие Р300, на данный момент получили широкое распространение, на их основе разрабатываются устройства для парализованных людей (системы для печати букв, инвалидные коляски), игровые приложения

(Kaplan et al., 2013), а также ИМК-P300 используются в таких необычных исследованиях, как создание виртуальной реальности и рисование силой мысли (Fazel-Rezai et al., 2012) [20][21][22].

1.3 Принцип действия ЭГГ и основы ЭГГ головного мозга человека

ЭЭГ представляет собой сложный колебательный электрический процесс, который может быть зарегистрирован при расположении электродов на мозге или на поверхности скальпа, и является результатом электрической суммации и фильтрации элементарных процессов, протекающих в нейронах головного мозга.

Многочисленные исследования показывают, что электрические потенциалы отдельных нейронов головного мозга связаны тесной и достаточно точной количественной зависимостью с информационными процессами. Для того чтобы нейрон генерировал потенциал действия, передающий сообщение другим нейронам или эффекторным органам, необходимо, чтобы собственное его возбуждение достигло определенной пороговой величины[23].

Уровень возбуждения нейрона определяется суммой возбуждающих и тормозных воздействий, оказываемых на него в данный момент через синапсы. Если сумма возбуждающих воздействий больше суммы тормозных на величину, превышающую пороговый уровень, нейрон генерирует нервный импульс, распространяющийся затем по аксону. Описанным тормозным и возбуждающим процессам в нейроне и его отростках соответствуют определенной формы электрические потенциалы.

Как показано выше, электрическая активность отдельных нервных клеток отражает их функциональную активность по переработке и передаче информации. Отсюда можно сделать заключение, что суммарная ЭЭГ также в преформированном виде отражает функциональную активность, но уже не отдельных нервных клеток, а их громадных популяций, т.е., иначе говоря, функциональную активность мозга. Это положение, получившее

многочисленные неоспоримые доказательства, представляется исключительно важным для анализа ЭЭГ, поскольку дает ключ к пониманию того, какие системы мозга определяют внешний вид и внутреннюю организацию ЭЭГ [24][25]. (рисунок 9)



Рисунок 9 – Основные отделы головного мозга человека

1.4 Анализ требования к выбору аппаратных составляющих для проекта

Для обеспечения стабильной работы комплекса необходимо выбрать соответствующую аппаратную часть. Основная плата, выполняющая роль вычислительного хаба, который принимает данные с ЭЭГ, подключена через последовательный порт. А после эти данные необходимо загрузить в нейронную сеть для обучения, также хаб должен иметь доступ в сеть для связи с облачным сервисом. Из вышеперечисленного можно сделать вывод, что вычислительный хаб должен быть достаточно мощным для обработки достачно большого объема данных и иметь разные интерфейсы для соединения и связи. Поэтому к вычислительному хабу предъявляются такие требования [26]:

- иметь сетевой интерфейс, такой как RJ45 или wifi;
- собственная ОС;
- возможность компиляции программ непосредственно на плате;

- интерфейс USB;
- невысокая потребляемая мощность.

1.4.1 Одноплатный компьютер Raspberry Pi 3

В настоящее время существует достаточно много плат на основе микроконтроллеров и микропроцессоров, отвечающих данным требованиям. Одним из таких является одноплатный компьютер Raspberry Pi 3. Он построен на базе процессоров архитектуры ARM, имеет 1ГБ ОЗУ, 4 USB порта, Ethernet и 40 пинов GPIO для подключения плат расширения и датчиков и при этом имеет сравнительно низкую цену (35\$) по сравнению с аналогами. На рисунке 10 показана структура одноплатного компьютера Raspberry Pi model B. На нём создаются проекты любой сложности от простого включения светодиода до систем умного дома с возможностью удалённого управления [27].

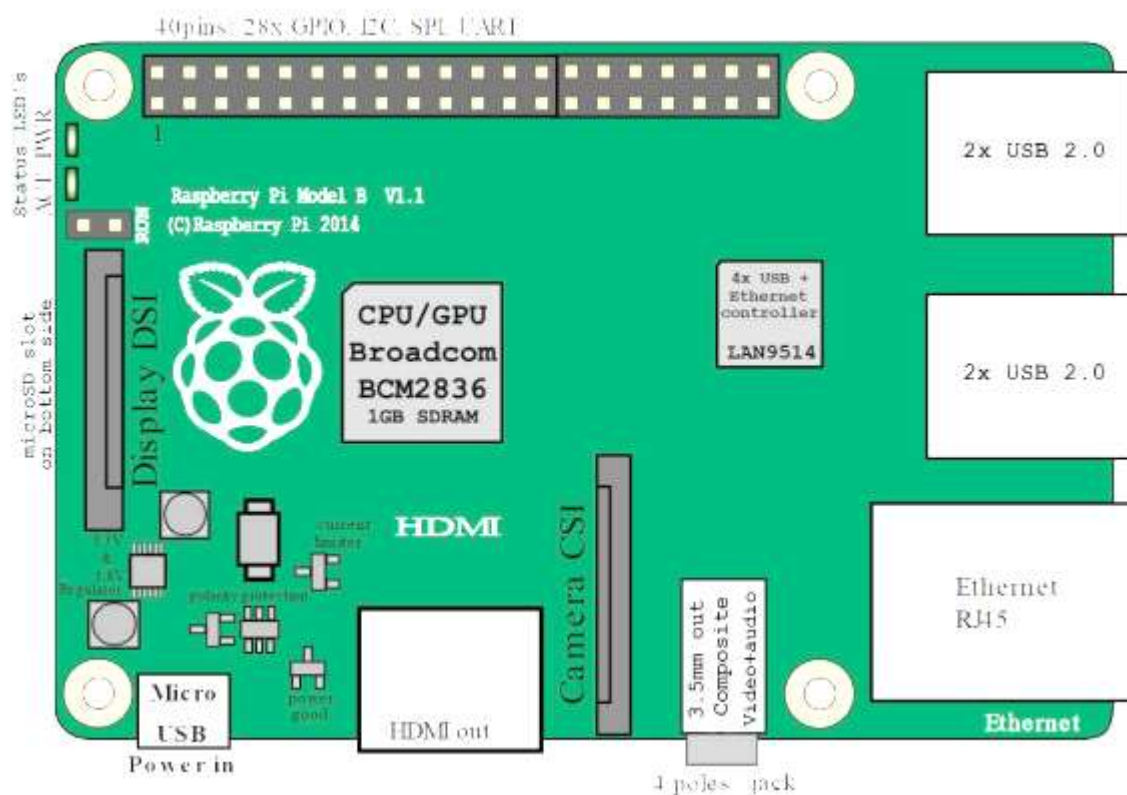


Рисунок 10 – Структура одноплатного компьютера Raspberry Pi model B

1.4.2 Восьмиканальный электроэнцефалограф ИНЭУМ им И.С.Брука

Электроэнцефалограф производства компании ИНЭУМ им.И.С.Брука (рисунок 11) представляют собой многоканальные регистрирующие устройства, объединяющие 8 идентичных усилительно-регистрирующих блоков (каналов) (рисунок 12), позволяющих таким образом регистрировать одномоментно электрическую активность от соответствующего числа пар электродов, установленных на голове обследуемого [28].



Рисунок 11 – Электроэнцефалограф ИНЭУМ им.И.С.Брука



Рисунок 12 – Сенсоры ЭЭГ

Электроэнцефалограф ООО ИНЭУМ им.И.С.Брука цифрового типа с сухими электродами преобразуют ЭЭГ в цифровую форму и вводят ее в микроконтроллер STM32, который управляет непрерывным процессом регистрации ЭЭГ, одновременно записываемым в память компьютера.

Микроконтроллер STM32 реализует систему генерации управления исполнительными механизмами и потоковую передачу данных по протоколу MODICON MODBUS RTU. Протокол реализован на физических линиях интерфейса RS232 через микросхему FTDI, транслирующую пакеты RS232 в USB. Таким образом, прием и передача пакетов ЭЭГ может осуществляться по интерфейсу USB [29][30].

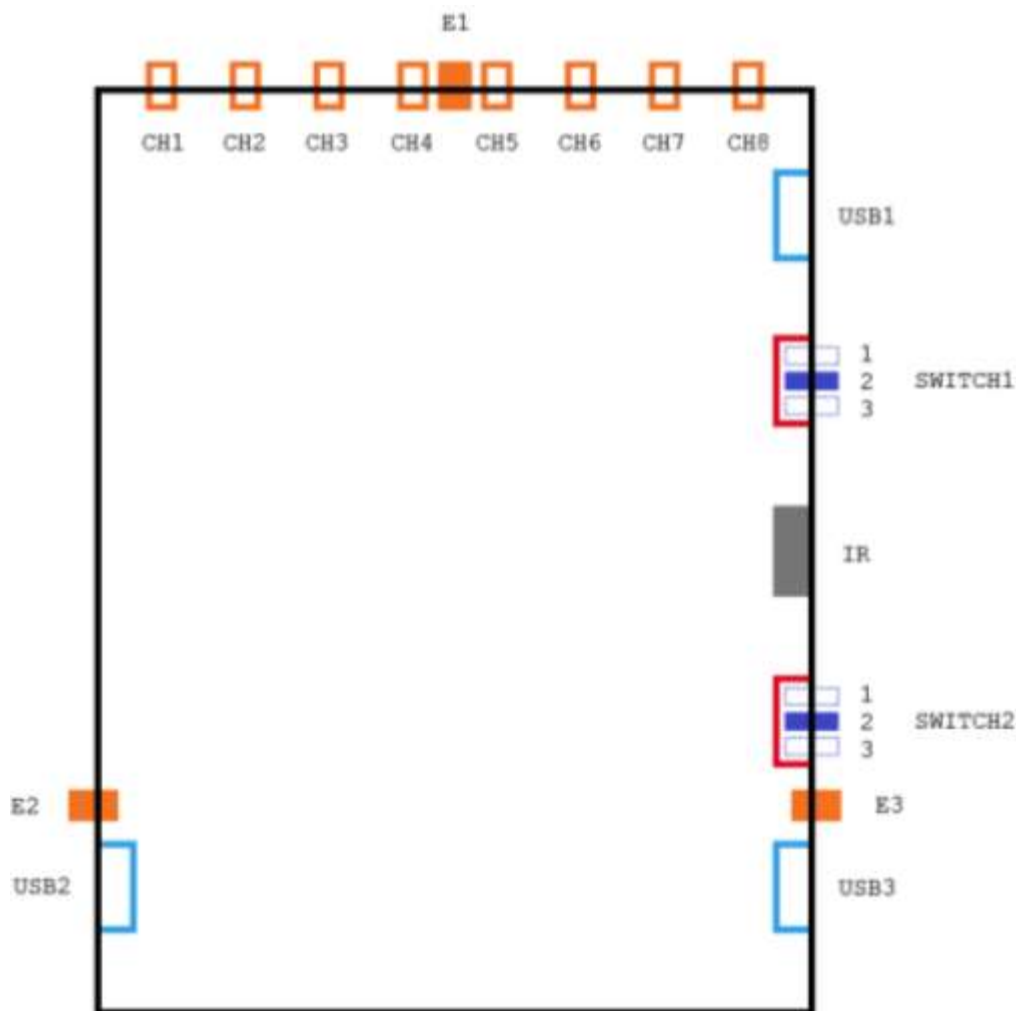


Рисунок 13 – Интерфейсы ЭЭГ

Интерфейсы Электроэнцефалографа представлены на рисунке 13.

Приведем описание и назначение портов и переключателей. Назначение MMCX портов: - CH1–CH8 – подключение электродов;

E1 – RLD (DRIVE);

E2 – RLD (REF–LEFT);

E3 – RLD (REF–RIGHT).

Назначение USB портов:

USB1 – прошивка микроконтроллера;

USB2 – питание устройства;

USB3 – чтение/запись данных устройства.

Инфракрасный порт IR – для управления внешними устройствами.

Назначение переключателей:

SWITCH1 – переключает режимы работы устройств (STM32F429 или BLUENRG-1). Положения переключателя: 1 – рабочий режим; 2 – сброс устройства; 3 – режим программирования (boot);

SWITCH2 – коммутирует устройства, для которых задаются режимы работы переключателем SWITCH1. Положения переключателя: 1–2 – выбор микроконтроллера STM32F429; 3 – выбор микроконтроллера BLUENRG-1.

1.5 Постановка задачи на проектирование ВСУЭУ

Интернет вещей и технология в области ИМК сейчас становятся все более популярным. Чаще всего понятие интернета вещей неразрывно связано с чем-то умным: умные дома, умный транспорт, умные предприятия и т. д. Можно сказать, что IoT – это проводная или беспроводная сеть, соединяющая устройства, которые имеют автономное обеспечение, управляются интеллектуальными системами, снабженными высокоуровневой операционной системой, автономно подключены к Интернету, могут исполнять собственные или облачные приложения и анализировать собираемые данные. Кроме того, они обладают способностью захватывать, анализировать и передавать (принимать) данные от других систем.

Направление человеко – компьютерного взаимодействия в современном мире значительно расширилось и включает в себя как уже привычные, так и

весьма экзотические примеры. Одним из таких проявлений является интерфейс мозг–компьютер, или ИМК, созданный для обмена информацией между мозгом и электронным устройством (например, компьютером). В настоящее время существует множество различных способов и областей применения интерфейса мозг–компьютер. Принцип работы заключается в распознавании активности областей головного мозга.

Из вышесказанного можно сделать вывод, что интернет вещей IoT и ИМК на сегодняшний момент является самыми актуальными направлениями в развитии технологии.

Вывод к исследовательской части

В ходе проведения исследовательской работы проанализировано принципы работы различных облачных систем и различные типы системы ИМК на основе вызванной волны P300. Также проанализирована аппаратная часть проекта, в результате чего принято решение использовать в качестве вычислительного хаба одноплатный компьютер, а в качестве устройства для сбора мозговой волны будем использовать ЭЭГ.

В результате проведенной исследовательской работы выбрана облачная платформа Bluemix, а в качестве интерфейса для управления выбираем неинвазивные ИМК первого типа. А со стороны аппаратной части будем использовать одноплатный компьютер raspberry pi 3 в качестве вычислительного хаба и восьмиканальный электроэнцефалограф ИНЭУМ им И.С.Брука.

Разрабатываемая система будет представлять из себя программно-аппаратный продукт и представляет собой архитектуру распределённой вычислительной системы для управления электронными устройствами с помощью ИМК.

2 Разработка программная и аппаратная часть проекта. Описание архитектуры и алгоритма работы отдельных подсистем (конструкторская часть)

2.1 Выбор архитектуры системы

Исходя из требований, описанных в ТЗ, в ВСУЭУ пользователи системы подразделяются на две категории: обычные пользователи и разработчики. На рисунке 14 представлена диаграмма вариантов использования, на которой показаны основные возможности системы по отношению к пользователям каждой категории.

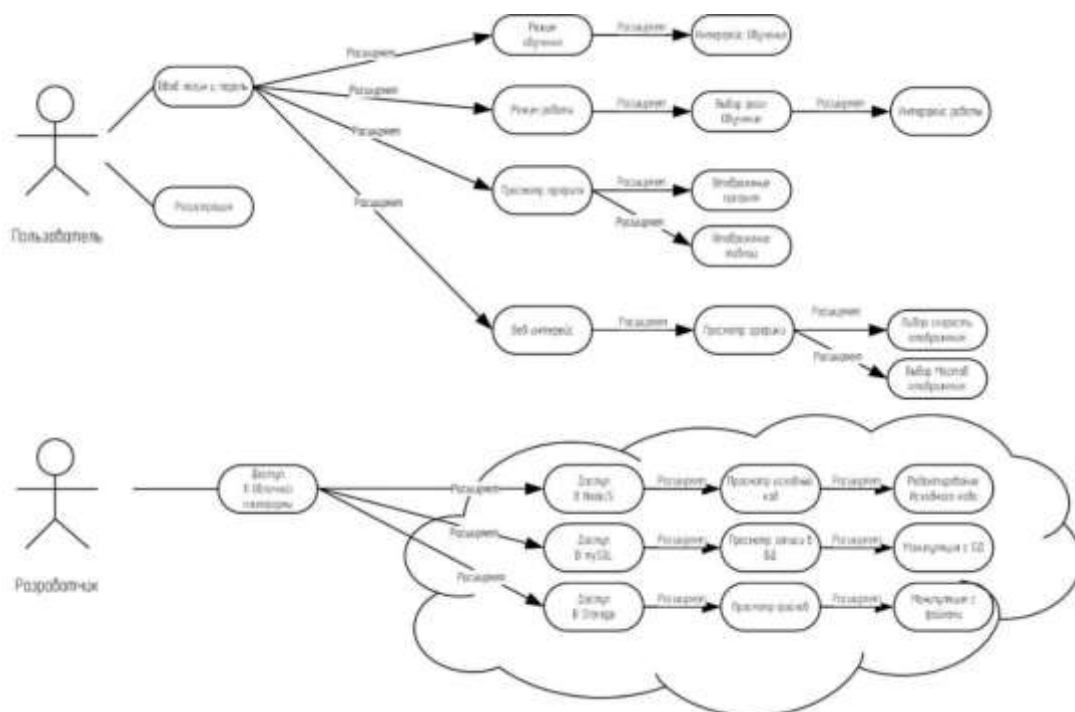


Рисунок 14 – Диаграмма вариантов использования системы

Так как для разработки ВСУЭУ применяется облачная платформа, в которую все клиентские устройства передают данные через Интернет для аналитической обработки, то целесообразно использовать клиент - серверную архитектуру. Это означает, что пользователи используют веб-браузер для доступа к ресурсу, на котором выполняется обработка на сервере. При этом в случае обновления системы, достаточно обновить её только на сервере. Также база данных может располагаться там же, что обеспечит консистентность данных для всех пользователей системы. Ещё одним преимуществом данного

подхода является то, что клиент не нуждается в дополнительном оборудовании и ПО. Все, что ему необходимо – это наличие Интернет соединения и веб-браузера. Таким образом, вся инфраструктура системы должна быть настроена только на сервере. В частности, компиляторы языков программирования, СУБД, веб-сервер и др. В случае клиент – серверного приложения для администрирования всей системы достаточно одного человека, который выполняет роль разработчика и осуществляет поддержку системы. В частности, ему необходимо проводить мониторинг состояния системы, восстанавливать систему в случае сбоев, а также исправлять имеющиеся в ней ошибки. Таким образом, основной задачей данного раздела является описание процесса разработки веб-приложения, которое содержит в себе базу данных, аналитическую обработку данных и разделение прав между разработчиком и пользователем, а также описание проектирования и сбора аппаратной части системы.

Описание диаграммы вариантов использования для пользователя представлено в таблицах 1 – 10.

Таблица 1 – Описание для варианта использования программы в режиме «Обучения»

Название варианта	Программа в режиме обучения
Цель	Визуализировать интерфейс обучения Сбор данных с ЭЭГ Обучать нейронная четь
Действующие лица	Пользователь
Краткое описание	Решение задачи предполагает получение данных с ЭЭГ и обучать нейронную сеть на основе собранных данных.
Тип варианта	Основной

Таблица 2 – Типичный ход событий. Режим «Обучение»

Действия пользователя	Отклик Системы
1. Пользователь зашел в программу	2. Система ждет пока все элементы интерфейса прогрузятся.
3. Пользователь вводит Логин и пароль	4. Система проверяет логин и пароль, если проходит, то к шагу 5. Иначе Ошибка
5. Выбор режим обучения	6. Появляется интерфейс обучения
	7. Запуск таймер
	8. Идет сбор данных
	9. Обучение нейронной сети
	10. Загрузка данных в Bluemix

Таблица 3 – Описание для варианта использования программы в режиме «Работы».

Название варианта	Программа в режиме работы
Цель	Расшифровать команды полученные из ЭЭГ
Действующие лица	Пользователь
Краткое описание	Решение задачи предполагает получение данных из ЭЭГ, а после загрузить данные в нейронную сеть для определения команды пользователя
Тип варианта	Основной

Таблица 4 – Типичный ход событий. Режим «Работы»

Действия пользователя	Отклик Системы
1. Пользователь зашел в программу	2. Система ждет пока все элементы интерфейса прогрузятся.
3. Пользователь вводит Логин и пароль	4. Система проверяет логин и пароль, если проходит то к шагу 5. Иначе Ошибка
5. Выбор режим работы	6. Появляется интерфейс в режиме работы 7. Идет процесс сбора данных 8. Распознавание данных 9. Реализация команд

Таблица 5 – Описание для варианта использования программы в режиме «Просмотр профиля»

Название варианта	Программа в режиме просмотр профиля
Цель	Просмотр данных, вводимых при регистрации, а также просмотр сохраненных файлов обучения
Действующие лица	Пользователь
Краткое описание	Просмотр список сохраненных файлов в облачном хранилище
Тип варианта	Основной

Таблица 6 – Типичный ход событий. Режим «Просмотр профиля»

Действия пользователя	Отклик системы
1.Пользователь зашел в программу	1.Система ждет пока все элементы интерфейса прогрузятся.
3.Пользователь вводит Логин и пароль	4.Система проверяет логин и пароль, если проходит то к шагу 5. Иначе Ошибка
5.Выбор режима просмотра профиля	6.Запрос в БД 7.Визуализация данных о профиле на интерфейсе программы

Таблица 7 – Описание для варианта использования программы в режиме «Просмотр графики»

Название варианта	Программа в режиме «Доступ к БД».
Цель	Построение графики и визуализация данных полученных из ЭЭГ в реальном времени
Действующие лица	Пользователь
Краткое описание	В данном режиме пользователь имеет возможность увидеть данные полученные из ЭЭГ в реальном времени в виде графики
Тип варианта	Основной

Таблица 8 – Типичный ход событий. Режим «Просмотр графики».

Действия пользователя	Отклик системы
1.Пользователь заходит на сайт https://expressjs-webpack-dydbt.eu-gb.mybluemix.net/	Система требует получения ссылки websocket для получения данных
3.Ввод ссылки на websocket	4.Получение данных 5.Вычисление и построение графики 6.Визуализация

Таблица 9 – Описание для варианта использования программы в режиме «Регистрация нового пользователя»

Название варианта	Программа в режиме «Регистрация нового пользователя».
Цель	Регистрация нового пользователя
Действующие лица	Пользователь
Краткое описание	В данном режиме пользователь имеет возможность зарегистрировать себя в систему
Тип варианта	Основной

Таблица 10 – Типичный ход событий. Режим «Регистрация нового пользователя».

Действия пользователя	Отклик Системы
1.Пользователь зашел в программу	2.Система ждет пока все элементы интерфейса прогрузятся.
3.Пользователь заходит в режим регистрации	4.Система загружает интерфейс для режима регистрации, отображает
5.Пользователь вводит персональные данные и нажимает кнопку «принимать»	необходимые поля ввода 6.Система получает данные 7.Проверяет на правильность 8.Загружает в БД 9.Возвращает в исходное окно

2.2 Выбор языка и технологии программирования

При разработке любых информационно – технических систем всегда нужно серьезно подходить к вопросу выбора языка программирования для реализации проекта. От правильного выбора языка программирования до корректного использования среды разработки можно достаточно сильно упрощать объем работы, путем подключения различных библиотек. Особенно этот вопрос становится важным при проектировании программно – аппаратного продукта, где часто необходимо объединять несколько языков программирования вместе так, чтобы конечный продукт функционировал правильно.

Все языки программирования можно разделить на две группы: интерпретируемые и компилируемые. Интерпретируемые языки обладают рядом достоинств, основным из которых является их простота и скорость изучения. Однако они намного медленнее, чем компилируемые языки, а также в большинстве из них не существует возможности одновременной работы нескольких потоков, что может быть полезным при написании веб-сервера. Компилируемые языки более сложны, обладают строгим синтаксисом, из-за чего процесс разработки дольше, чем на интерпретируемых языках, зато с их помощью возможно создать более быстродействующую и надёжную систему.

Основным языком для написания ВСУЭУ является JavaScript, Python с применением различными библиотек (например, jQuery, RPIO, Express, Pybrain, Tkinter, Websocket, RCswitch, JSChart, Bootstrap).

Для аппаратной части и для интерфейса были использованы такие языки как: JavaScript, HTML, Python и CSS.

Для одноплатного компьютера Raspberry Pi был выбран язык программирования Python. Этот выбор был основан тем, что язык Python является стандартным в ОС Raspbian. Raspbian представляет собой пересобранное ядро от ОС Linux. Также Python достаточно прост в

использовании. Кроме того, большинство библиотек для Raspberry также написаны на этом языке, что увеличивает масштабируемость систем.

Из сказанного выше можно делать вывод о том, что для Raspberry Pi язык Python является оптимальным, а по быстродействию соответствует требованию к проекту.

В качестве интерфейса для пользователей были принято решение реализовать отображения в виде веб страниц и локальный интерфейс взаимодействия с пользователем.

Веб интерфейс был выбран, так как данный вид интерфейса позволяет людям получить данные удаленно от оборудования. Также он более универсален при масштабировании систем, так как достаточно иметь браузер для доступа к интерфейсу.

Локальный интерфейс взаимодействия с пользователем в данном случае необходим в связи с тем, что наша система требует быстрого отображения данных, а также есть необходимость иметь быструю связь между пользователем и железом.

В качестве языка для написания серверной части, где находятся веб – приложения, был выбран язык JavaScript. На данный момент для многих устройств существуют версии NodeJS, исходя из чего можно сделать вывод, что язык JavaScript является кроссплатформенным. Также NodeJS предоставляется платформой Bluemix как сервис, что позволяет ей выступать в качестве хостинга.

Локальный интерфейс взаимодействия с пользователем будем реализовать с помощью Python, так как он встроен в ОС Raspbian

JavaScript, Python являются объектно-ориентированными языками, но при реализации проекта также был использован функциональный подход, поэтому в качестве технологии программирования был выбран смешанный подход.

Согласно ТЗ, ВСУЭУ состоит из следующих компонентов: подсистема базы данных, подсистема сбора данных с ЭЭГ, подсистема отправки данных,

подсистема работы с запросами для получения данных с БД, подсистема работа с нейронной сетью, подсистема визуализации данных.

На рисунке 15 показано как подсистемы взаимодействуют друг с другом, а также указаны используемые языки программирования.

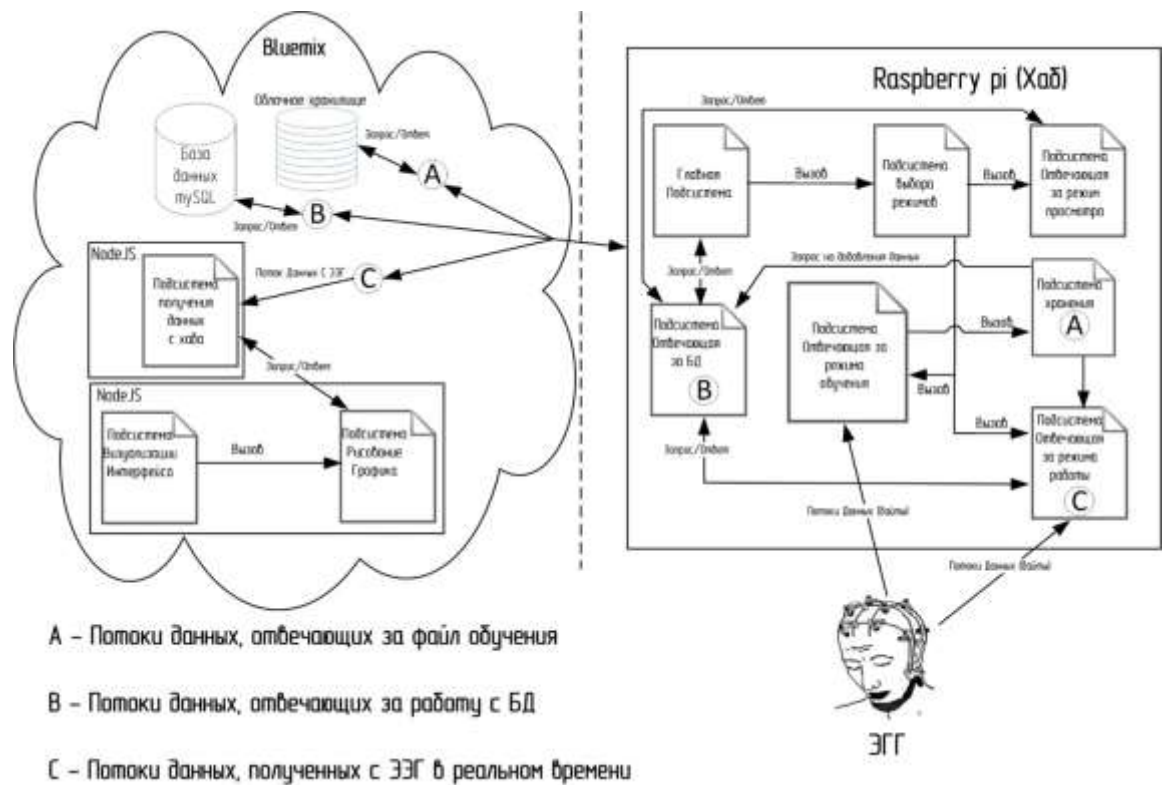


Рисунок 15 – Диаграмма взаимодействия подсистем

2.3 Описание работы подсистем и программные модули

2.3.1 Главная подсистема

Главная подсистема находится непосредственно на вычислительном хабе Raspberry pi. Для запуска проекта необходимо в первую очередь запустить главную подсистему с помощью скрипта `sudo python ex.py` (рисунок 16) в командной строке ОС Raspbian.

```
pi@raspberrypi:~/losa_linh/all together $ sudo python ex.py
```

Рисунок 16 – Скрипт запуска проекта

После запуска проекта появляется окно для аутентификации (рисунок 17), также имеется кнопки для входа в режиме регистрации нового пользователя (рисунок 18).

Основная задача главной подсистемы заключается в отображении интерфейса, а так же аутентификация пользователя. Главная подсистема состоит из следующих модулей:

- модуль отображения интерфейса системы;
- модуль регистрации нового пользователя;
- модуль хеширования данных;
- модуль аутентификации;
- модуль проверки вводимых данных.

На рисунке 20 изображена диаграмма взаимодействия модулей в главной подсистеме.

В главной подсистеме были использованы такие библиотеки:

- MySQLdb – Данная библиотека позволяет подключиться к базе данных, которая находится на облачной платформе Bluemix;
- re – Данная библиотека позволяет использовать регулярное выражение для проверки вводимых данных от пользователя при регистрации;
- Tkinter – Данная библиотека позволяет реализовать локальный интерфейс взаимодействия с пользователем;
- string – Данная библиотека позволяет расширить возможность разработчика в обработке строк;
- hashlib – Данная библиотека позволяет хешировать пароли пользователя.

Регулярное выражение для проверки правильности вводимой электронной почты `email_regex = re.compile('^(\s)[-a-z0-9_.]+@[(-a-z0-9]+\.)+[a-z]{2,6}(\s|$)').` Также предусмотрена проверка на правильность ввода паролей и проверка несовпадения электронной почты в БД.

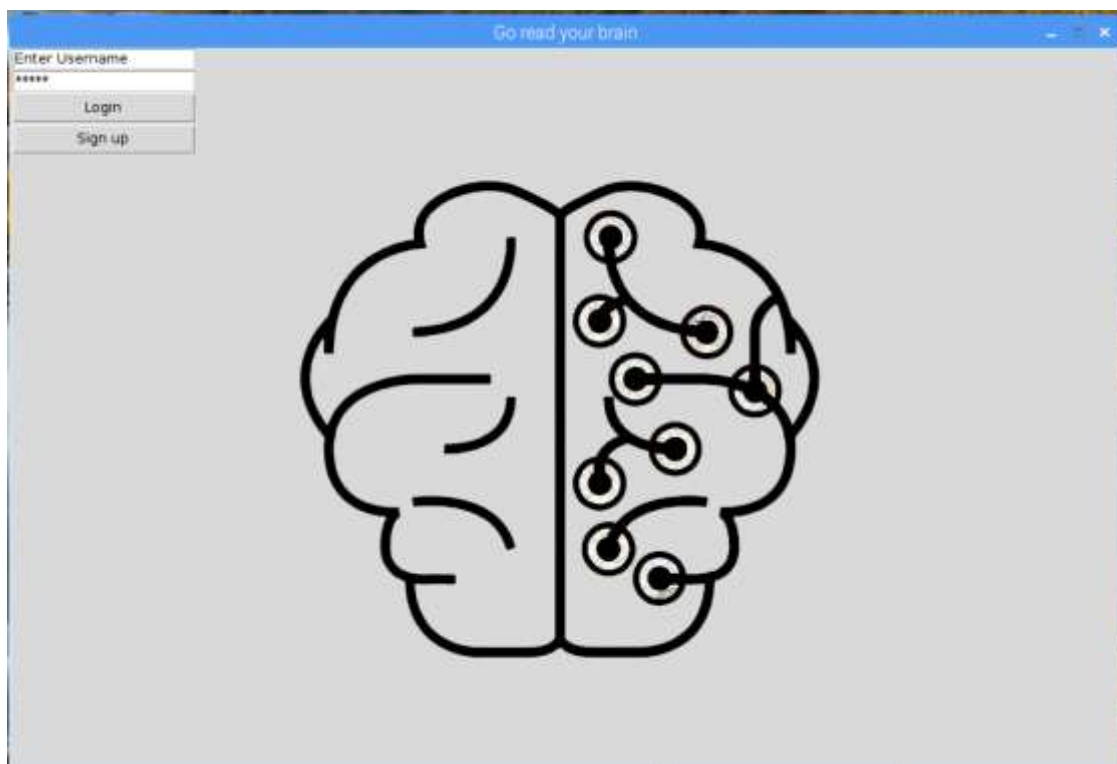


Рисунок 17 – Окно системы после запуска главной подсистемы

Рисунок 18 – Окно регистрации нового пользователя

При регистрации система просит пользователя ввести такие данные, как:

- Фамилия
- Имя
- Адрес
- Адрес электронной почты
- Пароль

Результат регистрации нового пользователя представлен на рисунке 19.

PersonID	LastName	FirstName	Address	Mail	Passwords
171	losa	losa	hanoi	moscow2019@gmail.com	10b6093a9e8ca9f407fc8ccf229727ce

Рисунок 19 – Данные нового пользователя в БД MySQL

Для хеширования пароля пользователя был использован алгоритм MD5, код для хеширования представлен ниже:

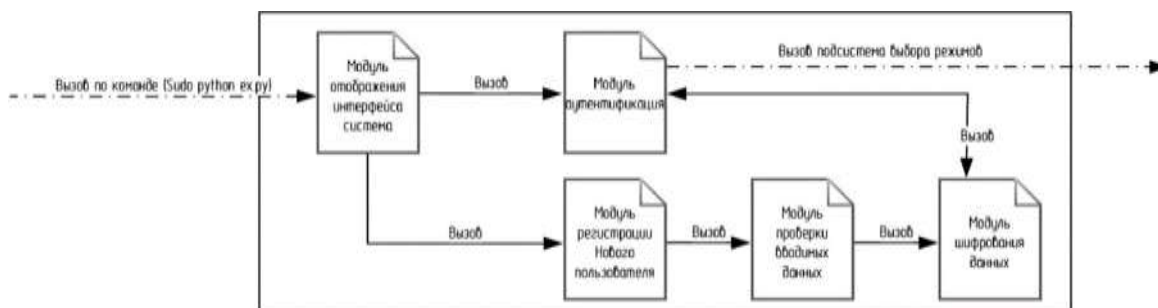


Рисунок 20 – Диаграмма взаимодействия модулей в главной подсистеме

2.3.2 Подсистема выбора режимов

Подсистема выбора режимов находится на вычислительном хабе Raspberry pi. Данная подсистема запускается при ее вызове от главной подсистемы, то есть после прохождении процесса аутентификации.

Основная задача подсистемы выбора режимов заключается в предоставлении выбора пользователю 3 – ёх режимов работы:

- режим работы;
- режим обучения;
- режим просмотра профиля.
- Подсистема состоит из следующих модулей:
- модуль отображения интерфейса в режиме выбора;
- модуль активации режима обучения;
- модуль активации режима работы;
- модуль активации режима просмотра профиля;

На рисунке 22 изображена диаграмма взаимодействия модулей.

В главной подсистеме были испльзованы такие библиотеки:

– Tkinter – Данная библиотека позволяет реализовать локальный интерфейс взаимодействия с пользователем;

– string – Данная библиотека позволяет расширить возможность разработчика в обработке строк;

Интерфейс подсистемы выбора режимов показан на рисунке 22.

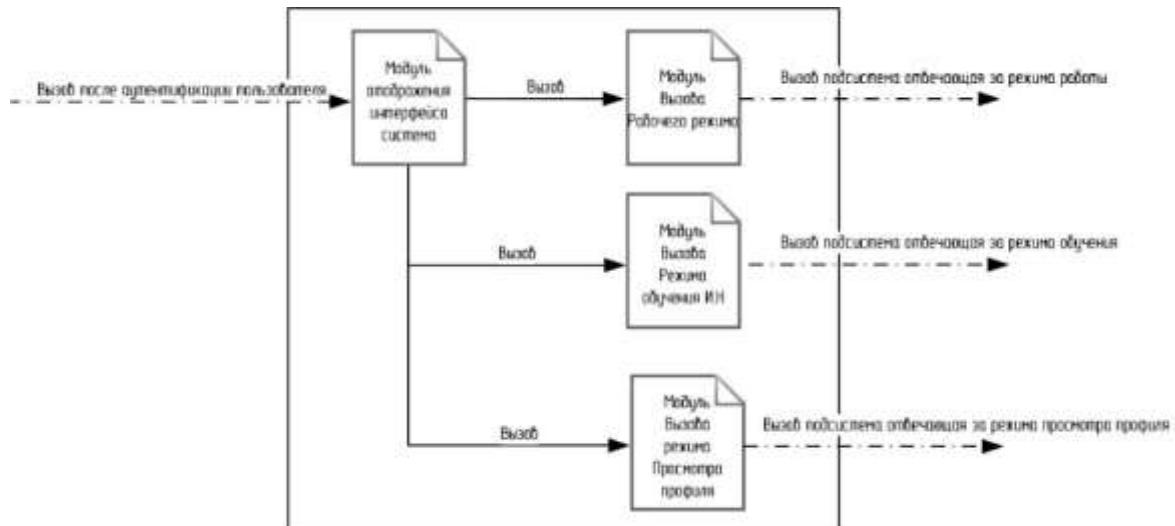


Рисунок 21 – Диаграмма взаимодействия модулей в подсистеме выбора режимов

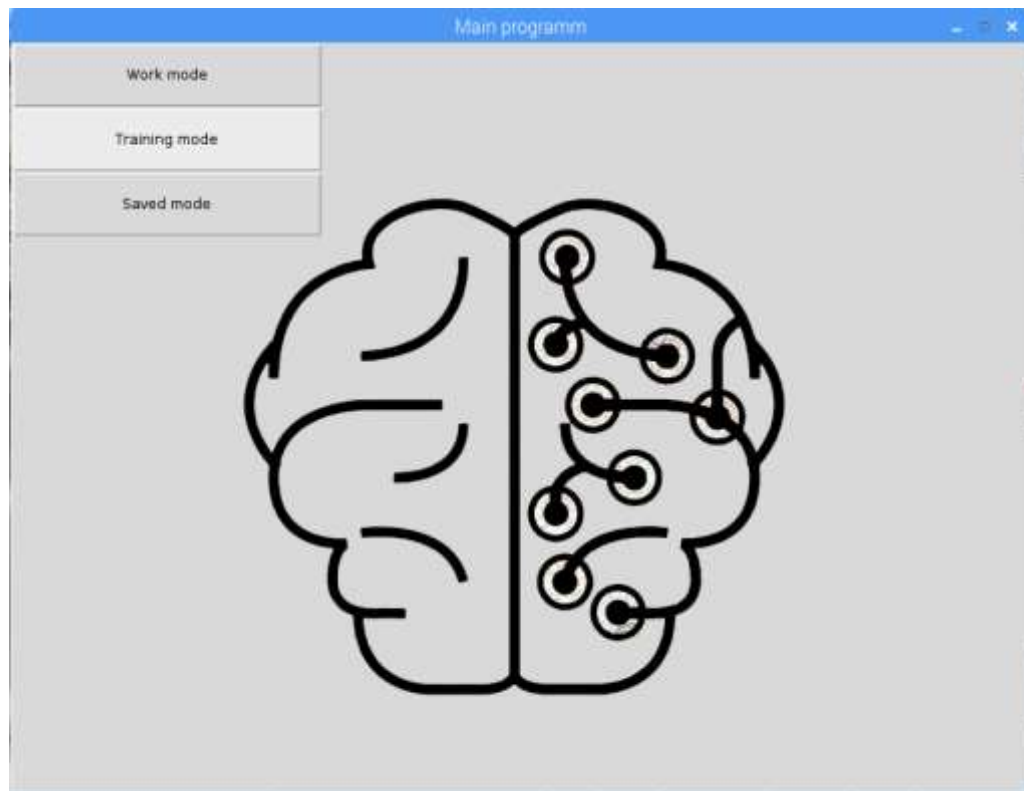


Рисунок 22 – Интерфейс подсистемы выбора режимов

2.3.3 Подсистема отвечающая за режим просмотра профиля

Подсистема отвечающая за режим просмотра профиля находится на вычислительном хабе Raspberry pi. Данная подсистема запускается при ее вызова от подсистемы выбора режимов, то есть после нажатия пользователя на кнопку “Просмотр профиля”.

На рисунке 23 представлен интерфейс в режиме просмотр данных.

Основная задача подсистемы выбора режимов заключается в предоставлении данных о пользователе. То есть такие данные, как:

- Ф. И. О;
- адрес;
- электронный адрес;
- список сохраненных файлов обучения, а также дата и время их создания.

Подсистема состоит из следующих модулей:

- модуль отображения интерфейса в режиме просмотра;
- модуль вызова рабочего режима;
- модуль вызова режима обучения
- модуль вызов режим просмотра профиля

Подсистема отвечающая за режим просмотра профиля тесно связана с подсистемой, отвечающей за базу данных. Так как данные о пользователе находится в БД.

Данные, необходимые в режиме просмотра вызываются с помощью двух модулей из подсистемы работы с БД (Рисунок 24):

- database.get_profile(username) – Модуль получения данных о профиле;
- database.get_data_file(username) – Модуль получения списка названия сохраненных бинарных файлов.

В подсистеме были использованы такие библиотеки:

- Tkinter – Данная библиотека позволяет реализовать локальный интерфейс взаимодействия с пользователем;
- string – Данная библиотека позволяет расширить возможность разработчика в обработке строк;
- tkintertable.Tables – Данная библиотека позволяет использовать таблицы для отображения данных.

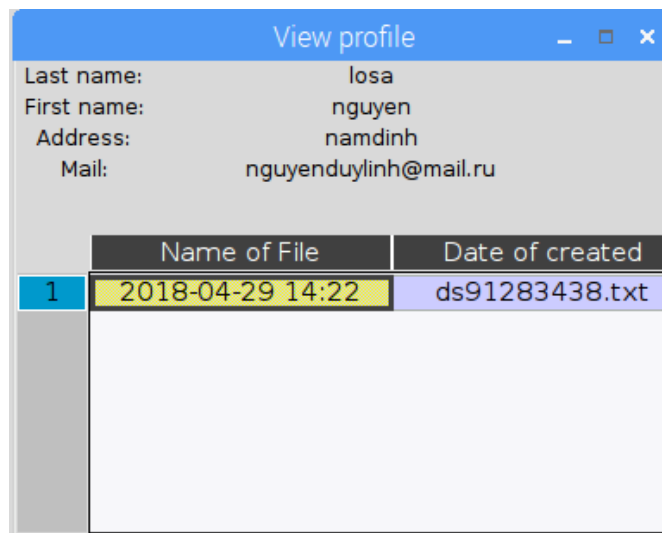


Рисунок 23 – Интерфейс в режиме просмотр профиля

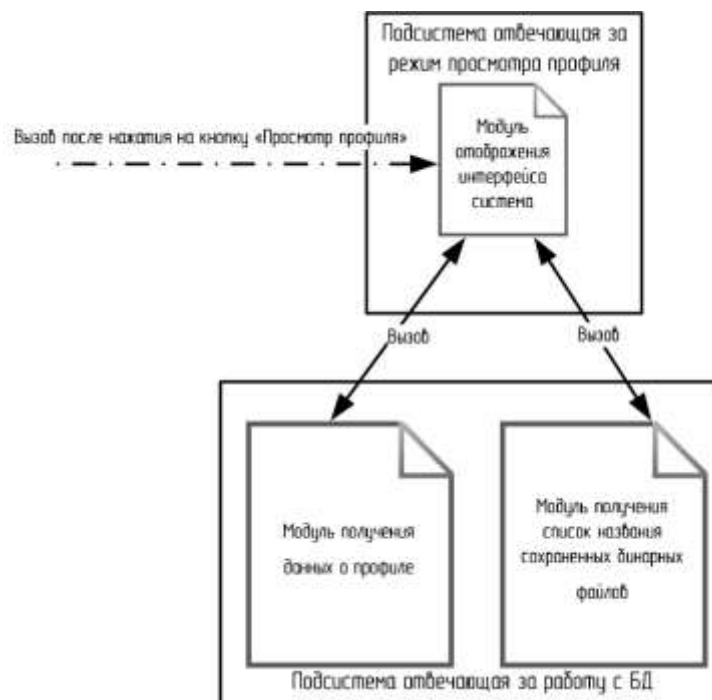
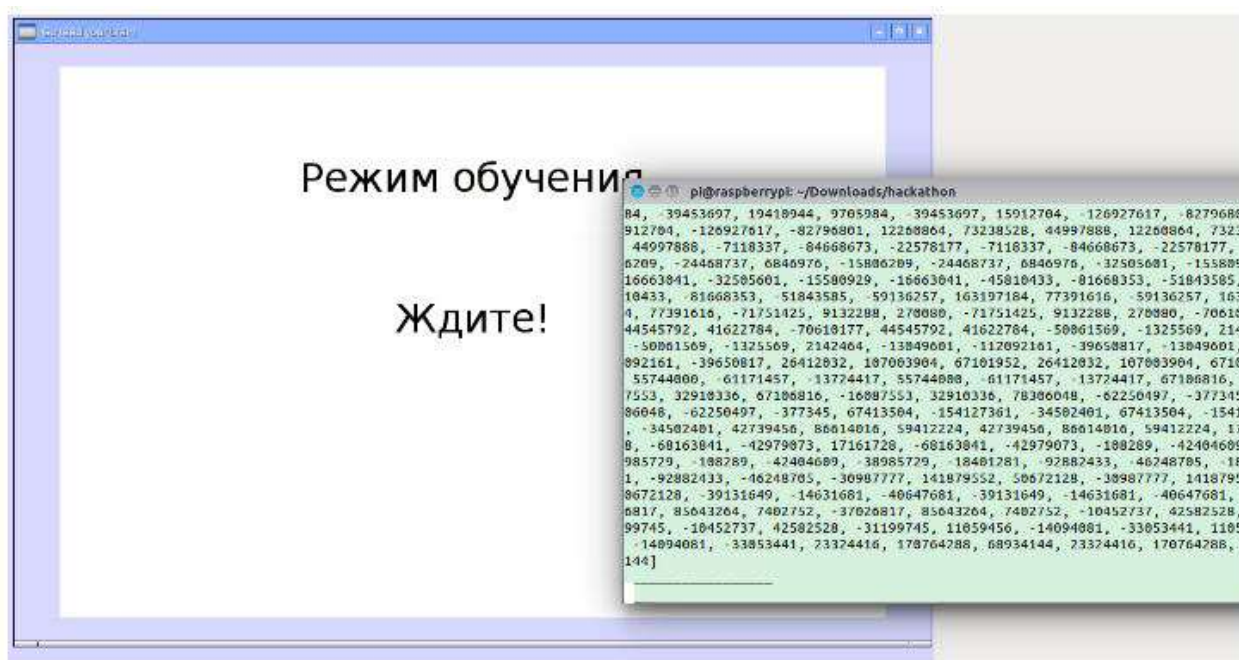


Рисунок 24 – Диаграмма взаимодействия модулей в подсистеме отвечающая за режим просмотра профиля

2.3.4 Подсистема, отвечающая за режим обучения И. Н. С.

Подсистема отвечающая за режим обучения И. Н. С. находится на вычислительном хабе Raspberry pi. Данная подсистема запускается при ее вызове от подсистемы выбора режимов, то есть после нажатия пользователя на кнопку “Запуск режим обучения” (рисунок 25).



Рисунке 25 - Интерфейс в режиме обучения И. Н. С.

Основная задача подсистемы выбора режимов заключается в получения данных от головного мозга человека и на базе полученных данных обучить И. Н. С.

Подсистема должна выполнять такие функции как:

- инициализация ЭЭГ;
- парсинг данных;
- визуализация локального интерфейса для взаимодействия с пользователем;
- обучать И. Н. С.;
- сохранить обученную сеть в виде бинарного файла;
- выгрузить файл в облачное хранилище;
- создать новую запись в БД.

Подсистема состоит из следующих модулей:

- Модуль отображения интерфейса в режиме просмотра;
- Модуль работы с ЭЭГ;
- Модуль обучения И. Н. С;
- Модуль работы с файлами;

Подсистема отвечающая за режим обучения также тесно связана с подсистемой, отвечающей за базу данных и за подсистему, отвечающую за работу с облачным хранилищем. Подробнее о взаимодействиях модулей в подсистеме можно посмотреть на рисунке 26.

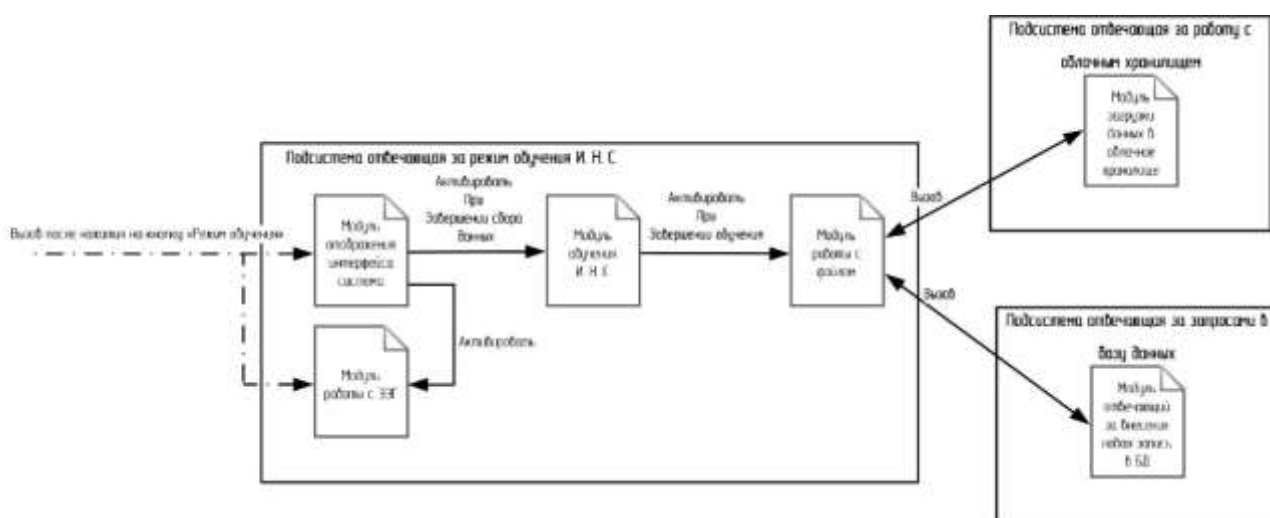


Рисунок 26 – Диаграмма взаимодействия модулей в подсистеме, отвечающей за режим обучения И. Н. С.

В подсистеме, отвечающей за режим обучения И. Н. С. были использованы такие библиотеки:

- deque – Библиотека, которая предоставляет специализированные типы данных, на основе словарей, кортежей, множеств, списков.
- serial – Библиотека, которая инкапсулирует доступ к порту;
- time – Библиотека для работы со временем;
- struct – Библиотека, которая предлагает функции pack() и unpack() для работы с форматами двоичных записей переменной длины;
- Tkinter – Данная библиотека позволяет реализовать локальный интерфейс взаимодействия с пользователем;

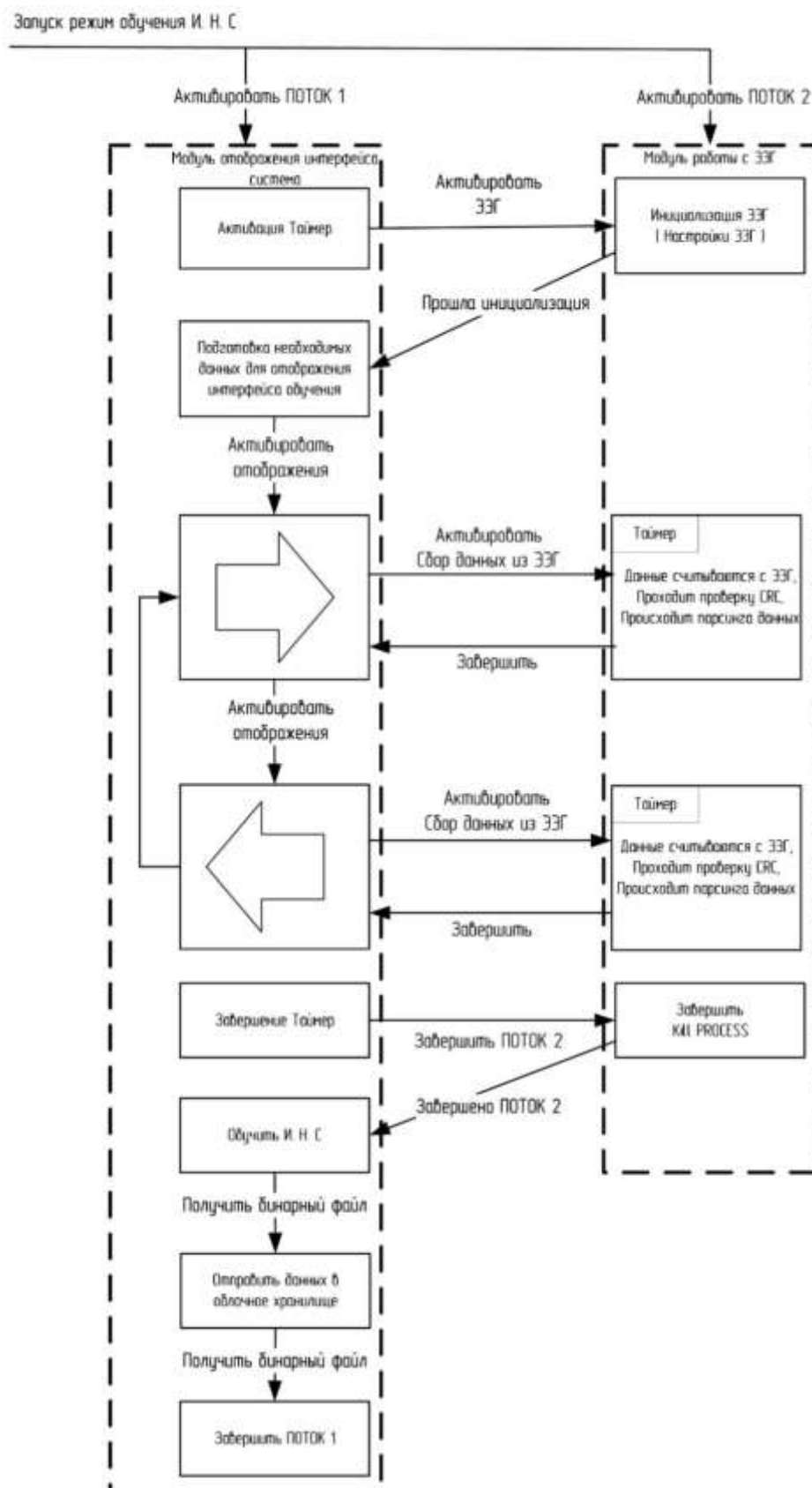
– pybrain - Одна из лучших Python библиотек для изучения и реализации большого количества разнообразных алгоритмов связанных с нейронными сетями. Являет собой удачный пример совмещения компактного синтаксиса Python с хорошей реализацией большого набора различных алгоритмов из области машинного интеллекта;

– Thread – Библиотека, предназначенная для мультипоточного программирования.

– pickle – Библиотека, которая реализует мощный алгоритм сериализации и десериализации объектов Python. "Pickling" - процесс преобразования объекта Python в поток байтов, а "unpickling" - обратная операция, в результате которой поток байтов преобразуется обратно в Python-объект. Так как поток байтов легко можно записать в файл, модуль pickle широко применяется для сохранения и загрузки сложных объектов в Python;

– SupervisedDataSet – Библиотека, предназначенная для создания датасета;

– BackpropTrainer – Библиотека предназначенная для обучения сети.



Рисунк 27 – Диаграмма взаимодействия потоков процесса в подсистеме отвечающая за режим обучения И. Н. С.

При разработки системы было использовано параллельное программирование, где задействованы несколько потоков для решения одной задачи. Так как процесс обучения И. Н. С. требуется параллельная работа между интерфейсом для обучения и процессом сбора данных с ЭЭГ, то для реализации данного требования было использовано параллельное программирование. Первый поток отвечает за отображения интерфейса обучения И. Н. С., а второй отвечает за сбора данных из ЭЭГ. При этом потоки взаимосвязаны между собой для обмена необходимых данных. Подробное описание процесса взаимодействия потоков показано на рисунке 27.

Программный код для обмена данных между потоками:

```
1  import threading
2  from Queue import Queue
3
4  def creator(data, q):
5      """
6      Creates data to be consumed and waits for the consumer
7      to finish processing
8      """
9      print('Creating data and putting it on the queue')
10     evt = threading.Event()
11     q.put((data, evt))
12     print('Waiting for data to be doubled')
13     evt.wait()
14 def my_consumer(q):
15     """
16     Consumes some data and works on it
17     In this case, all it does is double the input
18     """
19     data, evt = q.get()
20     print('data found to be processed: {}'.format(data))
21     evt.set()
22     q.task_done()
23 if __name__ == '__main__':
24     q = Queue()
25     data = True
26     thread_one = threading.Thread(target=creator, args=(data, q))
27     thread_two = threading.Thread(target=my_consumer, args=(q,))
28     thread_one.start()
29     thread_two.start()
30     q.join()
```

2.3.5 Модуль отвечающий за обучения И. Н. С

Нейронной сетью называется математическая модель, реализующая функции искусственного интеллекта путём воспроизведения нервной системы человека. Они используются для решения сложных задач, которые требуют аналитических вычислений, подобных тем, что делает человеческий мозг. К таким задачам относятся, например, классификация, кластеризация, прогнозирование, распознавание и т.д.

Искусственный нейрон представляет собой сумматор входных сигналов, применяющий к полученной взвешенной сумме некоторую простую функцию. Нейрон имеет синапсы - однонаправленные входные связи, соединённые с выходами других нейронов, а также аксон - выходную связь.

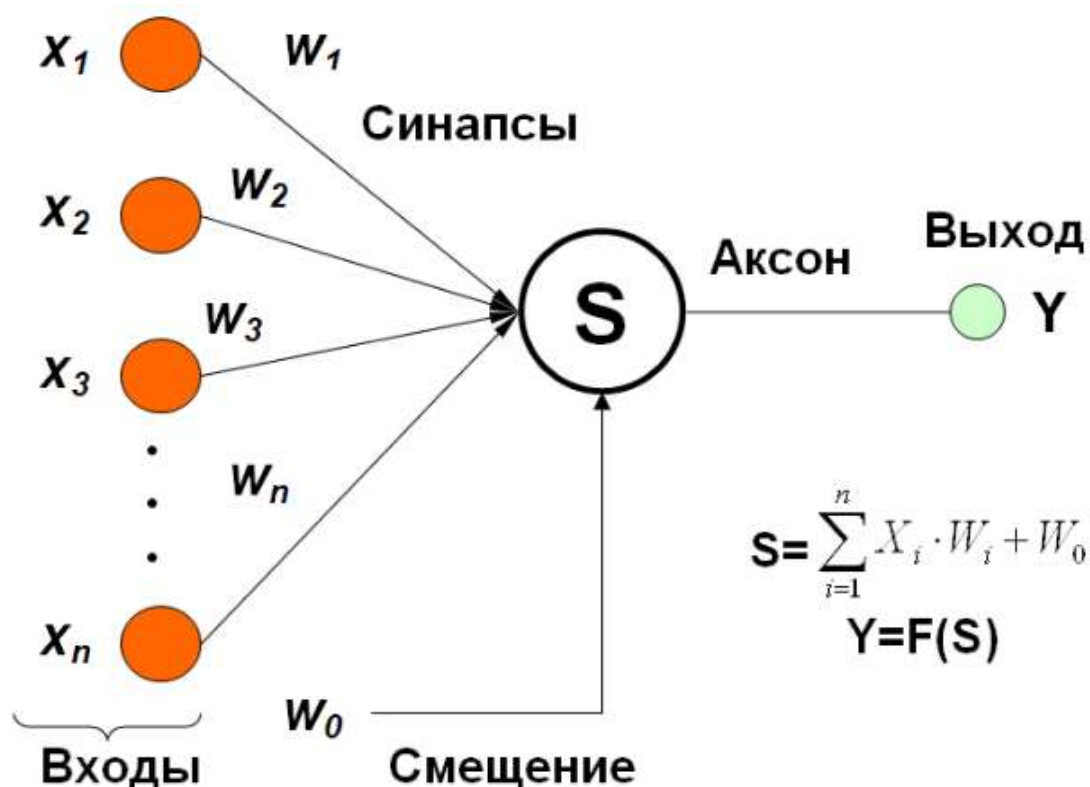


Рисунок 28 – Структурная схема И. Н. С

Текущее состояние нейрона определяется взвешенной суммой его входов (см. схему). Выход нейрона определяется его активационной функцией.

Совокупность нейронов, расположенных на одном уровне в нейронной сети, называется слоем. В общем случае нейронная сеть включает в себя

входной, выходной и промежуточные слои. Нейроны входного и выходного слоёв, как правило, имеют линейную функцию активации и предназначены для приёма и передачи данных. Нейроны промежуточных слоёв - нелинейные; их функцией активации чаще всего является сигмоид (логистическая функция):

$$f(x) = \frac{1}{1 + e^{-ax}}$$

На рисунке показан пример полносвязной нейронной сети, имеющей входной, промежуточный и выходной слой.

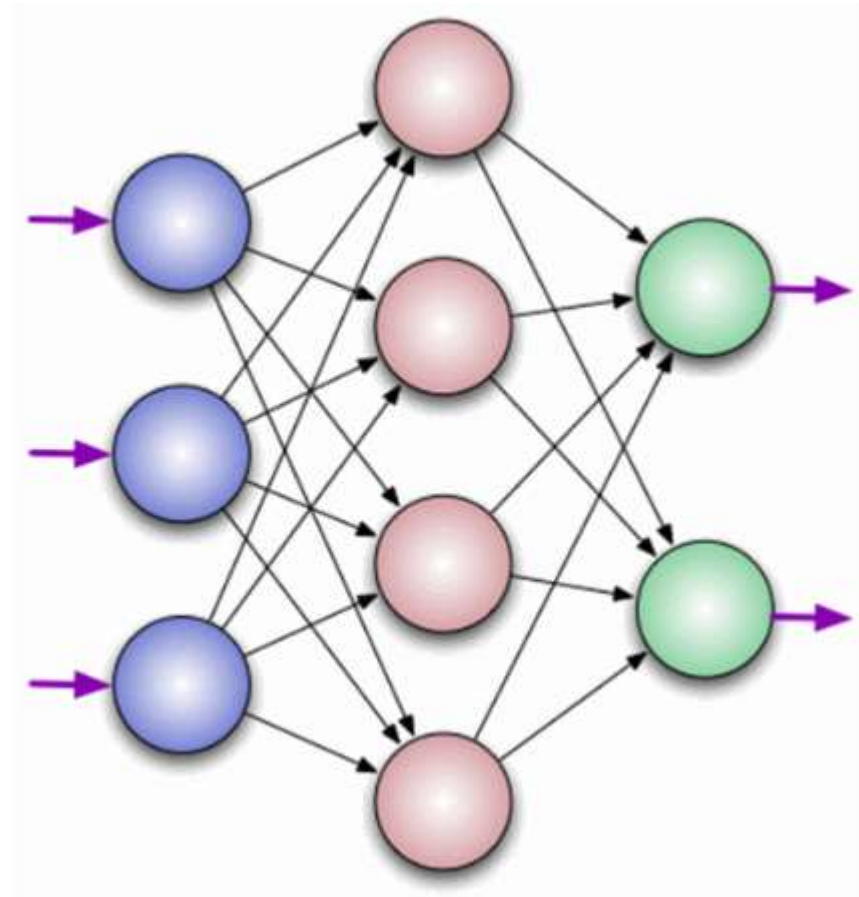


Рисунок 29 - Логическая схема перцептрона

Нейронная сеть обучается. В процессе обучения параметры сети настраиваются в соответствии с обучающими наборами данных, моделирующих среду, в которой будет функционировать сеть. В зависимости от способа подстройки параметров различают обучение с учителем и без учителя.

Обучение с учителем представляет собой предъявление сети выборки обучающих примеров. Каждый образец подаётся на входы сети, проходит обработку и перерабатывается в выходной сигнал, который сравнивается с эталонным значением. Затем в зависимости от степени расхождения реального и идеального результатов изменяются весовые коэффициенты связей внутри сети. Обучение длится до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

При обучении без учителя обучающее множество состоит лишь из входных векторов. Алгоритм обучения подстраивает веса внутри сети так, чтобы предъявление достаточно близких входных векторов давало одинаковые результаты.

2.3.5.1 Выбор топологии нейронной сети

В первую очередь нужно проанализировать задачу, для решения которой создаётся нейронная сеть.

Наша задача - обработать сигналы, поступающие от ЭЭГ, и определить, какой реакции они соответствуют. Сигнал от ЭЭГ представляет собой совокупность из 64 целочисленных значений, которые мы будем называть признаками. На выходе нейронной сети нам нужно получать вектор значений, который будет обозначать тип распознанной реакции. Единичное значение i -го элемента выходного вектора означает, что обнаружена соответствующая ему i -я реакция. Также может быть, что выходной вектор состоит из нулей: это значит, что на вход подан набор признаков, описывающий неизвестную сети реакцию.

Мы будем распознавать реакции двух типов: "Влево" и "Вправо".

За основу возьмём многослойный перцептрон - модель сенсорного нейрона, способного воспринимать, анализировать и реагировать на раздражение. Число слоёв в общем случае определяется требуемой точностью и производительностью вычислений.

Существует теорема о количестве слоёв, согласно которой:

1 Если функция определена на конечном множестве точек, то трехслойный перцептрон способен ее аппроксимировать.

2 Если функция непрерывна и определена на компактной области, то трехслойный перцептрон способен ее аппроксимировать.

3 Остальные функции, которым могут быть обучены нейронные сети, могут быть аппроксимированы трехслойным перцептроном.

На практике может использоваться и большее число слоёв, но с его увеличением возрастает сложность сети, время её обучения и работы и объём требуемых ресурсов. Здесь мы рассмотрим трслыйный перцептрон.

Для лучшей работы сети входные данные нужно нормировать - привести к отрезку $[0, 1]$. Проще всего сделать это по следующей формуле:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Число нейронов во входном слое совпадает с числом входных признаков:
 $i = 64$.

Число нейронов в выходном слое совпадает с числом распознаваемых реакций: $o = 2$.

Число нейронов в промежуточных слоях, как правило, определяется экспериментально, однако существуют эмпирически выведенные рекомендации для 4 – слойного перцептрона:

$$r = \sqrt[3]{\frac{i}{o}} \approx 3.17$$

$$k_1 = o \cdot r^2 \approx 20$$

Было выбрано в качестве функции активации нейронов скрытых слоёв СИГМОИД.

Сеть была обучена по алгоритму обратного распространения ошибки. Этот метод относится к классу обучения с учителем. Краткое разъяснение принципов работы этого алгоритма изложено здесь.

Этот базовый проект можно улучшить, меняя число слоёв, число нейронов в скрытых слоях, активационные функции, способ обучения и многое другое. О том, как это сделать, а также о том, как построить базовый вариант - следующий раздел.

2.3.5.2 Библиотека PyBrain

PyBrain предоставляет инструментарий для работы с нейронными сетями на языке Python. Документация по PyBrain находится здесь.

Пример работы с нейронной сетью с помощью PyBrain.

PyBrain подключается при помощи следующей директивы:

```
import pybrain
```

Самый простой способ создать сеть - воспользоваться функцией `buildNetwork`:

```
net = buildNetwork(64, 20, 6, 2, bias=True, hiddenclass=SigmoidLayer)
```

```
1 # Создание перцептрона
2 # Входные данные: имя сети, список - количество нейронов в каждом слое
3 # Выходные данные: необученная нейронная сеть
4 def constructPerceptron (name, numNeurons):
5     # Создаём сеть
6     net = FeedForwardNetwork(name)
7     # Создаём слои и добавляем их в сеть
8     prevLayer = None
9     newLayer = None
10    for i, val in enumerate(numNeurons):
11        # Если слой входной, он линейный
12        if (i == 0):
13            newLayer = LinearLayer(val, 'input')
14            net.addInputModule(newLayer)
15            prevLayer = newLayer
16        # Если слой выходной, он линейный
17        elif (i == len(numNeurons) - 1):
18            newLayer = LinearLayer(val, 'output')
19            net.addOutputModule(newLayer)
20        # Иначе - слой сигмоидный
21        else:
22            newLayer = SigmoidLayer(val, 'hidden_' + str(i))
23            net.addModule(newLayer)
24        # Если слой не входной, создаём связь между новым и предыдущим слоями
25        if (i > 0):
26            conn = FullConnection(prevLayer, newLayer, 'conn_' + str(i))
27            net.addConnection(conn)
28            prevLayer = newLayer
29    # Готовим сеть к активации, упорядочивая её внутреннюю структуру
30    net.sortModules()
31    # Готово
32    return net
```

Рассмотрим более подробно, что происходит при создании сети:

Далее нужно обучить сеть. Для этого необходимо получить обучающую выборку и преобразовать её к требуемому формату.

Каждая обучающая пара состоит из двух элементов: вектора признаков (64 числа) и выходного вектора-образца (2 числа). Фактически оба эти элемента - кортежи из чисел, например, такие: $i_data = (0.5, 1, 0, 0.2, \dots, 0.7)$; $o_data = (0, 1)$.

После необходимо объединить вектор признаков и вектор результата в единый кортеж и получается обучающую пару: $learnPair = (i_data, o_data)$.

На последок, необходимо объединить все обучающие пары в список: $learnData = [learnPair_1, learnPair_2, \dots, learnPair_N]$.

В таком формате нужно предоставить обучающее множество функции, которая создаст из него dataset PyBrain:

```
1 # Создание обучающего датасета
2 # Входные данные: имя, обучающая выборка
3 # Обучающая выборка должна иметь вид списка с элементами в виде кортежа из двух кортежей:
4 # первый - набор входных признаков, второй - вектор результата
5 # Выходные данные: обучающий датасет
6 def constructDataset (name, learnData):
7     # Вычисляем размерность входных данных
8     dimIn = len(learnData[0][0])
9     dimOut = len (learnData[0][1])
10    ds = SupervisedDataSet(dimIn, dimOut)
11    for d in learnData:
12        ds.addSample(d[0], d[1])
13    return ds
```

Теперь сеть можно обучать. Будем тренировать сеть на заданном наборе в течение одной эпохи, чтобы избежать переобучения. Чтобы следить за качеством обучения, будем также получать оценку ошибки.

Сценарий создания, обучения и использования сети выглядит так:

```
1 # Инициализация сети
2 n = constructPerceptron('perc', [64, 20, 6, 2]) # Создаём перцептрон
3 # Формирование обучающей выборки
4 # Данные для обучения data поступают от ЭЭГ в описанном выше формате - списке обучающих пар
5 ds = constructDataset('data', data)
6 # Запуск 1 полной эпохи обучения
7 (trained_net, err) = trainNetwork (n, ds)
8 # Активируем сеть! На вход подаем i_vec - список из 64 чисел - вектор входных признаков
9 print(trained_net.activate(i_vec))
```

Любой объект библиотеки PyBrain можно распечатать при помощи функции `print()`.

Чтобы изменить число слоёв или число нейронов в слоях сети, нужно подать соответствующий список чисел на вход функции, формирующей сеть.

Чтобы изменить типы скрытых слоёв, достаточно указать другое имя типа слоя при создании сети, например, `TanhLayer` - слой с активационной функцией в виде гиперболического тангенса.

PyBrain предоставляет также обширные возможности для обучения сети. Можно обучать сеть до достижения сходимости, применяя другой метод объекта `trainer`:

2.3.6 Модуль работы с ЭЭГ

Модуль работы с ЭЭГ включает в себе две основные функции:

- Получения данных с ЭЭГ;
- Парсинга данных полученных с ЭЭГ.

Для получения пакета данных из ЭЭГ необходимо подключить библиотеку `serial`, которая инкапсулирует доступ к порту. Далее происходит процесс инициализации ЭЭГ, то есть загрузка настроечных параметров.

Микроконтроллер `STM32F429BIT8` реализует систему генерации команды исполнительному механизму и потоковой передачи данных, для анализа и проведения дополнительных экспериментов, по протоколу `MODICON MODBUS RTU`.

Для получения доступа к отснятым или уже обработанным данным, система управления, роль которой выполняет драйвер, направляет модулю запрос в формате, представленном в таблице 30.

Таблица 10 – Формат запроса к устройству нейроинтерфейса

Адрес Устройства	Доступ к объекту		Начальный адрес				Количество строк		CRC	
	Команда	Область	Ст.байт			Мл.байт	Ст.байт	Мл.байт	Ст.байт	Мл.байт
50	1С	20	00	00	00	00	00	С8	D7	FE

В качестве команды доступны следующие операции:

- 1Ch – чтение кольцевого буфера данных АЦП;
- 20h – смена режима красного светодиода, для инициализации отображения ошибки в программе драйвера;
- 2Fh – установка усиления на каскаде PGA;
- 3Fh – передача тестового массива данных;
- 4Fh – передача массива состояний внутреннего автомата;
- 33h – перезагрузка устройства;
- 2Ch – чтение БПФ по данным содержащимся в буфере.

В ответ на команды чтения, устройство генерирует последовательность данных. Каждые 60 семплов приходит два пакета данных. Первый – данные от электроэнцефалографа (таблица 11). Второй пакет – переменные (таблица 12).

Таблица 11 – Пакет данных от ЭЭГ

	Адрес Устройства	Доступ к объекту		Начальный адрес (1, 62 итд.)			Количество строк (60 строк)		Данные ЭЭГ	CRC	
		Команда	Область	Ст. Байт	2 байта	Мл. Байт	Ст. Байт	Мл. Байт		Ст. Байт	Мл. Байт
	01	1C	20	ADR'			SIZE'		XX	XX	XX
Размер (Байт)	1	1	1	4			2		24 байта x 8 каналов = 32	2	

Всего 523 байт – 1 байт (Адрес устройства) – 2 байта (Доступ к Объекту) – 4 байта (Начальный адрес) – 2 байта (Количество строк) – 2 байта (CRC) = 512 байта данных

Один семпл содержит (байт) = $512 / 16 = 32$ байта

А данный с одного канала = $32 / 8 = 4$ байта

Таблица 12 - Пакет переменных

	Адрес Устройства	Доступ к объекту		Эффективность Распознавания Количество ложных распознаваний Checksum (для отладки)	Количество о строк (60 строк)		@AD R'+ 0 @AD R'+(S IZE' - 1) XX XX	CRC	
		Команда	Область		Ст. байт	Мл. Бай т		Ст. байт	Мл. Бай т
		01	1C	40	SIZE'			XX	XX
Размер (байт)	1	1	1	4	2		16	2	

2.3.6.1 Описание процесса взаимодействия с проборм

В текущей версии прошивки ЭЭГ на базе микроконтроллера STM32F429BIT8 предполагается подключение по последовательному порту с параметрами 460800 8N1.

Для управления используется две основные команды:

/put/memory?address=%d&value=%d&

- запись значения в регистр;

/get/memory?address=%d&

- чтение значения регистра.

Здесь %d - целые числа.

В ответ, после команды чтения, приходит сообщение (пример):

<MEMORY>

<ADDRESS>

78

</ADDRESS>

<VALUE>

128

<VALUE>

</MEMORY>

С номером и записанным значением регистра.

Список регистров (ConfigMem). Из них доступны: 0x00000027 - 0x00000034; 0x00000057 - 0x00000060.

Все, кроме регистров REG_ADC_REG14 и REG_ADC_ORDER, являются конфигурационными для установленного АЦП, перед тем как подать команду трансляции потока нужно их сконфигурировать (пример конфигурации можно посмотреть в matlab-скрипте VVsel, функции pushbuttonApply_Callback, этот скрипт позволяет переводить устройство во все возможные режимы АЦП).

Регистр REG_ADC_REG14 со значением 0x01 включает поток ЭЭГ-сигналов; 0x02, 0x03, 0x04 - не используются; любое другое записанное значение, отличное от приведенных, переводит устройство в режим default - нормальный съем сигналов с электродов с 1-ым коэффициентом усиления (запуск потока семплов). Регистр REG_ADC_ORDER выставляет порядок сигналов в потоке.

Пример команды записи регистра (изменение регистра REG_ADC_REG14):

/put/memory?address=52&value=5&

Код для инициализации ЭЭГ:

```
1 def send_cmd(cmd):  
2     ser.write(cmd.encode())  
3     time.sleep(0.5)
```



```

1  def ser_init():
2      # Open serial
3      ser.baudrate = 460800
4      if ser.isOpen():
5          ser.close()
6      ser.open()
7      # Check serial is opened
8      ser.isOpen()
9      # Send init commands
10     print ("Start device initialization!")
11     send_cmd("/put/memory?address=39&value=214&\0")
12     send_cmd("/put/memory?address=40&value=194&\0")
13     send_cmd("/put/memory?address=41&value=96&\0")
14     send_cmd("/put/memory?address=42&value=0&\0")
15     send_cmd("/put/memory?address=43&value=0&\0")
16     send_cmd("/put/memory?address=44&value=0&\0")
17     send_cmd("/put/memory?address=45&value=0&\0")
18     send_cmd("/put/memory?address=46&value=0&\0")
19     send_cmd("/put/memory?address=47&value=0&\0")
20     send_cmd("/put/memory?address=48&value=112&\0")
21     send_cmd("/put/memory?address=49&value=0&\0")
22     send_cmd("/put/memory?address=50&value=0&\0")
23     send_cmd("/put/memory?address=51&value=0&\0")
24     send_cmd("/put/memory?address=87&value=0x76543210&\0")
25     send_cmd("/put/memory?address=88&value=0x40&\0")
26     send_cmd("/put/memory?address=89&value=0x40&\0")
27     send_cmd("/put/memory?address=90&value=0x40&\0")
28     send_cmd("/put/memory?address=91&value=0x40&\0")
29     send_cmd("/put/memory?address=92&value=0x40&\0")
30     send_cmd("/put/memory?address=93&value=0x40&\0")
31     send_cmd("/put/memory?address=94&value=0x40&\0")
32     send_cmd("/put/memory?address=95&value=0x40&\0")
33     send_cmd("/put/memory?address=52&value=1&\0")
34     print ("УСТРОЙСТВА ИНИЦИАЛИЗИРОВАНО ")
35     time.sleep(3)
36     # Очистка буфер
37     ser.flushInput()
38     ser.flushOutput()
39     ser.flush()

```

2.3.6.2 Описание процесса парсинга данных

По таблице 11 можно увидеть, что значение 1 28 32 (или в шестнадцатеричном значении 01 1С 20) - это заголовок пакета (Рисунок 30).

Первый шаг:

Код программы отвечающий за поиск заголовков пакета:

```

while headerfirstbyte != 1 and headersecondbyte != 28 and headerthirtbyte != 32:
    data_of_byte = data_of_byte[1:]
    data_of_byte += get(1)
    header_nextbyte = changebyte(data_of_byte[2])
    headerfirstbyte = headersecondbyte
    headersecondbyte = headerthirtbyte
    headerthirtbyte = int(header_nextbyte[0])

```

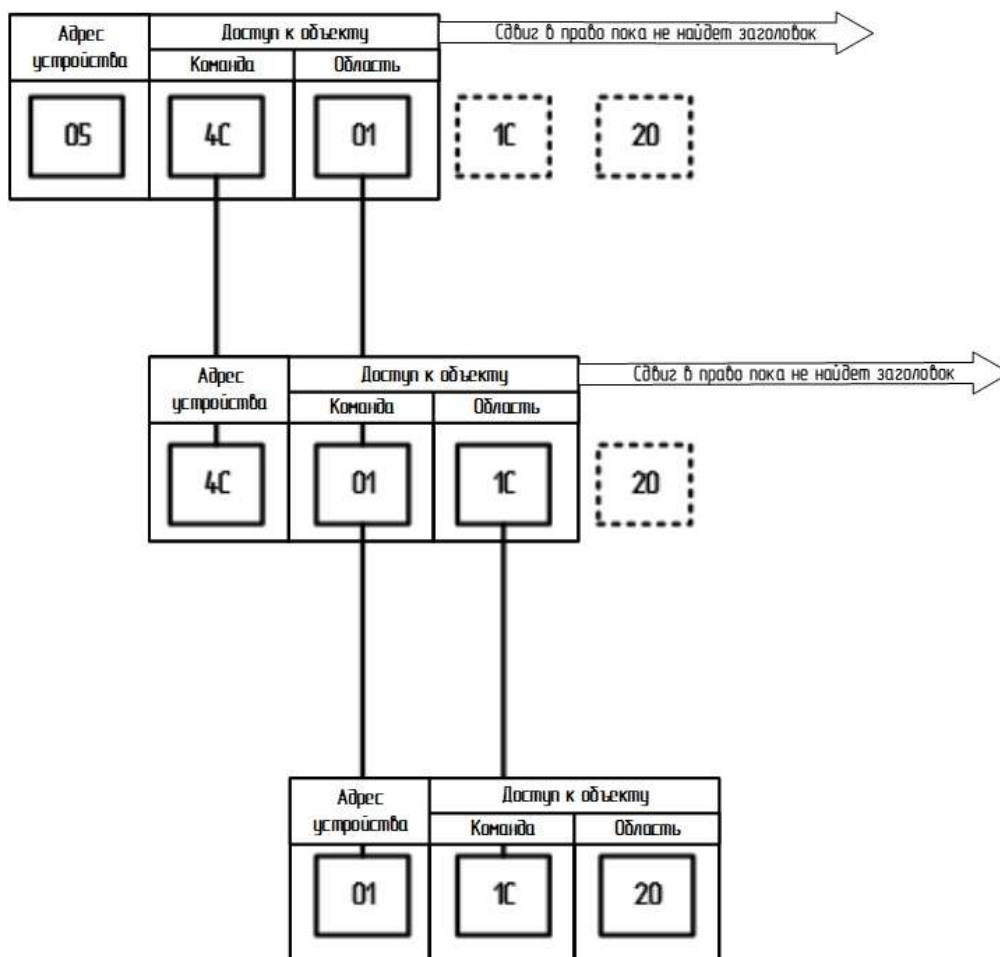


Рисунок 30 – Описание процесса поиска заголовков

Второй шаг:

Далее происходит проверка размера пакета. Пакет должен содержать 521 байт, причем не учитывая 3 байта заголовка. Получается 4 байта (Начальный адрес) + 2 байта (Количество строк) + 2 байта (CRC) + 512 байт данных = 521 байт.

После проверки размера пакета, данные должны пройти проверку на целостность. Для проверки было принято решение использовать циклический избыточный код (CRC), так как он является практическим применением помехоустойчивого кодирования, основанным на определённых математических свойствах циклического кода. После прохождения проверки пакет разбивается на матрицу. Где каждый столбец матрицы это данные от 1 – ого до 8 –ого канала (то есть электрода), а каждая строка это количество раз, когда данные считывали (в одном пакете содержится 16 раз, когда данных считываются с ЭЭГ) (рисунок 31).

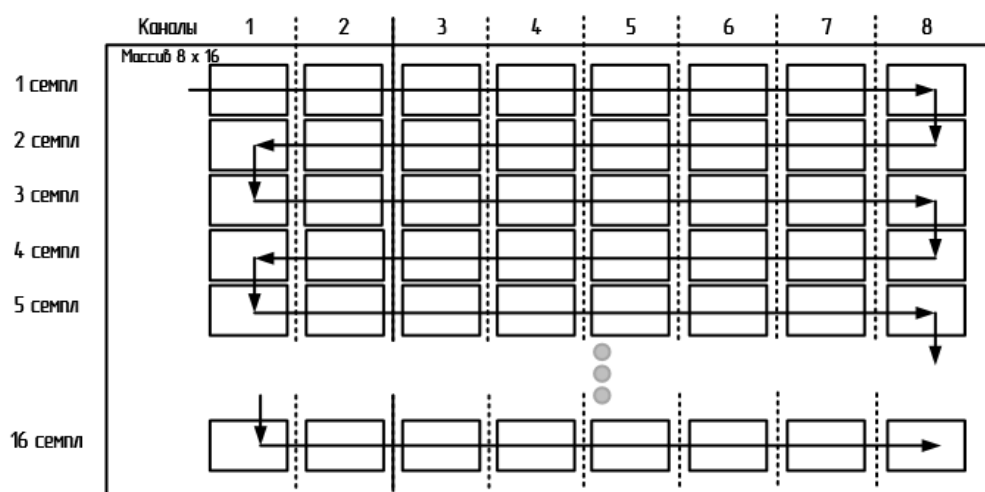


Рисунок 31 – Матрица значений от одного пакета

Третий шаг:

Так как буфер у последовательного порта не так велик для хранения больших данных, а данные, считываемые с ЭЭГ имеют большую скорость (460800 бот), то возникла необходимость хранить данные в ОЗУ. В связи с этим было принято решение использовать список, реализованный как FIFO – буфер для загрузки данных в И. Н. С.

Размер FIFO – буфера (рисунок 32) будет зависеть от количества полученных значений из ЭЭГ за 500 мс.

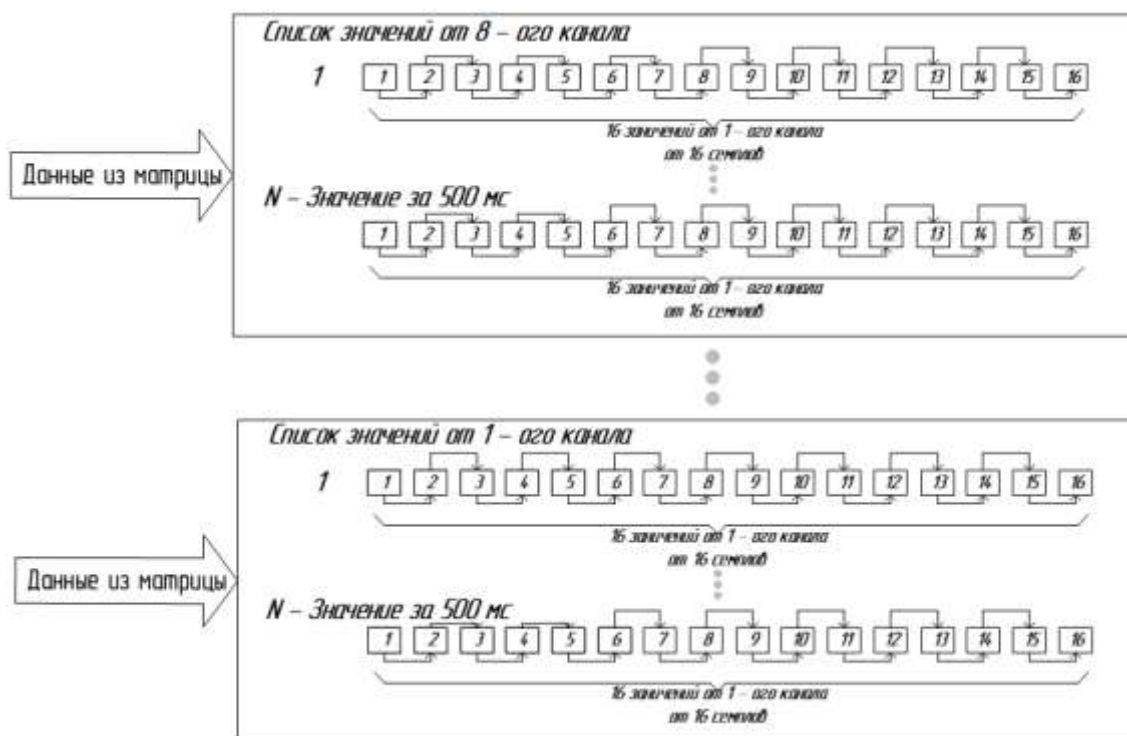


Рисунок 32 – FIFO – буфер от 8 – ми каналов

2.3.7 Модуль отображения интерфейса

Для реализации модуля отображения локального интерфейса пользователя была использована библиотека Tkinter.

Примерный код отображения интерфейса:

```
1 root = Tk.Tk()
2 root.wm_geometry("1000x650+20+40")
3 root.title('Go read your brain') #Заглавление окна
4 background_image=Tk.PhotoImage(file="wait.png") #Отображения фона
5 background_label = Tk.Label(root, image=background_image)
6 background_label.place(x=0, y=0, relwidth=1, relheight=1)
7 root.wm_geometry("1000x650+20+40")
8 root.resizable(False,False)
9 root.title('Go read your brain')
10 root.update()
```

Классический интерфейс на основе волны P300 – система для печати, которая представляет собой матрицу из букв, в которой последовательно подсвечиваются строки и столбцы. В данном проекте вместо строк и столбцов используются знаки. Пользователь отмечает про себя слово при появлении знака, которое соответствует этому знаку. Каждый знак подсвечивается по несколько раз в быстром темпе, что позволяет усреднить реакции на стимулы и выделить, в каких из них наблюдается P300. Таким образом, находя знак, на который удалось выделить такую реакцию, определяют команду, которую хотел пользователь (Рисунок 33 и 34).

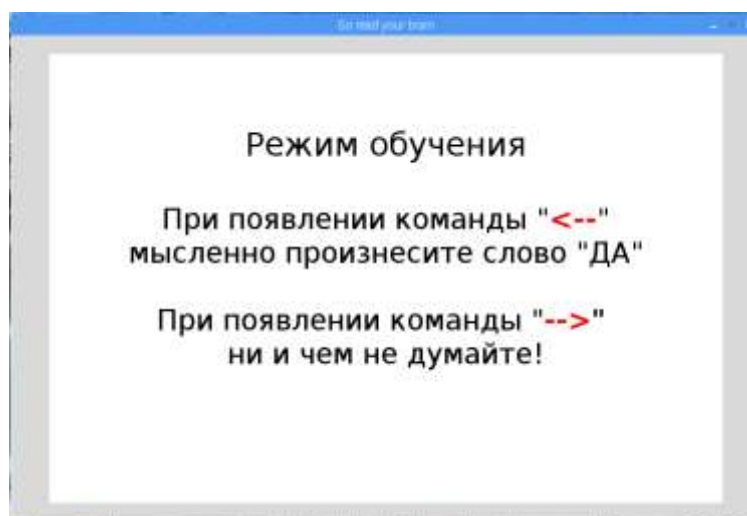


Рисунок 33 – Интерфейс системы в режиме обучения
«Обучения команды влево»

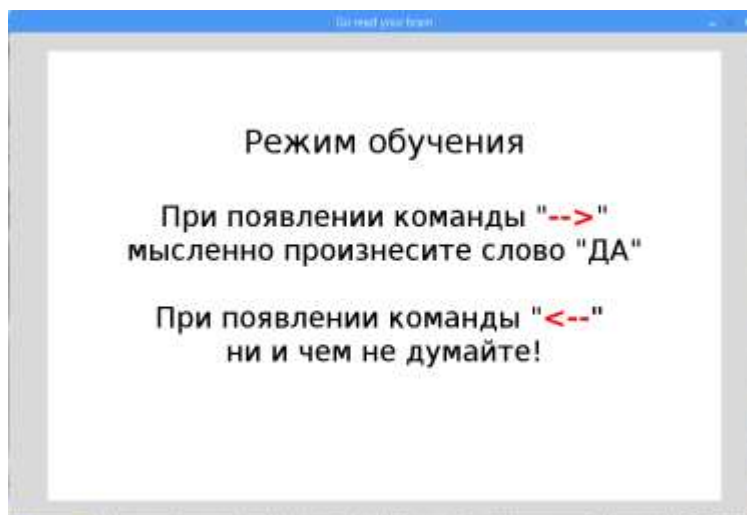


Рисунок 34 – Интерфейс системы в режиме обучения
«Обучения команды вправо»

2.3.8 Модуль работы с файлом

Так как в системе предусмотрена функция сохранения обученной И. Н. С., то необходимо предусмотреть случай, когда пользователь будет иметь возможность выбрать один из нескольких сохраненных файлов с обученной сетью в облачном хранилище. Для этой цели все имена сохраненных файлов содержатся в БД MySQL, которая находится на облачной платформе Bluemix.

Для работы с файлом была использована библиотека pickle.

Примерный код:

```
1 import pickle
2 file_name = 'ds' + str(random.randint(100,99999999)) + '.txt'
3 with open(file_name, "wb") as file:
4     pickle.dump(ds, file)
5 bucket_name = storage.Up_to_storage(username1, file_name)
6 database.input_dataperson(username1, file_name, bucket_name)
```

Подсистема, отвечающая за запросы в базу данных, необходима для модулей работы с файлом потому, что она позволяет сохранить имена всех созданных файлов.

Запрос на добавление новых данных в БД:

```
“INSERT INTO dataofperson (Person_ID ,data_name, bucket_name, date_of_file)
VALUES (%s, %s, %s, %s);”, (str(data[0][0]), data_name, bucket_name, date_now)
```

– Bucket_name – Каталог данных для определенного пользователя в системе;

– Date_now – Дата создания файла;

– Data_name - Имя созданного файла с обученной И. Н. С.

Имя нового файла или каталога генерируется системой автоматически с помощью библиотеки random.

```
file_name = 'ds' + str(random.randint(100,99999999)) + '.txt'
bucket_name = 'bucket' + str(random.randint(100,99999999));
```

DataID	Person_ID	data_name	bucket_name	data_of_file
401	201	ds11151909.txt	bucket17525541	2018-04-30 20:58
411	201	ds80022579.txt	bucket17525541	2018-05-02 14:52
421	201	ds19433640.txt	bucket17525541	2018-05-03 11:36
431	201	ds65200733.txt	bucket17525541	2018-05-04 17:57
441	201	ds56609900.txt	bucket17525541	2018-05-04 19:37
451	201	ds67769083.txt	bucket17525541	2018-05-10 20:13

Рисунок 34 – Содержимое в таблице dataofperson в БД

Подсистема, отвечающая за работу с облачным хранилищем, необходима для модуля работы с файлом потому, что она позволяет выгрузить файл в облачное хранилище на платформе Bluemix.

```
1 cos.upload_file('./' + file_name, bucket_name, file_name)
```

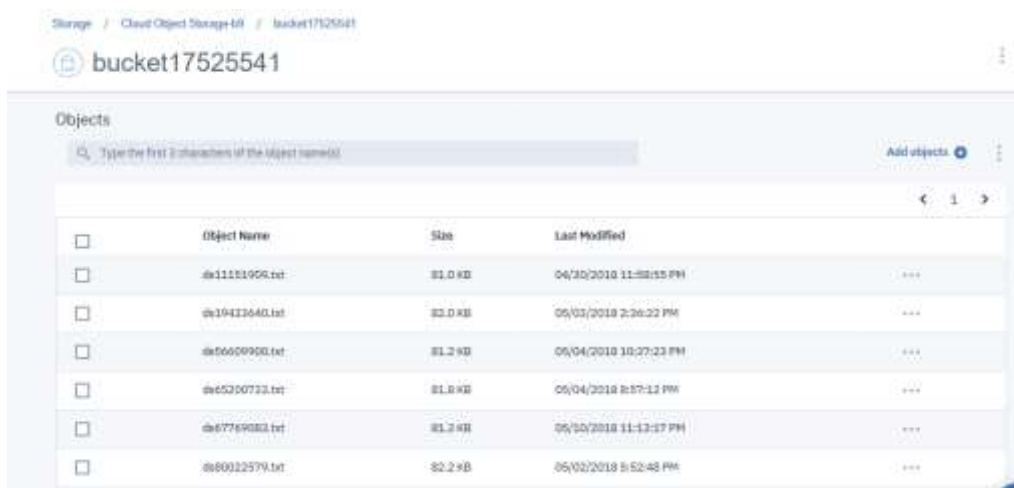


Рисунок 35 – Содержимое в облачном хранилище
Object Storage от компании IBM

2.3.9 Подсистема отвечающая за режим работы

Подсистема отвечающая за режим работы находится на вычислительном хабе Raspberry pi. Данная подсистема запускается при ее вызове от подсистемы выбора режимов, то есть после нажатия пользователя на кнопку “Запуск режима работы” (рисунок 36).

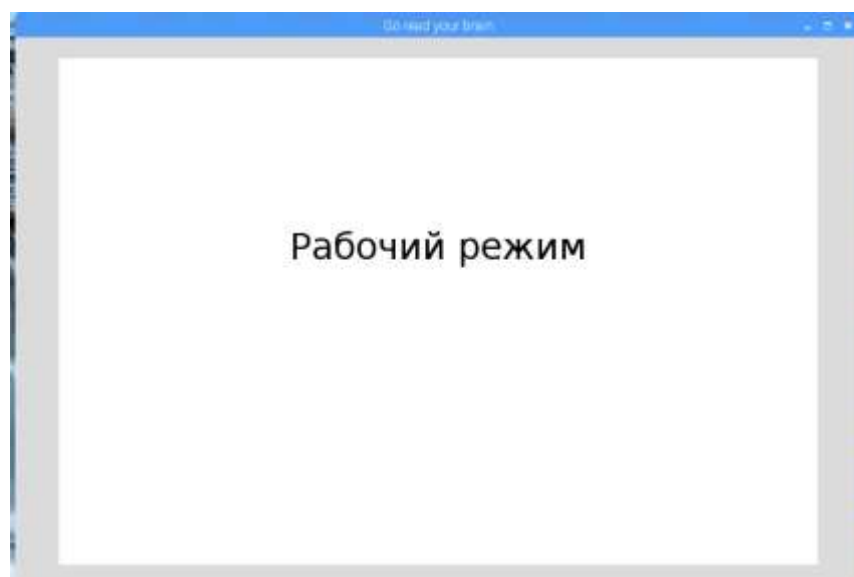


Рисунок 36 - Интерфейс в режиме работы

Основная задача подсистемы выбора режимов заключается в получении данных от головного мозга человека и на базе полученных данных появится возможность управлять робототизированной рукой.

Подсистема должна выполнять такие функции как:

- инициализация ЭЭГ;
- парсинг данных;
- визуализация локального интерфейса для взаимодействия с пользователем;
- распознать и рашифровывать сигналы из головного мозга на базе обученной И. Н. С.;
- отправить данные снятые с ЭЭГ в облачную платформу Bluemix;
- скачивать нужный файл из облачного хранилища;
- подсистема состоит из следующих модулей:

- модуль отображения интерфейса в режиме просмотра;
- модуль работы с ЭЭГ;
- модуль распознавания сигнала из головного мозга;
- модуль работа с файлами;

Подсистема отвечающая за режим обучения также тесно связана с подсистемой, отвечающей за базу данных и с подсистемой, отвечающей за работу с облачным хранилищем. Подробнее о взаимодействиях модулей в подсистеме можно посмотреть на рисунке 37.

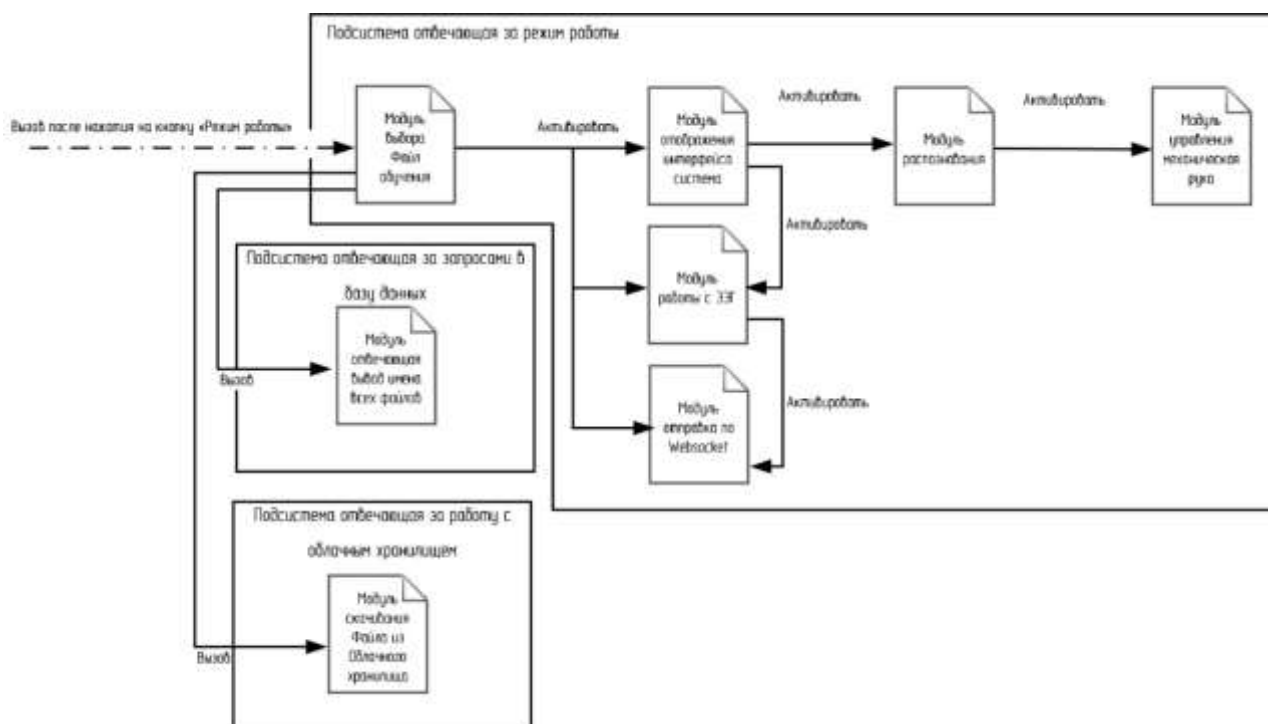


Рисунок 37 – Диаграмма взаимодействия модулей в подсистеме, отвечающей за режим работы

В подсистеме, отвечающей за режим обучения И. Н. С. были использованы такие библиотеки:

- deque – Библиотека, которая предоставляет специализированные типы данных, на основе словарей, кортежей, множеств, списков.
- serial – Библиотека, которая инкапсулирует доступ к порту;
- time – Библиотека для работы со временем;
- struct – Библиотека, которая предлагает функции pack() и unpack() для работы с форматами двоичных записей переменной длины;

– Tkinter - Данная библиотека позволяет реализовать локальный интерфейс взаимодействия с пользователем;

– pybrain - Одна из лучших Python библиотек для изучения и реализации большого количества разнообразных алгоритмов, связанных с нейронными сетями. Являет собой удачный пример совмещения компактного синтаксиса Python с хорошей реализацией большого набора различных алгоритмов из области машинного интеллекта;

– Thread – Библиотека, предназначенная для мультипоточного программирования;

– pickle – Библиотека, которая реализует мощный алгоритм сериализации и десериализации объектов Python. "Pickling" - процесс преобразования объекта Python в поток байтов, а "unpickling" - обратная операция, в результате которой поток байтов преобразуется обратно в Python-объект. Так как поток байтов легко можно записать в файл, модуль pickle широко применяется для сохранения и загрузки сложных объектов в Python;

– SupervisedDataSet – Библиотека, предназначенная для создания датасета;

– BackpropTrainer – Библиотека предназначенная для обучения сети;

– Adafruit_PCA9685 – Библиотека предназначенная для работы с 16 канальным ШИМ контроллером;

– os – Библиотека для работы с операционной системой, не зависящая от используемой операционной системы;

– buildNetwork – Библиотека для построения нейронной сети;

– SigmoidLayer – Библиотека для построения сигмоидного слоя;

– LinearLayer – Библиотека для построения линейного слоя;

– FeedForwardNetwork – Сеть прямой передачи;

– FullConnection – Библиотека для соединения слоя;

– Websocket – Библиотека для использования протокола полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Запуск режим работы

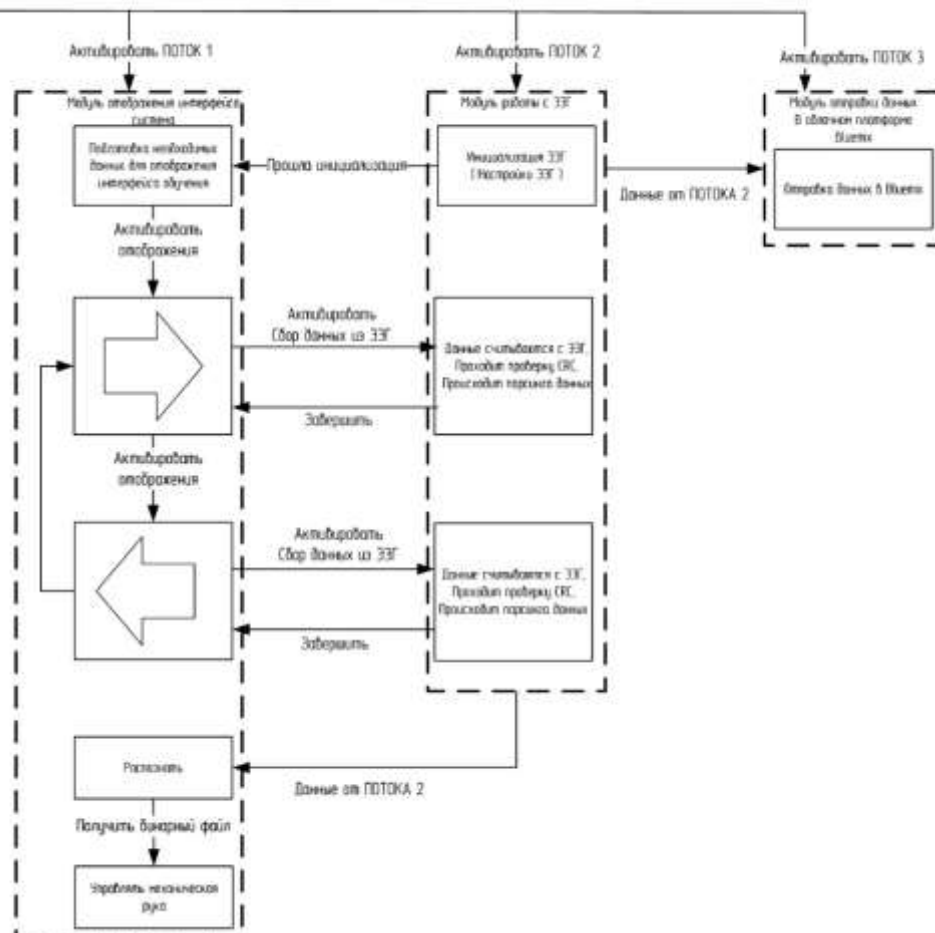


Рисунок 38 – Диаграмма взаимодействия потоков процесса в подсистеме отвечающая за режим обучения И. Н. С

Также как и в подсистеме, отвечающей за обучение И. Н. С., в подсистеме отвечающей за режим работы было использовано параллельное программирование, где задействованы несколько потоков для решения одной задачи. Так как процесс требует параллельных действий между интерфейсом работы и сбором данных с ЭЭГ подсистеме необходимо, кроме того, постоянно отправлять полученные данные из ЭЭГ в Bluemix. Следовательно, для реализации данного требования было использовано параллельное программирование.

Первый поток отвечает за отображение интерфейса обучения И. Н. С., второй отвечает за сбор данных из ЭЭГ, третий поток отвечает за отправку данных из ЭЭГ в облачную платформу Bluemix. При этом все потоки взаимосвязаны между собой для обмена необходимых данных. Подробное описание процесса взаимодействия потоков показано на рисунке 38.

В рабочем режиме также как и в режиме обучения И. Н. С. используются такие модули как:

- модуль отображения интерфейса;
- модуль парсинга данных.

Между подсистемой отвечающей за режим обучения И. Н. С. и подсистемой отвечающей за режим работы много сходства, но при этом в каждой из них есть свои определенные задачи. В режиме работы И. Н. С. должна работать и выдавать управляющие команды на основе информации, полученной от головного мозга.

Код активирующий режим работы И. Н. С.:

```
1  # Инициализация сети
2  n = constructPerceptron('perc', [64*8, 50, 10, 2]) # Создаём перцептрон
3  (trained_net, err) = trainNetwork(n, ds)
4  recognition = trained_net.activate(learnData)
5      if recognition[0] > recognition[1]:
6          # print "<=="
7          pwm.set_pwm(2, 0, servo_min)
8          time.sleep(1)
9      else:
10         # print "=="
11         pwm.set_pwm(2, 0, servo_max)
12         time.sleep(1)
```

ds – Значение, содержащее в себе обученную И. Н. С.

learnData – Данные собранные из ЭЭГ.

n = constructPerceptron('perc', [64*8, 50, 10, 2]) # Создание перцептрон

Перцептрон состоит из трёх типов элементов, а именно: поступающие от датчиков сигналы передаются ассоциативным элементам, а затем реагирующим элементам. Таким образом, перцептроны позволяют создать набор «ассоциаций» между входными стимулами и необходимой реакцией на выходе. В

биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов.

2.3.10 Модуль отвечающий за выбор бинарного файла

Перед тем как пользователь заходит в рабочий режим, система предлагает выбрать один из бинарных файлов для загрузки в И. Н. С. В бинарном файле содержится обученная нейронная сеть. У каждого пользователя имеется свои файлы, которые система создала автоматически при прохождении обучения (рисунок 39).

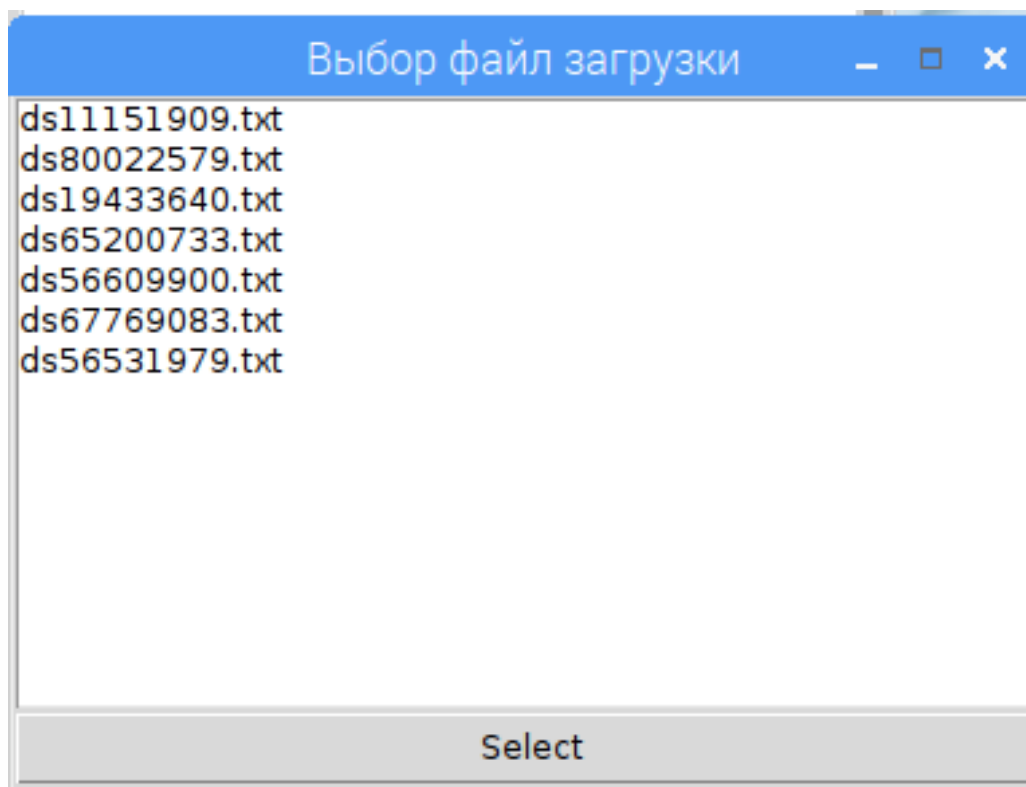


Рисунок 39 – Интерфейс выборки бинарного файла

Модуль отвечающий за выбор бинарного файла работает совместно с модулем работы с БД и модулем работы с облачным хранилищем. Перед тем как система переходит в режим работы, пользователю необходимо выбрать один из нескольких ранее созданных бинарных файлов для работы И. С. При нажатии на кнопку «Режим работы» в меню выбора режима, система выдает окно (рисунок 39) со списком. Список выдает имена всех файлов. Данные в списке заполняются

через БД, которая находится на сервере Bluemix. После выбора нужного файла, система отправляет запрос в облачное хранилище для скачивания. Далее файл находится на локальном диске и загружается в БД.

Структурная схема БД, где показаны связи между пользователем и файлами показана на рисунке 40.

Запрос в БД для выгрузка всех файлов у определенного пользователя:

Шаг 1: Получение ID - пользователя

```
sql="SELECT PersonID FROM Mibmx_29e62892a53c4f3.persons
```

```
WHERE persons.Mail LIKE " + username
```

Шаг 2: Получение списка файлов

```
sql="SELECT      data_name,      data_of_file FROM persons INNER      JOIN      dataofperson WHERE PersonID = (%s) AND Person_ID = (%s);", (str(data[0][0]), str(data[0][0]))
```

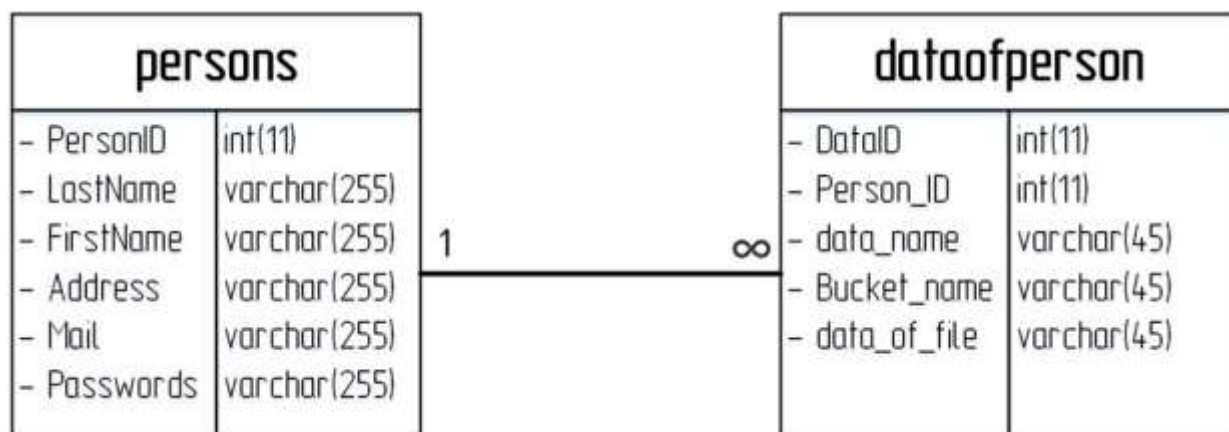


Рисунок 40 – Структурная схема БД

Для скачивания данных из облачного хранилища, необходимо вызвать функцию `download_ds(bucket_name, name_key):`

```
1 def download_file_cos(bucket_name, local_file_name, key):
2     try:
3         res=cos.download_file(Bucket=bucket_name,Key=key,Filename=local_file_name)
4     except Exception as e:
5         print(Exception, e)
6     else:
7         print('File Downloaded')
8
9 def download_ds(bucket_name, name_key):
10     print "from = ", name_key
11     download_file_cos(bucket_name, '/home/pi/losa_linh/all together/losa132.txt', name_key)
```

Bucket_name – Название персонального каталога у пользователя;

Name_key – Название файла.

2.3.11 Модуль, отвечающий за работу механизированной руки

Для управления механической рукой был использован контроллер PCA9685.

PCA9685 - это 16-канальный контроллер с I2C-шиной (рисунок 41), оптимизированный для использования в цветной подсветке ЖК-дисплея Red / Green / Blue / Amber (RGBA) либо для управления механической рукой.

Механическая рука состоит из четырех сервоприводов. Каждый привод имеет свой собственный 12-битный (4096 шагов) фиксированный частотный индивидуальный PWM-контроллер, который работает на программируемой частоте от 40 Гц до 1000 Гц с рабочим циклом, который регулируется от 0% до 100%.

Все выходы установлены на одну и ту же частоту ШИМ. Контроллер PCA9685 также имеет встроенный генератор для управления ШИМ. Однако частота, используемая для управления ШИМ в PCA9685, регулируется от примерно 40 Гц до 1000 Гц по сравнению с типичной частотой 96,6 кГц PCA9635. Это позволяет использовать PCA9685 с внешними контроллерами питания. Все биты устанавливаются с одинаковой частотой.

Интерфейсы PCA9685:

- 1 J3 Mini USB 5V input for PWM V+ J2 2P Terminal Block 5V input for PWM V+ ;
- 2 J1 Rs-Pi V2 GPIO output;
- 3 U2 PCA9685 (PWM Port 0 ~ 15);
- 4 R4,R5,R6,R7R8,R9(for U2 Address select A0,A1,A2,A3,A4,A5);
- 5 U3 PCA9685 (PWM Port 0 ~ 15);
- 6 R10,R11,R12,R13,R14,R15(for U3 Address select A0,A1,A2,A3,A4,A5);
- 7 Red power-good V+ LED 8. 1.6A PolySwitch Fuse for V+ input protect.

Код для управления механизированной рукой:

```
1  # Simple demo of of the PCA9685 PWM servo/LED controller library.
2  # This will move channel 0 from min to max position repeatedly.
3  # Author: Tony DiCola
4  # License: Public Domain
5  # from __future__ import division
6  import time
7  # Import the PCA9685 module.
8  import Adafruit_PCA9685
9  # Uncomment to enable debug output.
10 #import logging
11 #logging.basicConfig(level=logging.DEBUG)
12 # Initialise the PCA9685 using the default address (0x40).
13 pwm = Adafruit_PCA9685.PCA9685()
14 # Alternatively specify a different address and/or bus:
15 #pwm = Adafruit_PCA9685.PCA9685(address=0x41, busnum=2)
16
17 # Configure min and max servo pulse lengths
18 servo_min = 150 # Min pulse length out of 4096
19 servo_max = 600 # Max pulse length out of 4096
20 # Helper function to make setting a servo pulse width simpler.
21 def set_servo_pulse(channel, pulse):
22     pulse_length = 1000000 # 1,000,000 us per second
23     pulse_length //= 60 # 60 Hz
24     print('{0}us per period'.format(pulse_length))
25     pulse_length //= 4096 # 12 bits of resolution
26     print('{0}us per bit'.format(pulse_length))
27     pulse *= 1000
28     pulse //= pulse_length
29     pwm.set_pwm(channel, 0, pulse)
30 # Set frequency to 60hz, good for servos.
31 pwm.set_pwm_freq(60)
32 print('Moving servo on channel 0, press Ctrl-C to quit...')
33 while True:
34     # Move servo on channel 0 between extremes.
35     pwm.set_pwm(2, 0, servo_min)
36     time.sleep(1)
37     pwm.set_pwm(2, 0, servo_max)
38     time.sleep(1)
```


2.3.12 Подсистема отображения графиков

При работе в области нейрокомпьютерного интерфейса, наблюдение за мозговой активностью пользователя является одной из важнейших задач.

В реализуемой системе предусматривается функция визуализации мозговой активности в виде графиков.

Так как локальный интерфейс пользователя в режиме работы будет использован для взаимодействия с пользователем, а так же вычислительная мощность хаба (raspberrypi) почти полностью будет загружена И. Н. С. , то было принято решение реализовать дополнительный веб интерфейс, который будет параллельно работать с локальным.

Веб интерфейс был запущен и развернут на облачной платформе Bluemix. Веб интерфейс позволяет пользователю наблюдать за активностью мозговой волны в виде графика (рисунок 43).

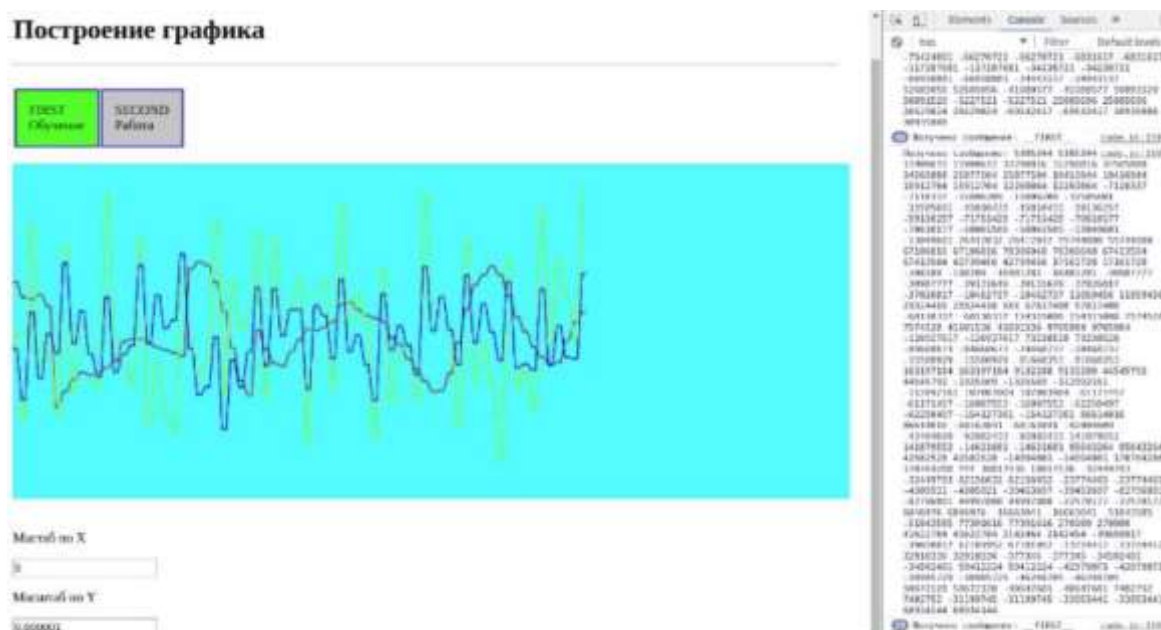


Рисунок 43 – веб интерфейс системы

Для более удобного наблюдения была реализована дополнительная функция для манипуляции графиков:

- изменение масштаба по оси x;
- изменение масштаба по оси y;

– изменение скорости отображения.

Для реализации подсистемы отображения графиков была использована среда разработки как Nodejs (рисунок 43), а также такие языки программирования как Javascript, html, css.

Cloud Foundry Applications					
Name	Region	CF Org	CF Space	Memory (MB)	Status
neuro-console-ws-client	United Kingdom	loza	Brainstorm	256	Running
severws	United Kingdom	loza	Brainstorm	256	Running

Рисунок 43 – Рабочее пространство Bluemix с разворачиваемыми веб приложениями

2.3.13 Описание процесса передачи данных по Websocket

Поскольку в системе работает параллельно два интерфейса, а также некоторые программные модули, находящиеся на веб – интерфейсе необходимо взаимодействовать с модулем из локального интерфейса для получения данных из ЭЭГ. Было принято решение использовать Websocket для передачи данных между хабом (клиент) и сервером Bluemix. На рисунке 44 показано, какие программные модули были использованы в процессе передачи данных.

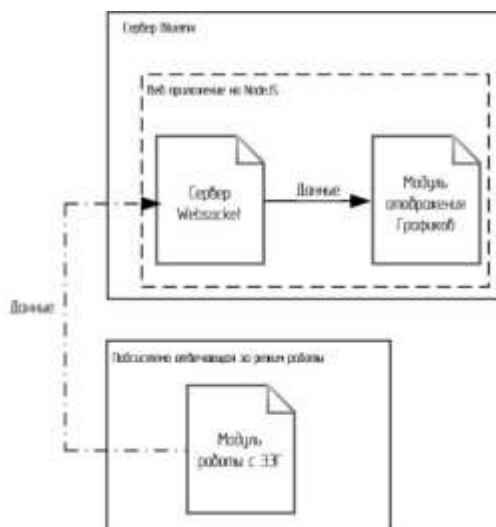


Рисунок 44 – Диаграмма взаимодействия модулей в процессе передачи данных по Websocket

Для построения графиков для 3х каналов необходимо передавать с клиента-отправителя хаб (Raspberry Pi3) на сокет сервер строку с данными. В строку помещаются массивы целых чисел (int): числа одного массива разделены пробелами, между 1м и 2м массивом разделитель XXX, между 2м и 3м - YYY.

Формат: int int int ... XXX ... int int int ... YYY ... int int int

Пример: "1234567 1234567 1234567 XXX 2345678 2345678 2345678 YYY 3456789 3456789 3456789".

Данные передаются в виде JSON объекта на сервер Websocket.

Код сервера Websocket:

```
1  "use strict";
2
3  let fs = require('fs');
4  let WebSocketServer = new require('ws');
5
6  let portNumber = process.env.PORT || 5005;
7  let websocketServer = new WebSocketServer.Server({
8    port: portNumber
9  });
10
11 let clients = {};
12 let nameCounter = 1;
13
14 console.log("PORT: " + portNumber);
15 console.log("_____ \n");
16
17
18 websocketServer.on("connection", function(ws) {
19   let id = "id_" + nameCounter;
20   nameCounter++;
21   clients[id] = ws;
22   console.log("новое соединение " + id);
23
24   ws.on("close", function() {
25     console.log('соединение закрыто ' + id);
26     try {
27       delete clients[id];
28     } catch (err) {
29       // err
30     }
31   });
32
33   ws.on("message", function(message) {
34     console.log("получено сообщение " + message + " от " + id);
35
36     for (let key in clients) {
37       try {
38         clients[key].send(message);
39       } catch (err) {
40         // err
41       }
42     }
43   });
44 });
```

Код для передачи данных из хаба в сервер по websocket:

```
1 def connection():
2     enableTrace(True)
3     ws = WebSocketApp("ws://ws-myserver.eu-gb.mybluemix.net/",
4                       on_message=on_message,
5                       on_error=on_error,
6                       on_close=on_close)
7     ws.on_open = on_open
8     ws.run_forever()
9 def on_open(ws):
10    v1 = []
11    v2 = []
12    v3 = []
13    while True:
14        while break_flag: #ждем разрешения чтения
15            time.sleep(0.02) # Задержка 20 мс
16        while get_packets() < 4:
17            time.sleep(0.02) # Задержка 20 мс
18        for j in range(16):
19            v1 += valuechannel14[-1][j]
20            v2 += valuechannel12[-1][j]
21            v3 += valuechannel13[-1][j]
22            # v4 += valuechannel14[-1][j]
23            # v5 += valuechannel15[-1][j]
24            # v6 += valuechannel16[-1][j]
25            # v7 += valuechannel17[-1][j]
26            # v8 += valuechannel18[-1][j]
27        s = ""
28        time.sleep(0.07)
29        s += " ".join(map(str, v1)) + " XXX " + " ".join(map(str, v2)) + " YYY " + " ".join(map(str, v3))
30        ws.send(s)
31        del v1[:]
32        del v2[:]
33        del v3[:]
34        # del v4[:]
35        # del v5[:]
36        # del v6[:]
37        # del v7[:]
38        # del v8[:]
```

Код приема данных в веб приложение:

```
1 socket.onopen = function() {
2     console.log("Соединение установлено");
3 };
4
5 socket.onclose = function(event) {
6     console.log("Соединение закрыто");
7 };
8
9 socket.onmessage = function(event) {
10    console.log("Получено сообщение: " + event.data);
11    let message = event.data.toString();
```

Вывод конструкторской части

В этом разделе рассмотрены вопросы, касающиеся выбора языка программирования, выбора архитектуры приложения, также выполнен анализ способов работы сервера, обзор способов обеспечения передачи данных. Рассмотрены такие механизмы И. Н. С., как облачные сервисы, параллельное программирование, веб – сокетные соединения, выделены их достоинства и недостатки. Рассмотрена структура базы данных, а также назначение и применение ресурсов системы. Рассмотрены принципы работы всех подсистем и программные модули, представлены необходимые графические материалы для описания архитектуры этих подсистем и программных модулей.

3. Разработка технологии использования ВСУЭУ

В прошлых разделах были описаны поэтапные разработки ВСУЭУ, также на каждом этапе было аргументировано почему и по какой причине мы применили определенные технология для разработки.

В этом разделе будет проведено этапное тестирование, то есть тестирование каждой подсистемы и каждого модуля, а после проведем комплексное тестирование. Также будет описан процесс развёртывания систем и последовательности сценариев его использования.

3.1 Настройка удаленного доступа к raspberry pi

Перед настройкой удаленного доступа к raspбери пи 3, рекомендуется задать статичный IP – адрес. Ниже описаны необходимые шаги для установки:

- 1) Необходимо узнать IP-адрес шлюза в своей локальной сети.

`netstat -r -n`

```
pi@raspberrypi:~ $ netstat -r -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window  irtt Iface
default          192.168.1.1     0.0.0.0         UG      0 0        0 eth0
192.168.1.0      *               255.255.255.0   U        0 0        0 eth0
```

- 2) Узнав IP-адрес роутера, отредактируем файл конфигурации DHCP на Raspberry Pi:

`sudo nano /etc/dhcpd.conf`

Допишем в конце строку:

`nodhcp`

И после этой строки назначим статический адрес для Ethernet-подключения:

`interface eth0`

`static ip_address=192.168.1.20/24`

`static routers=192.168.1.1`

`static domain_name_servers=192.168.1.1`

3) После этого остается только перезагрузить Raspberry Pi для применения изменений:

```
sudo reboot
```

```
#option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# A hook script is provided to lookup the hostname if not set by the DHCP
# server, but it should not be run by default.
nohook lookup-hostname
nodhcp
interface eth0
static ip_address=192.168.1.20/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Результат находится на рисунке 45

```
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:a4:02:5f
          inet addr:192.168.1.20  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f256:ebbd:5f14:5144/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:604885 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91928 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:84745344 (80.8 MiB)  TX bytes:27141630 (25.8 MiB)
```

Рисунок 45 – Результат после настройки IP

Для удаленного управления используются следующие программы:

1) Putty – Для доступа к консоли по локальному либо по внешнему IP – адресу (Рисунок 46).

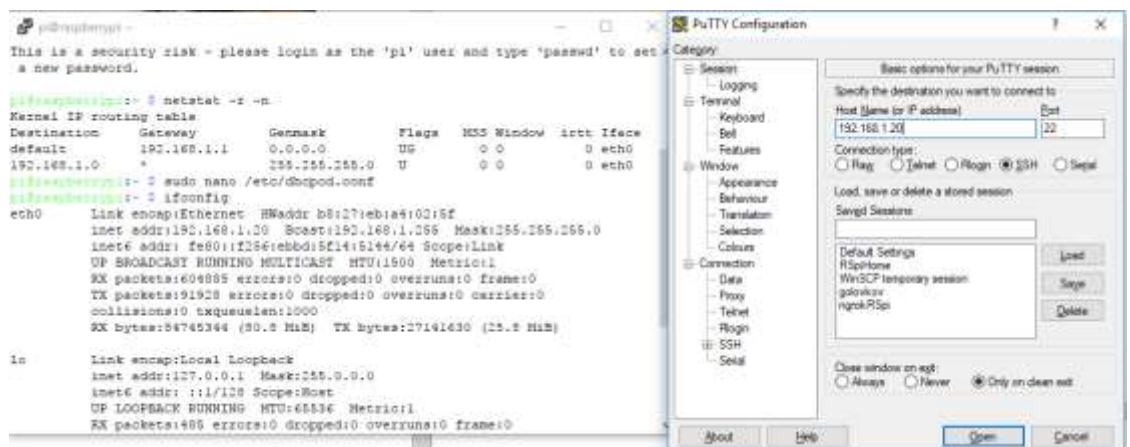


Рисунок 46 – Работа по удаленной консоли Putty

2) VNC Viewer – Используется для доступа к удаленному рабочему столу (Рисунок 47).



Рисунок 47 – Доступ к удаленному рабочему столу

3) WinSCP – программа используется для удаленного доступа к файловой системе raspberry pi (Рисунок 48).

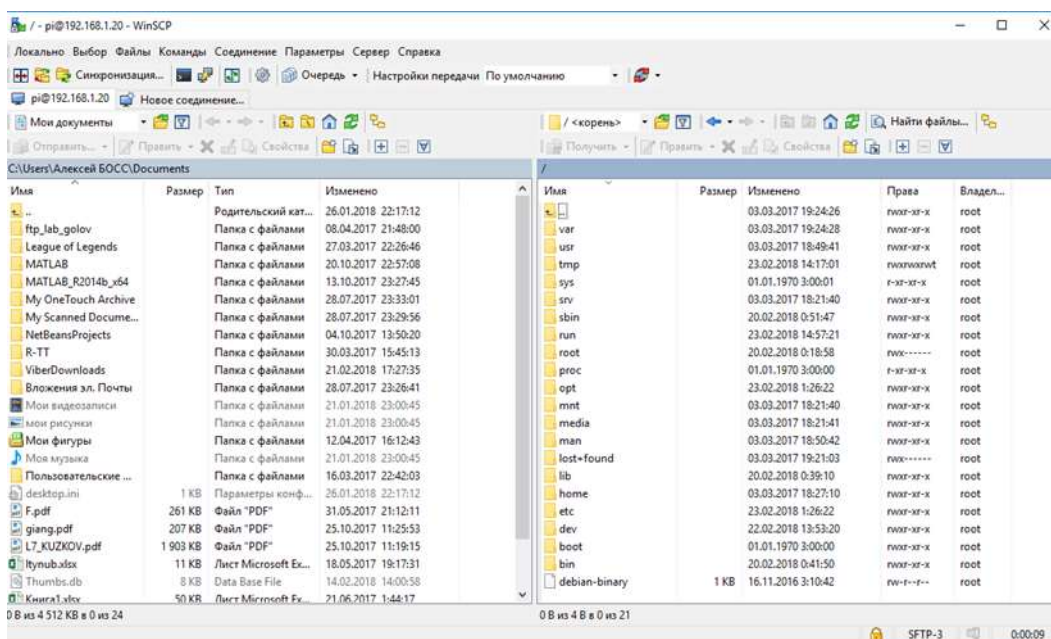


Рисунок 48 – Удаленный доступ к файловой системе

Перед использованием набора программ, необходимых для удаленной работы, необходимо в настройках Raspberry pi 3 разрешить доступ по SSH, VNC и SFTP (Рисунок 49).

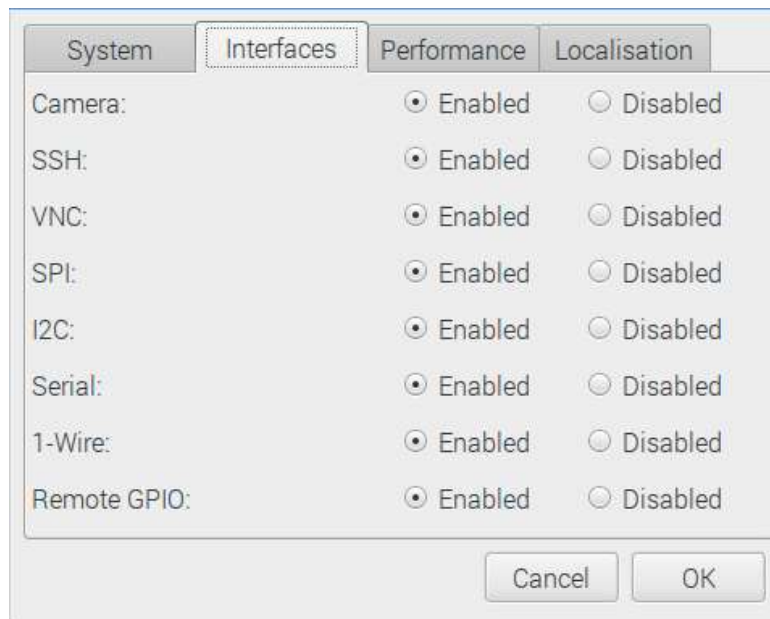


Рисунок 49 – Настройка разрешения доступа

3.2 Установка необходимых библиотек для запуска и отлаживания программного кода

Перед установкой необходимого набора библиотек для тестирования модуля, рекомендуется обновить ОС raspbian. Ниже приведены скрипты:

```
sudo rpi-update
```

```
sudo reboot
```

Далее обновляем пакеты:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

В набор библиотек входят такие библиотеки как:

PyBrain – одна из лучших Python библиотек для изучения и реализации большого количества разнообразных алгоритмов связанных с нейронными сетями. Являет собой удачный пример совмещения компактного синтаксиса Python с хорошей реализацией большого набора различных алгоритмов из области машинного интеллекта.

```
pip install pybrain
```

```
sudo apt -y install nodejs
```

```
sudo pip install websocket-client
```

Установка Cutesom для проверки работоспособности устройства через последовательный порт (Рисунок 13).

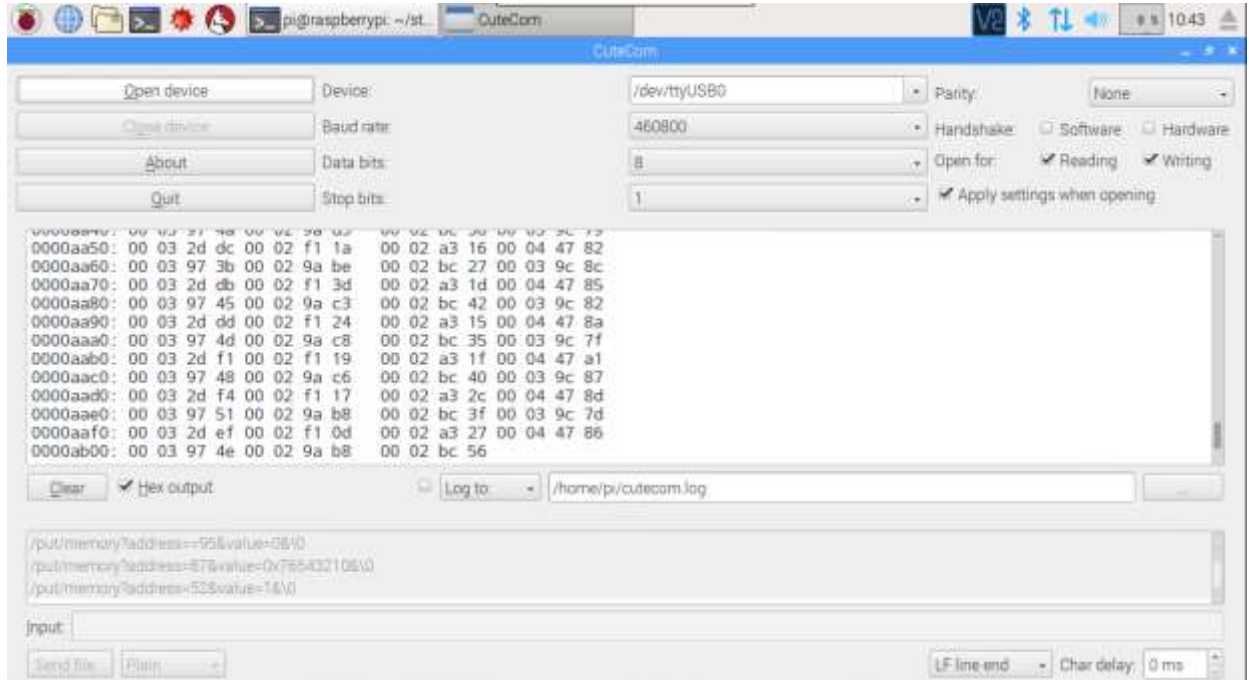


Рисунок 50 – Программа cutesom для работы с последовательным портом

3.3 Публикация веб приложения на облачной платформе Bluemix

3.3.1 Установка Bluemix CLI

Bluemix CLI – Это интерфейс командной строки IBM Cloud.

1 Необходимо скачать и устанавливать Bluemix CLI (bx cli):

```
$ tar -xvf Bluemix_CLI.tar.gz
```

```
$ cd Bluemix_CLI
```

```
$ sudo ./install_bluemix_cli
```

2 Проверка доступности команды bx: \$ bx --version

Далее нужно зайти под своей учетной записью IBM Bluemix:

```
$ bx login
```

3 Установим dev пакет для bx с сайта

```
$ curl -sL https://ibm.biz/idt-installer | bash
```

После установки dev пакета необходимо выбирать Express.js + Webpack приложение node.js из списка Starter Kit Bluemix:

- 1 Скачать архив с файлами тестового hello world приложения
- 2 Разархивировать в папку с helloworld-проектом

Тестирование работоспособность приложения, развернув его локально.

Для этого в папке проекта создадим и запустим контейнер:

```
bx dev build
```

```
bx dev run
```

Здесь и далее для выполнения команд bx dev может понадобиться sudo, например: sudo bx dev build

После этого можно посмотреть результат запуска приложения в браузере: localhost:3000

3.3.2 Публикации проекта в облаке Bluemix

- 1) Необходимо скачать архив:

https://github.com/akenoq/IBM_hackathon_deploy

В этом архиве файлы, необходимые для публикации в облаке Bluemix 3D-тренажера и приложения для построения графиков при помощи StarterKit Node.js + webpack. Каждая директория - одно приложение:

– neuro-console-ws-client - Приложение для построения графика по данным, полученным с энцефалографа;

– neuro-console-ws-server - Сокет-сервер на Node.js к которому подсоединен клиент1 - приложение для построения графика и клиент2 - Raspberry Pi3.

- 2) Публикация приложения для построения графиков

Для работы приложения необходимо развернуть в облаке:

– neuro-console-ws-client - приложение для построения графика по данным, полученным с энцефалографа (консоль);

– neuro-console-ws-server - сокет-сервер на Node.js к которому подсоединен клиент1 - приложение для построения графика и клиент2 - Raspberry Pi3.

Для этого, следуя инструкции с 3.3.1, выбрав приложение Node.js + webpack из списка Starter Kit Bluemix и заменив папки, package.json и webpack в полученном starter-kit проекте на скаченные из директорий файлы.

Проверка локально корректность получившегося проекта:

`bx dev build`

`bx dev run`

Публикация в облаке:

`bx dev deploy`

3.4 Доступ к облачному платформу Bluemix

Доступ в облачной платформе Bluemix позволяет администратору запустить систему. А также изменить исходный код подсистемы, отвечающей за работы веб интерфейса. Имея доступ к облачному платформу можно войти во всех используемых сервисах (ClearDB Managed MySQL Database, NodeJS, Cloud Object Storage) (Рисунок 51).



Рисунок 51 – Интерфейс в режиме «Доступ к серверу Bluemix»

В сервере Bluemix администратор может выделить каждому приложению определенный программный и аппаратный ресурс. Это позволяет администратору регулировать быстродействие работы каждой приложения по-своему.

3.5 Доступ в БД

Доступ в БД осуществляется через СУБД MySQL workbench CE (рисунок 52). Для создание БД необходимо активировать сервиса ClearDB Managed MySQL Database на облачной платформе Bluemix (Рисунок 53).

Удаленное подключение СУБД у БД MySQL осуществляются с помощью протокола TCP / IP:

- Hostname: eu-cdbr-sl-lhr-01.cleardb.net
- Username: bd51db6b9db508
- Password: 1561ab78

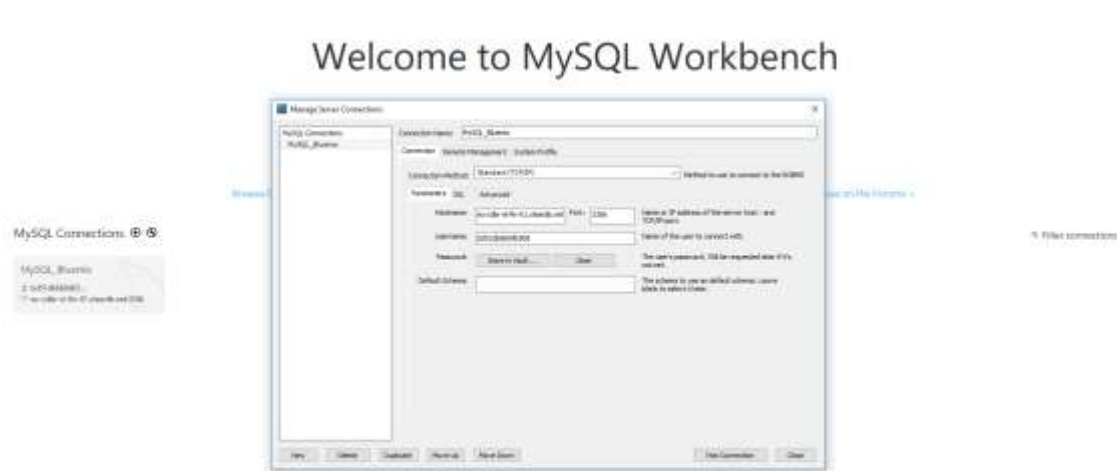


Рисунок 52 – Интерфейс СУБД MySQL Workbench

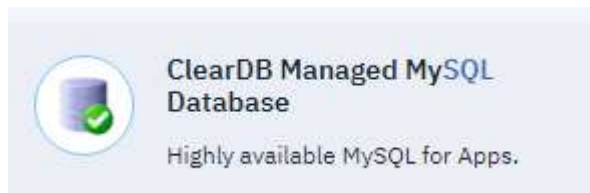


Рисунок 53 – Сервис ClearDB Managed MySQL Database

В Сервисе ClearDB Managed MySQL Database администратор может добавлять, удалять, создавать новые таблицы, а так же запускать SQL – запросы.

Кроме этого в сервисе позволяет администратору следить за загруженностью БД (рисунок 47).



Рисунок 54 – Панель отображения загруженности БД

Сервис позволяет администратору сделать Ваксуп в случае сбоя БД. Точки восстановления определяется сервисом (рисунок 55).

ibmx_29e62892a53c4f3: Backups & Jobs (Max Retention: 5 Days)						New backup >>
Current Jobs						Refresh
Created	Last Updated	Type	Status	Server	Message	
No jobs are currently queued. Automated jobs, such as backups, restores and cluster migrations will show up here when they are active.						
Available Backups						
Created	Last Restored	Size	Server	Action		
5/16/2018 04:32AM	Never restored	1MB	eu-mm-auto-sl-lhr-01-b.cleardb.net	Restore Backup Export		
5/16/2018 04:32AM	Never restored	1MB	eu-mm-auto-sl-lhr-01-a.cleardb.net	Restore Backup Export		
5/16/2018 04:29AM	Never restored	1MB	eu-mm-auto-sl-ams-01-c.cleardb.net	Restore Backup Export		
5/15/2018 04:38AM	Never restored	1MB	eu-mm-auto-sl-lhr-01-b.cleardb.net	Restore Backup Export		
5/15/2018 04:30AM	Never restored	1MB	eu-mm-auto-sl-lhr-01-a.cleardb.net	Restore Backup Export		
5/15/2018 04:26AM	Never restored	1MB	eu-mm-auto-sl-ams-01-c.cleardb.net	Restore Backup Export		
5/14/2018 04:35AM	Never restored	1MB	eu-mm-auto-sl-lhr-01-a.cleardb.net	Restore Backup Export		

Рисунок 55 – Режим восстановления БД

3.6 Доступ к облачному хранилищу IBM Cloud Object Storage

IBM Cloud Object Storage можно разворачивать как локально, так и в облачной среде IBM (IBM Cloud), что делает его идеальным хранилищем объектов для гибридного облака. Кроме этого, IBM Cloud Object Storage входит

в семейство IBM Storage с широкими возможностями хранения данных (рисунок 56 и 57).

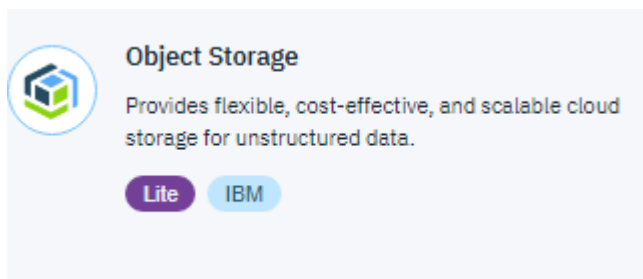


Рисунок 56 – Сервис IBM Cloud Object Storage

Достоинства:

– Гибкость: Благодаря возможности выбора варианта развертывания в соответствии с вашими потребностями: развертывание выделенного и частного или общедоступного облака в одном или нескольких регионах.

– Масштабируемость: Широкие возможности масштабирования независимо от объема данных и потребностей благодаря облачным ЦОД IBM, расположенным по всему миру.

– Простота: Повышение эффективности и сокращение затрат благодаря единой интуитивно понятной панели управления хранением данных.



Рисунок 57 – Рабочее пространство в сервисе IBM Cloud Object Storage

3.7 Выбор стратегии тестирования и разработка тестов

Так как модель жизненного цикла была разработана на основе спиральной модели, то тестирование программы проводилось после каждого завершеного этана проектирования.

Существует 3 стратегии тестирования программно – аппаратного продукта: ручное тестирования, тестирования по принципу «белого ящика», тестирование по принципу «черного ящика».

Ручное тестирования эффективно, когда его проводят несколько человек, разрабатывавших программно – аппаратный продукт. Здесь же система разрабатывалась одним человеком и тестировалась им же, следовательно, в данном случае метод неэффективен.

В данной работе в основном применялись 2 метода: черного ящика и оценочное тестирование.

Метод тестирования черного ящика был выбран так, как он позволяет тестировать функциональное поведение объекта (системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта. Таким образом тестировать данный программный продукт может любой пользователь, что увеличивает количество тестирующих системы и помогают разработчику улучшить свой программно – аппаратный продукт.

После завершения тестирования черного ящика приступаем к оценочному тестированию, целью которого является тестирование системы на соответствие основным требованиям. Эта стадия тестирования особенно важна для программно - аппаратных продуктов, предназначенных для продажи на рынке.


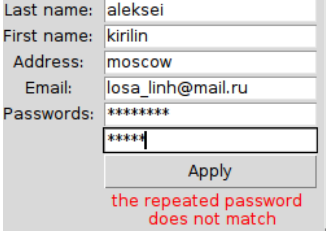
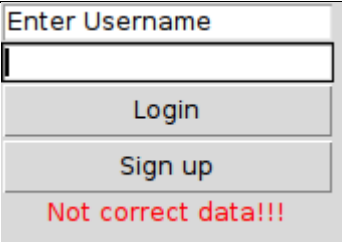
Оценочное тестирование, которое также называют «тестированием системы в целом», включает следующие виды:

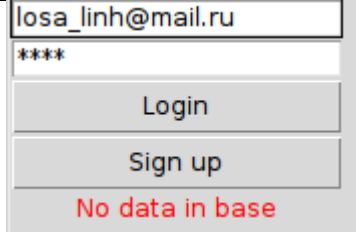
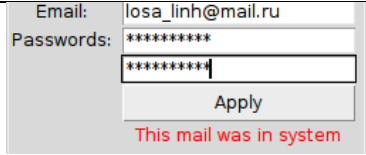

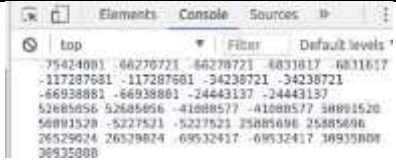

- тестирование удобства работы пользователя с интерфейсом программы;
- тестирование на соответствие программно – аппаратного продукта и документации основным положениям технического задания.

3.8 Тестирование по черному ящику



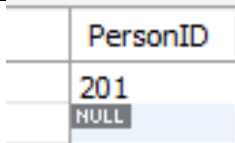
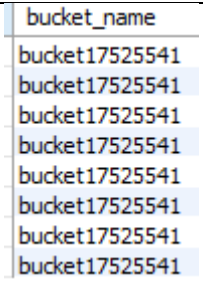
В тестировании на правильность построения сравнивают результат построения данной системы с верными предполагаемыми результатами. Сравнение систем можно посмотреть в таблицах 13.


Таблица 13 – Комплексное тестирование

Номер теста	Назначение теста	Ожидаемый результат	Реакция системы	Вывод
1	Проверка функции регистрации нового пользователя	Новая запись в БД		Успешно
2	Проверка на некорректность ввода пароля при регистрации	Система выдаст ошибки		Успешно
3	Проверка на некорректность ввода электронной почты при регистрации	Система выдаст ошибки		Успешно
4	Проверка на пустой ввод логина и пароля	Система выдаст ошибки		Успешно

5	Проверка на ввода неправильного пароля	Система выдаст ошибки		Успешно
6	Проверка на повторный ввод email при регистрации	Система выдаст ошибки		Успешно
7	Проверка на получения данных из ЭГГ	Система должна вывести пакет данных в консоли		Успешно
8	Проверка передачи данных по websocket	Система должна вывести пакет данных в консоли браузера		Успешно
9	Проверка подключения к БД	Система должна получить данные из БД		Успешно

10	Проверка подключения к облачному хранилищу	Система должна вывести все файлы	objects in bucket17525541: ["ds11151909.txt", "ds19433640.txt", "ds56531979.txt", "ds56609900.txt", "ds65200733.txt", "ds67769083.txt", "ds7464157.txt", "ds80022579.txt"] input_dataperson end	Успешно
11	Проверка процесса записи в файл	В бинарном файле должен содержать данные	1 6370 7962 7261 696e 2e64 6174 6173 6574 2 732e 7375 7065 7276 6973 6564 0a53 7570 3 6572 7669 7365 6444 6174 6153 6574 0a70 4 300a 2849 3531 320a 4932 0a74 7031 0a52 5 7032 0a28 6470 330a 5327 6c69 6e6b 270a 6 7034 0a28 6c70 350a 5327 696e 7075 7427 7 0a70 360a 6153 2774 6172 6765 7427 0a70 8 370a 6173 5327 6461 7461 270a 7038 0a28	Успешно
12	Проверка процесса чтения из файла	Вывести сохраненную И. Н. С		Успешно
13	Проверка процесса хэширование пароля	В БД в поле "Passwords" содержится зашифрованный пароль	Passwords 979d7eb6b6b0cedbd8275fd7cef05a75 10b6093a9e8ca9f407fc8ccf229727ce	Успешно
14	Проверка подключения к хабу	Имеет доступ к рабочему столу по VNC	V2 192.168.1.20 (raspberrypi) - VNC Viewer 	Успешно
15	Проверка подключения к файловому пространству	Имеет доступ к файлому пространству	all together - pi@192.168.1.20 - WinSCP	Успешно

	пространству хаба	у хаба по SFTP		
16	Проверка модуля параллельного программирования	Система должна выдать результат и в локальном интерфейсе и в консольном		Успешно
17	Проверка запроса на добавления нового пользователя	Новая запись в БД		Успешно
18	Проверка запроса на поиск ID пользователя по email	ID пользователья по email		Успешно
19	Проверка запроса на вывод всех файлов от определённого пользователя	Всех файлов от определённого пользователя		Успешно

20	Проверка запроса на вывод всей информации о пользователе по email	Вся информаци я от пользовател я по email		Успешно
----	--	---	--	---------

3.9 Оценочное тестирование

После завершения всех предыдущих тестов программно - аппаратный продукт тестируется на соответствие требованиям, предъявленным в техническом задании. Для осуществления данного вида тестирования привлекается разработчик системы и некоторое количество потенциальных пользователей данного программного продукта. Результаты оценочного тестирования представлены ниже.

Тестирование удобства эксплуатации проводится на стадии разработки прототипа интерфейса (на данном этапе участвует два человека) и на заключительной стадии (проект реализован).

На данном этапе пользователями даны оценки некоторых характеристик интерфейса, которые приведены в десятибалльной шкале (таблица 14).

Таблица 14 – Пользовательская оценка интерфейса программы

Пользователь	Оценка интерфейса	Удобство интерфейса	Полнота и доступность основных функций
1	8	9	7
2	7	7	6

Данные оценки являются более чем удовлетворительными, поэтому интерфейс не отклонен, а изменен в соответствии с рекомендациями пользователей.

Ниже представлены результаты тестирования программы после её реализации.

Пользовательские оценки некоторых характеристик программного продукта приведены в таблице 15.

Таблица 15 – Результаты тестирования программы

Пользователь	Соответствие функций описанию	Понятность последовательности действий	Отображение результатов	Оценка программы
1	8	9	7	8
2	9	10	9	8
3	9	8	9	7
4	10	8	8	9

Вывод технологической части

В данном разделе описаны ряд сервисов, использованных в системе, методы тестирования и также инструкции по развертыванию системы.

Также провели комплексное тестирование системы, описали и показали все формы интерфейса, с которыми работает администратор и пользователь в процессе всего цикла работы: от визуализации до управления приборов. Объяснены технологические решения, которые использовали. Отдельное внимание уделено анализу достоинств и недостатков видов тестирования.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы создана вычислительная система управления электронными устройствами с помощью ИМК.

В процессе выполнения работы проведены аналитические обзоры применения микроконтроллерных систем в инфраструктуре решений Интернета вещей, выявлены его основные понятия и области применения, рассмотрены основные характеристики облачных вычислений и их сервисные услуги. Также проанализированы различные типы ИМК.

На основании проведенного анализа выбрана облачная платформа IBM Bluemix типа PaaS в качестве основного инструмента разработки и ИМК на основе вызванной волны P300 для взаимодействия человека с устройствами.

Также проведены ряд работы по проектированию систем:

- Обосновали рациональные принципы построения, структурной схемы и алгоритма функционирования ВСУЭУ;
- Разработали графические материалы для ВСУЭУ;
- Собрали модель ВСУЭУ на реальном оборудовании;
- Реализовали проводную и беспроводную передачу данных;
- Реализовали аутентификацию;
- Разработали веб-интерфейс для пользователя;
- Разработали локальный интерфейс для взаимодействия с пользователем;
- Провели комплексное тестирование работоспособности ВСУЭУ.

Результат данной работы имеет образовательный характер и применяется на факультете «Информатика и системы управления» кафедры «Компьютерные системы и сети» МГТУ им. Н. Э. Баумана для проведения хакатона по быстрого прототипирования решений Интернета вещей с применением интерфейса "Мозг-компьютер".

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Маслов. В.С. Методические указания к курсовому/дипломному проекту и квалификационной работе. – М.: МГТУ им. Н.Э. Баумана, 2005. – 5 с. ил.

2 Иванова Г.С. Оформление текстовых документов. Методические указания по оформлению расчетно-пояснительных записок дипломных и квалификационных работ. - М: МГТУ им. Н.Э. Баумана, 2004. – 10 с.

3 M2M [Электронный ресурс] // Webopedia. URL:<http://www.webopedia.com/TERM/M/M2M.html> (дата обращения: 5.01.2015).

4 John Breeden II. What is M2M, and why is it the future of code? [Электронный ресурс] // GSN. 2013. 12 March. URL: <http://gcn.com/articles/2013/03/22/m2m-future-of-code.aspx> (дата обращения: 5.01.2015).

5 Stephen Lawson. From M2M to IoT: Old industries have to learn new tricks [Электронный ресурс] // IDG News Service. 2014. 14 December. URL: <http://www.pcworld.com/article/2861236/from-m2m-to-iot-old-industries-have-to-learn-new-tricks.html> (дата обращения: 13.02.2015).

6 Charles McLellan. M2M and the Internet of Things: A guide [Электронный ресурс] // 2013. 10 January. URL: <http://www.zdnet.com/article/m2m-and-the-internet-of-things-a-guide/> (дата обращения: 13.02.2015).

7 Michael Hausenblas. Key Requirements for an IoT Data Platform [Электронный ресурс] // MAPR. 2015. 19 January. URL: https://www.mapr.com/blog/key-requirements-iot-data-platform#.VX1xq_ntmko (дата обращения: 13.02.2015).

8 Global Strategy, Business Development, Freescale, Emerging Technologies, ARM. What the Internet of Things (IoT) needs to become a Reality [Электронный ресурс] // Freescale. 2014. May. URL: freescale.com/IoT (дата обращения: 19.02.2015).

9 OAuth 2.0. [Электронный ресурс] // OAuth.URL: <http://oauth.net/2/> (дата обращения: 19.02.2015).

10 Stan Schneider. Understanding The Protocols Behind The Internet Of Things [Электронный ресурс] // Electronic Design. 2013. 9 October. URL: <http://m.electronicdesign.com/embedded/understanding-protocols-behind-internet-things#Protocol%20Overview> (дата обращения: 20.02.2015).

11 Quality of service. [Электронный ресурс] // IBM. URL:<https://www.eclipse.org/paho/files/mqtt/doc/Cclient/qos.html> (дата обращения: 21.02.2015).

12 MQTT. [Электронный ресурс] // MQTT. URL: <http://mqtt.org/> (дата обращения: 21.02.2015).

13 CoAP. [Электронный ресурс] // TZI. URL: <http://coap.technology/> (дата обращения: 21.02.2015).

14 Policy Decision Points & Policy Enforcement Points. [Электронный ресурс] // CCSK Guide. 2013. 11 November. URL:<http://ccskguide.org/policy-decision-points-policy-enforcement-points/> (дата обращения: 23.02.2015).

15 Rackspace Support. Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS. [Электронный ресурс] // Rackspace Support Network. 2013. 22 October. URL: http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas (дата обращения: 25.02.2015).

16 Types of Cloud Computing: Private, Public and Hybrid Clouds. [Электронный ресурс] // AppCore. URL:<http://www.appcore.com/types-cloud-computing-private-public-hybrid-clouds/> (дата обращения: 28.02.2015).

17 Материалы сайта <http://azure.microsoft.com/ru-ru/> (дата обращения: 1.03.2015).

18 Что такое IBM Bluemix? [Электронный ресурс] // IBM. URL:<https://www.ibm.com/developerworks/ru/library/cl-bluemixfoundry/> (дата обращения: 1.03.2015).

19 IBM Bluemix. [Электронный ресурс] // IBM. URL: <http://www.ibm.com/cloud-computing/bluemix/> (дата обращения: 3.03.2015).

20 Node-RED [Электронный ресурс] // IBM. URL:<http://nodered.org/> (дата обращения: 3.03.2015).

21 Стандарт ANSI/TIA/EIA-607-1994 (Август 1, 1994). Требования к телекоммуникационной системе выравнивания потенциалов и заземления коммерческих зданий.

22 Правил устройства электроустановок, 7-е издание, утв. Приказом Минэнерго РФ от 20.06.2003 N 242.

23 Raspberry Pi Hardware. [Электронный ресурс] // Raspberry Pi Foundation. URL:<https://www.raspberrypi.org/documentation/hardware/raspberrypi/> (дата обращения: 12.03.2015).

24 Documentation. [Электронный ресурс] // IBM Cloudant. URL: <https://docs.cloudant.com/> (дата обращения: 16.03.2015).

25 Электроэнцефалография. // Wikipedia. 2017. URL: <https://ru.wikipedia.org/wiki/Электроэнцефалография> (дата обращения: 01.04.2018).

26 Кирой В.Н. Электрографические корреляты реальных и мысленных движений: спектральный анализ. // Журнал высшей нервной деятельности им. И.П.Павлова. 2010. Т.60, №5. С. 525-533.

27 Сотников П.И. Обзор методов обработки сигнала электроэнцефалограммы в интерфейсах мозг-компьютер. // Инженерный вестник. 2014. №10. С. 612-632.

28 Мокиенко О.А. Основанный на воображении движения интерфейс мозг-компьютер в реабилитации пациентов с гемипарезом. // Бюллетень сибирской медицины. 2013. Т. 12, №2. С. 30-35.

29 MQTT Version 3.1.1. // MQTT. 2014. URL: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html> (дата обращения: 02.05.2018).

30 Хакатон IoT ИМК, МГТУ 18 ноября - 9 декабря 2017. // GitHub. 2017. URL: <https://github.com/alexbmstu/bmstu-hackathon> (дата обращения: 15.04.2018).