СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫЙ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ ТЕРМИНОВ

ACK (англ. Acknowledge) — подтверждение.

BLE (англ. Bluetooth Low Energy, Bluetooth LE, представленная также как Bluetooth Smart) — выпущенная в декабре 2009 года версия спецификации ядра беспроводной технологии Bluetooth, наиболее существенным достоинством которой является сверхмалое пиковое энергопотребление, среднее энергопотребление и энергопотребление в режиме простоя.

CAN (англ. Controller Area Network) — стандарт промышленной сети, ориентированный, прежде всего, на объединение в единую сеть различных исполнительных устройств и датчиков. Режим передачи — последовательный, широковещательный, пакетный.

CLK (англ. clock signal) – тактирующий сигнал.

DFD (англ. data flow diagrams) - диаграммы потоков данных.

ERR (англ. Error) — ошибка.

HAL (англ. Hardware Abstraction Layer) – слой аппаратных абстракций.

HCI (англ. Host Controller interface) – контроллер хоста интерфейса Bluetooth.

I2C (англ. Inter-Integrated Circuit) — последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов.

IoT (англ. internet of things, IoT) - концепция сети передачи данных между физическими объектами («вещами»), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой.

KNN (англ. K-nearest neighborhood) - Метод k-ближайших соседей. Метрический алгоритм для автоматической классификации объектов или регрессии.

L2CAP (англ. Logical Link Control and Adaptation Protocol) – протокол предоставляющий услуги по работе с данными, как ориентированные на

соединения, так и без ориентации на них, протоколам более высокого уровня с возможностями мультиплексирования и обеспечением операций по сегментации и обратной сборке.

MAV (англ. Mean absolute value) – среднее абсолютное значение

MCU (англ. Microcontroller Unit) - микроконтроллер (микросхема, содержащая процессор, память и периферийные устройства).

MPR (англ. Myoelectric pattern recognition) – система распознавания миоэлектрических паттернов

RFCOMM (англ. Radio Frequency Communication) – протокол, который эмулирует последовательные порты поверх протокола L2CAP.

SPI (англ. Serial Peripheral Interface) — последовательный периферийный интерфейс.

STM – семейство битных микроконтроллеров производства STMicroelectronics.

UART (англ. Universal Asynchronous Receiver-Transmitter, UART) - Универсальный Асинхронный Приёмопередатчик.

USB (англ. Universal Serial Bus — «универсальная последовательная шина») — последовательный интерфейс для подключения периферийных устройств к вычислительной технике.

WAN (англ. Wide Area Network) – это глобальная компьютерная сеть. Проще говоря, это интернет.

Wi-Fi — технология беспроводной локальной сети с устройствами на основе стандартов IEEE 802.11.

АЦП (Аналого-цифровой преобразователь, англ. Analog-to-digital converter, ADC) — устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал).

ФНЧ — Фильтр ни́жних часто́т. Электронный или любой другой фильтр, эффективно пропускающий частотный спектр сигнала ниже некоторой частоты (частоты среза) и подавляющий частоты сигнала выше этой частоты.

ЭМГ (англ. EMG, Электромиография, ЭНМГ, миография, электронейромиография) — метод исследования биоэлектрических потенциалов, возникающих в скелетных мышцах человека и животных при возбуждении мышечных волокон; регистрация электрической активности мышц.

ВВЕДЕНИЕ

Каждый год в мире делают более 1 млн ампутаций. На США приходится более 185 тыс., на Россию — более 70 тыс. [61].

Главные причины в современных условиях— диабет и травмы, но есть еще и те, у кого конечностей нет от рождения. К началу 2019 года доля инвалидов составила 8,1% [75] от общего населения России, или 11,947 млн человек. Более 200 тысяч [77] нуждаются в протезировании рук или ног. По данным МСЭ количество может только увеличиваться.

В современном мире оптимальным решением для таких людей будет бионический протез [25]. Преимущество его перед остальными разновидностями протезов в том, что бионические протезы позволяют людям, оставшимся без ноги или руки, жить полноценной жизнью, а не просто иметь муляж конечности.

Часто из-за преувеличения в средствах массовой информации и фильмах многие люди с ампутированными конечностями возлагают большие надежды на уровень достижений и функций, который может обеспечить протез верхней конечности. Пациенты во многих случаях ожидают футуристических устройств, которые можно будет использовать сразу же при покупке. Хотя протезы технически очень сложны, они гораздо менее продвинуты, чем ожидают инвалиды [32]. В действительности период обучениях использования бионическими протезами довольно долог и не все могут ими пользоваться. Например, при сильных ожогах нервы повреждаются и более не способны проводить электрические импульсы. Выбор лечения уникален для каждого человека, и команда реабилитации должна адаптировать каждое лечение к способностям и предпочтениям пациента, что затягивает процесс реабилитации и возможность использовать бионический протез с миоэлектрическим управлением до полугода.

Еще одним ограничением является количество исполняемых жестов протезом. Управление современными коммерческими протезами заключается в использовании электрокимографических датчиков (ЭМГ электроды) для

установки одной степени свободы (положения) протеза и переключения между степенями за счет различных сокращений мышц пользователя [11]. Сокращения мышц улавливаются системой MPR (система распознавания миоэлектрических паттернов) протеза и в зависимости от распознанного паттерна происходит установка заранее заложенного положения протеза. В основном можно выделить недорогие протезы, которые легкие и компактные, но позволяют лишь ограниченное количество движений (как правило в них устанавливаются 1–2 ЭМГ канала с возможность распознавания 1-3 паттернов), и высококачественные протезы, которые намного сложнее и функциональнее, но также более тяжелые, громоздкие, трудноуправляемые и очень дорогие (4–8 ЭМГ каналов с возможность распознавания 7-19 паттернов) [36].

С каждым годом область протезирования развивается за счет технологических достижения и инноваций. Так, помимо доработок миоэлектрического управления [36, 12, 33, 22, 44] для улучшения точности распознавания и расширения количества паттернов, предпринимаются попытки внедрения бионических протезов на основе управления с помощью мозговых активности [44], либо хирургических вмешательств в нервы человека [37, 26], но для массового клинического использований такие системы все еще не подходят по причинам сложности проведения хирургических операций.

Исходя их вышеизложенных проблем было принято решение разработать аппаратно-программную систему бионического протеза руки человека позволяющую осуществлять управление бионическим протезом на основании комбинированного управления в виде графических интерфейсов, голосового и миоэлектрического управления, а также позволяющие выполнять индивидуальную настройку исполняемых протезом жестов бионического протеза.

Таким образом, целью работы является уменьшение постреабилитационного периода и увеличение количества исполняемых

жестов бионического протеза за счет перепрограммируемого управления и возможности подключения человеко-машинных интерфейсов.

За счет использования дополнительных интерфейсов в виде мобильного приложения человеку будет проще пройти первичный реабилитационный период, в котором ему требуется активная тренировка мышц для использования миоэлектрических датчиков. Пользователь, либо реабилитолог, смогут самостоятельно вносить исполняемые протезом жесты в систему с помощью приложения для персонального компьютера, тем самым увеличив количество возможных принимаемых положений протеза. Так же, система предоставит возможность использования бионического протеза тем людям, чьи нервные окончания были повреждены и более не способны проводить нейронные импульсы, которые и требуются для использования миоэлектрических датчиков.

Задача разработки аппаратно-программной системы бионического является примером создания встраиваемой системы и для достижения поставленной цели требует решения следующих задач:

1) Исследование современных подходов и решений в сфере протезирования верхних конечностей.

2) Проектирование аппаратно-программной системы бионического протеза руки человека.

3) Исследование и разработка протоколов взаимодействия между устройствами системы, включающих форматы передачи данных, управляющий автомат и механизм синхронизации жестов между устройствами.

4) Исследование и разработка алгоритмов обработки миоэлектрических сигналов.

5) Проектирование центрального контроллера системы.

6) Разработка программного обеспечения центрального контроллера системы для обеспечения исполнения жестов по протокольным командам и миоэлектрическим показаниям.

7) Исследование и проектирование структуры жестов системы для обеспечения возможности самостоятельного создания жестов и их исполнения.

8) Разработка пользовательского интерфейса для формирования собственных жестов для персонального компьютера.

Разработанная аппаратно-программная система бионического протеза является уникальной и не имеет аналогов. В качестве основы исследований были использованы работы по миоэлектрическому управления бионическими протезами [36, 33, 22, 14] и работы по проектированию и созданию аппаратно-программного обеспечения бионических протезов [9, 10, 40, 47]. Основными отличительными возможностями, характеризующими научную новизну аппаратно-программной системы, являются:

1) Разработанная система хранения, передачи и исполнения жестов протеза, позволяющая абстрагироваться от реализуемой системы двигательных механизмов и устройств управления и осуществлять создания новых или изменение старых жестов.

2) Комбинированная система управления, основанная на взаимодействии миоэлектрических сенсоров, голосового управления и графического интерфейса, позволяющая существенно расширить набор жестов системы.

Аппаратно-программная система бионического протеза предназначена для людей с отсутствующей конечностью руки до локтевого сустава и может быть применена в области медицинской реабилитации верхних конечностей.

В рамках образовательной программы основные результаты диссертационного исследования были представлены на научно-практических конференциях

1) 11th Majorov International Conference on Software Engineering and Computer Systems на тему «Development of a hardware-software system for a bionic prosthesis of a human hand» (12.12.2019)

2) 12th Majorov International Conference on Software Engineering and Computer Systems на тему «Development of a Linear Actuator Controller For a Hand Prosthesis» (10.12.2020)

По теме диссертации опубликована 1 статья [24] на тему «Development of a hardware-software system for a bionic prosthesis of a human hand», в которой нашли отражение теоретический принципы и результаты работы.

Результаты диссертационной работы включены в отчет этапа 2 о научно-исследовательской работе на тему «Разработка аппаратно-программной системы автоматизированного перепрограммируемого управления протезом руки человека» по программе "УМНИК", договор № 13927ГУ/2019. Работа принята фондом содействия инновациям 11.05.2021 и считается завешенной в полом объеме.

По результатам диссертационной работы подана заявка на регистрацию интеллектуальной собственности в виде регистрации программы на ЭВМ «Программная система автоматизированного перепрограммируемого управления протезом руки человека». Дата поступления заявки в ФИПС: 11.03.2021. Номер заявки: No 2021613235/69.

Также, разработанный бионический протез был ранее апробирован в виде статей [1, 4, 5] и выставок в рамках образовательной программы бакалавриата. Наиболее значимым результатам для проекта является выраженная заинтересованность и одобрение со стороны представителей сообщества инвалидов на финале XI международной олимпиады в сфере информационных технологий «IT-ПЛАНЕТА 2017/18» на которой проект занял первое место в конкурсе «Неограниченные возможности». Проект был отмечен как перспективный и действительно способным внести вклад в сферу протезирования.

1. ОБЗОР БИОНИЧЕСКИХ ПРОТЕЗОВ ВЕРХНЕЙ КОНЕЧНОСТИ

1.1. Состояние сферы протезирования

В США около 2 миллионов человек с ампутациями конечностей, и ежегодно проводится 185000 ампутаций [61]. Согласно статистическим данным по Германии, Италии, Ирландии и США, в ЕС примерно 3,18 миллиона человек с ампутациями конечностей (4,66 миллиона по всей Европе), и около 295 000 ампутаций выполняются каждый год (431 000 по всей Европе) [6]. Это создает огромную медицинскую и экономическую проблему. Ампутации оказывают разрушительное воздействие на здоровье пациентов и вызывают психологический стресс с вытекающими отсюда экономическими потерями и ухудшением качества жизни.

Ампутации и, следовательно, протезы как их самое непосредственное решение, имеют долгую историю, начиная с крючков и других протезов средневековья, до современных роботизированных и бионических конечностей [28, 16, 31, 18, 7, 30, 39], которые являются результатом как медицинского, так и технического прогресса.

В общем виде протезы можно разделить на пассивные и активные. Пассивные протезы служат только для косметических целей и не пытаются восстановить функциональность биологической руки.

Протезы с электроприводами или активные [41] предпочтительны по сравнению с пассивными. Они позволяют выполнять различные захваты и изменения положения, поскольку их движение осуществляется двигателями постоянного тока. Такими протезами можно управлять с помощью различных средств, таких как датчики силы [8], акустические интерфейсы [44] и сигналы ЭМГ [38]. Активные протезы восстанавливают некоторую функциональность людей с ампутированными конечностями.

Ранее их контроль ограничивался только одной или двумя степенями свободы, но в последнее время биомехатронная конструкция этих протезов была усовершенствована, и на рынке появились современные и надежные

протезы с множеством степеней свободы за счет разделения пальцев. Bebionic 3 [51], рука Микеланджело [62] и i-LIMB [58] представляют собой миоэлектрические протезы с управлением на основании ЭМГ сигналов. Они имеют от 7 до 24 различных положений рук. Пальцы независимо приводятся в движение специальными двигателями постоянного тока, и, благодаря своей прочной конструкции, эти руки могут выдерживать до 45 кг. Выделенная беспроводная связь позволяет врачам контролировать эти устройства и программировать зависящие от пользователя параметры, например пороговые значений обнаружения сигнала на мышцах.

Но пациентам все еще трудно полностью вернуться на работу в после реабилитационный период. Обычно они воспринимают протез как инородное тело, к которому сложно привыкнуть, обучиться использованию и который ограничивается заранее запрограммированными жестами, поэтому существует потребность в разработке новых протезных решений, которые должны быть более эффективными и более легкими для обучения пациента.

Человеческая рука способна исполнить репертуар сложных движений, которые позволяют нам взаимодействовать с окружающей средой и друг с другом. Для выполнения движений мы получаем огромное количество информация об окружающей среде, включая прикосновение, вибрацию, боль, температуру и проприоцепцию (ощущение относительного положения частей тела).

Современные бионические протезы рук стараются максимально приблизиться к соответствию естественной конечности как по форме, так и по функциям. Несмотря на это, даже лучшие бионические протезы до сих пор не воспроизводят большинство функций биологической руки и на этой уйдут еще многие десятилетия.

Учёные и инженеры исследует и улучшают бионические конечности в разных направлениях. В конструкторском направлении происходит улучшения качества материалов изготовления, степеней свободы протеза и различных захватов [60]. С медицинской точки зрения, происходит

исследование тактильной и двунаправленной связи, которая позволит человеку ощущать прикосновения за счет стимуляции нервов [25]. В данной работе выбрано направление улучшение аппаратно-программной составляющей бионического протеза с целью увеличения количества исполняемых жестов и упрощения процесса обучения человека протезу. Исходя из этого, в выбранном направлении, бионический протез является примером разработки встраиваемой, либо в более узком плане киберфизической, системы и должен рассматриваться соответствующе.

1.2. Бионические протезы как аппаратно-программная система

Традиционным подходом к управлению протезом является использование электромиографии для распознавания и классификации образов. Современные бионические протезы контролируются с помощью электромиографических (ЭМГ) сигналов, регистрируемых с помощью поверхностных электродов, определяющих электрическую активность, связанную с мышцами руки пациента. Они позволяют интерпретировать произвольные импульсы, действующего на руку, путем соответствующего сокращения мышц.

Достижение конца 20 и начала 21 века в развитии интегральной схемотехники, 3-D печати, малоразмерных и мощных приводов, а также емких и компактных аккумуляторов, обеспечили скачек использования бионических протезов на замену механическим аналогам. С каждым годом область разработки бионических протезов развивается, уже сейчас имеются футуристические устройства, которые помимо установки степеней свободы на основании приема и обработки ЭМГ сигнала, позволяют выполнять мониторинг процесса реабилитации и повседневного использования [17], проводить NFC оплату [49], взаимодействовать с протезом с помощью мобильного телефона [20].

Подобные достижения становятся доступны благодаря возрастающей технологической составляющей протеза. Если раньше протез содержал только систему распознавания миоэлектрических образов, то теперь, уровень

современного развития интегральных схем и вычислительных мощностей, позволяет добавлять все больший функционал. Современные бионические протезы, в общем виде, представляют из себя набор из трех компонентов [2] (рисунок 1):

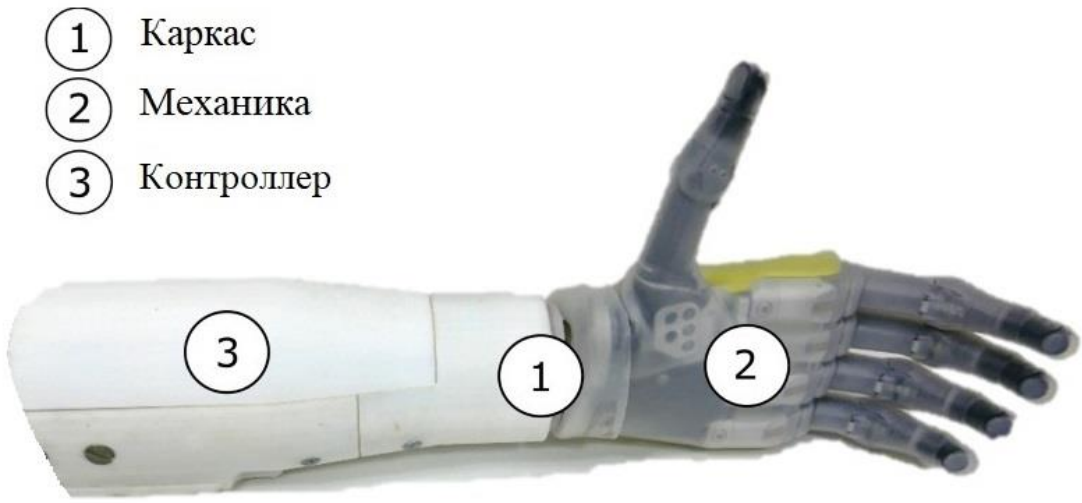

Рисунок 1 - Состав бионического протеза руки

1) Каркас. Корпус протеза, произведенный из лёгких металлических сплавов, чаще всего титана. В последнее время, с развитием 3D-печати все чаще начинает изготавливаться из пластика для удешевления конструкции. Обеспечивает антропоморфность конструкции для имитации биологической руки и защищает внутренние компоненты протеза от повреждений. Имеет культеприемную гильзу, изготавливаемую индивидуально. В культеприемник встроены ЭМГ электродами, максимально близко прилегающими к мышцам человека. При начале использования протеза пользователь помещает культю в гильзу.

2) Механика. Набор шарниров, тяг и электрических приводов, обеспечивающих подвижность протеза. Примитивные версии протезов содержат всего один привод, который позволяет только сжимать и разжимать кисть целиком. Как, было отмечено, в более современных версия доступно изменение положения отдельных пальцев. Таким образом, количество подвижных частей в современных протезах обеспечивается как минимум

пятью приводами на каждый палец и сложной рычажной системой. В дополнение встречаются производители, которые добавляют возможность поворота кисти, что так же требует добавление поворотного механизма.

3) Контроллер. Электронная начинка протеза, представляющая из себя электрическую схему на базе микропроцессора. Обеспечивает обработку ЭМГ показаний и выполняет установку позиций приводов. В зависимости от сложности устройства может обеспечивать выполнение других функций, таких как взаимодействие с внешними устройствами, вывод информации, обработку показаний сторонних датчиков.

В соответствии с данной структурой рассмотрим бионический протез проекта «EMG based Hand Gesture Recognition on Embedded Low Power Platforms» [35] от команды под руководством Simone Benatti.

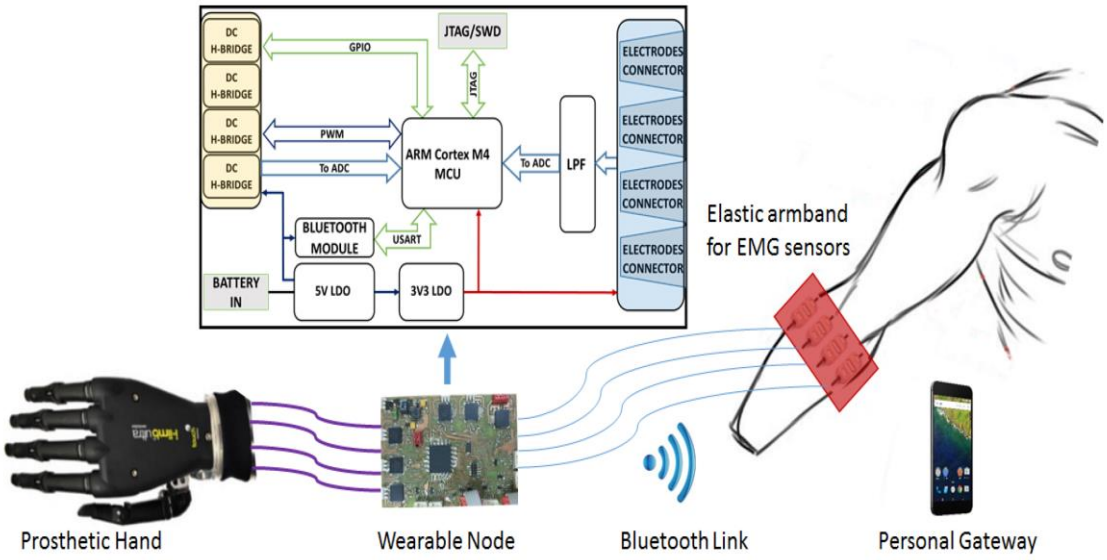

Рисунок 2 - Диаграмма архитектуры системы проекта «EMG based Hand Gesture Recognition on Embedded Low Power Platforms» [35]

На рисунке 2 представлена заявленная проектом архитектура протеза, которая состоит из: (1) эластичной повязки с 4 датчиками ЭМГ; (2) носимого узла (контроллер), отвечающего за сбор и классификацию данных, приведение в действие приводов протеза и связь через Bluetooth; (3) полиартикулярного (механика) протеаза руки (каркаса) и (4) персонального устройства для сбора данных, обучения алгоритму распознавания и настройки параметров системы.

Основной интересен в рамках встраиваемой системы представляет носимый узел (контроллер), структурная схема которого была выделена авторами проекта (рисунок 3).

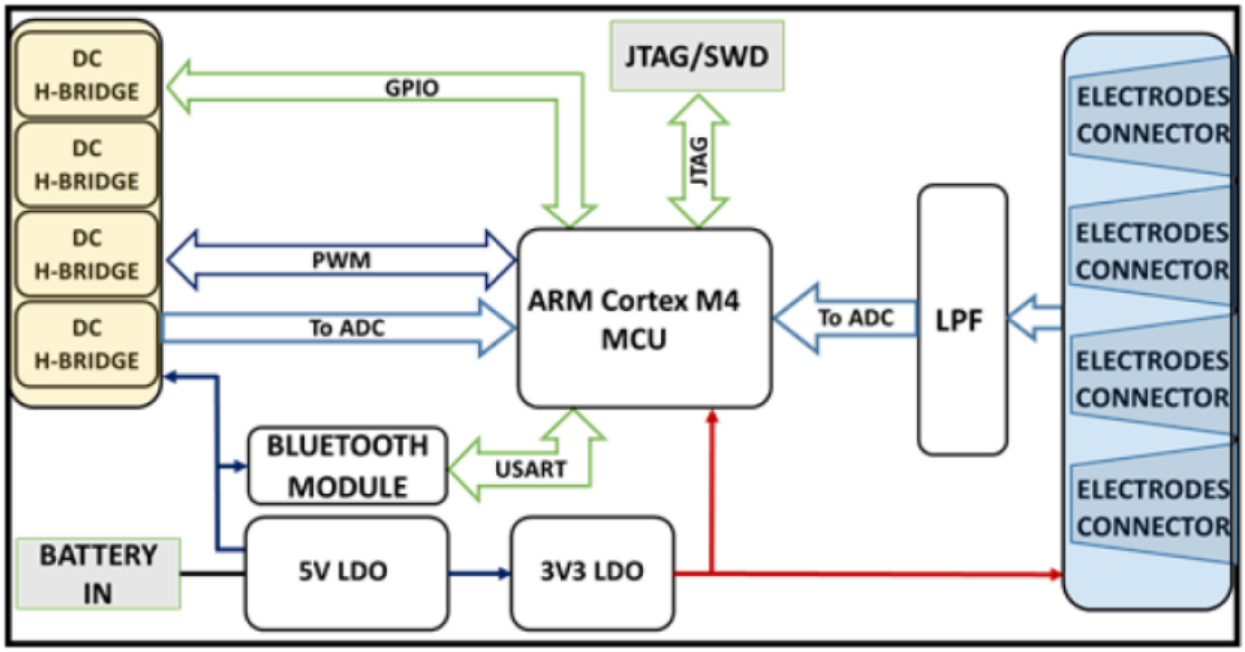

Рисунок 3 – Структурная схема контроллера в проекте «EMG based Hand Gesture Recognition on Embedded Low Power Platforms» [35]

В рассматриваемой системе контроллер представляет из себя электрическую схему на базе микроконтроллера с процессором ARM Cortex M4. Для понимания принципов ее работы следует рассмотреть схему детальнее.

Схема разработана на 4-слойной печатной плате и включает 32-битный микроконтроллер для получения сигнала ЭМГ и управления приводами, а также включает приемопередатчик Bluetooth для связи с внешними устройствами.

Система основана на микроконтроллере ARM Cortex M4 от NXP. Наличие двух независимых 16-битных SAR АЦП (АЦП0 и АЦП1) и выделенного выходного ШИМ периферийных позволяют управлять приводами и считывать данные с 4-х каналов непосредственно периферийными устройствами контроллера, что сводит к минимуму необходимость внешних компонентов и, следовательно, сложность платы.

Данные собираются с частотой 500 Гц. На каждом канале RC-фильтры нижних частот минимизируют высокочастотный электрический шум. Дополнительный резистивный делитель напряжения защищает входы АЦП, ограничивая диапазон сигнала до диапазона 0–3,3 В, в то время как диапазон выходного сигнала электрода составляет 0–5 В. Программное обеспечение микроконтроллера выполняет классификацию входного сигнала SVM сетью.

Двигатели постоянного тока, приводящие в действие движение пальца искусственной руки, управляются драйвером H-моста, расположенном на схеме и подключенным к микроконтроллеру. Обратную связь возвращается к MCU в виде напряжения, которое пропорционален потребляемому току мотора.

Движение пальца считается завершенным, когда он полностью открыт, полностью закрыт или если он остановился при захвате объекта. В таких ситуациях двигатель постоянного тока увеличивает потребление тока, измеряемого через шунтирующий резистор, и используется пороговый триггер для остановки двигателя.

Аппаратная конфигурация завершается приемопередатчиком Bluetooth (SPBT2632C2A от STM), позволяющим системе связываться с другими устройствами на теле пользователя или поблизости. Этот двунаправленный беспроводной интерфейс используется для потоковой передачи полученных данных ЭМГ на ПК или для сохранения на устройстве индивидуальных параметров и настроек. Потоковая передача данных на хост-устройство используется для тестирования системы и получения экземпляров жестов для автономного анализа и для обучения алгоритма классификации. Обученные модели распознавания и дополнительные системные настройки затем отправляются обратно на встроенное устройство. Потоковая передача осуществляется по протоколу, который был разработан авторами проекта.

Рассмотрев данную схему, можно заметить в ней недостатки. Так, например, использование электрических моторов с обратной связью в виде тока потребления не дает возможности устанавливать точные положения

протеза, а гарантирует только две степени свободы для каждого пальца с возможностью остановки при захвате объекта. Но рассмотренная схема явно дает понять задачи, которые встали при разработке у авторов решения:

1) Управление приводами.
2) Прием и обработка показаний датчиков.
3) Классификация паттернов.
4) Взаимодействие с внешними системами.
5) Создание каркаса и механики протеза.

Данные задачи являются отражением поставленных задач работы во введении, что позволяет судить, что работа проводиться в верном направлении.

2. ПРОЕКТИРОВАНИЕ АППАРАТНО-ПРОГРАММНОЙ СИСТЕМЫ БИОНИЧЕСКОГО ПРОТЕЗА РУКИ

2.1. Проектирование архитектуры бионического протеза

В разрабатываемом решении, достижение поставленной цели увеличения принимаемых положений и упрощения процесса обученные, ожидается добиться за счет возможности самостоятельного создания принимаемых положений протеза пользователем, либо реабилитологом и их установки на основании как классического подхода использования миоэлектрического управления, так и управления с пользовательских устройств, например приложения для мобильного телефона.

Такое решение позволит пользователю не ограничиваться набор запрограммированных производителем протеза положений, а создавать свои решения, которые будут применимы индивидуально для его потребностей. Пользователь сможет устанавливать положения с внешних устройств во время постреабилитазионного периода, что позволит использовать бионический протез уже в первые часы после начала использования, в отличии от миоэлектрического управления. Со временем, при развитии мышц руки, либо проведения хирургических операций, пользователь сможет также использовать и миоэлектрическое управление, которое также присутствует в разрабатываемом решении.

Для обеспечения заявленного поведения требуется разработка каждого из трех компонентов, составляющей бионический протез, а именно корпуса, механики и контроллера. В данной работе основное упор сделан на контроллер и внешние интерфейсы взаимодействия по причине рассмотрения бионического протеза как встраиваемой системы, но следует рассмотреть и остальные компоненты, так как они являют важную роль в протезе.

2.1.1. Корпус и механика протеза

Для разрабатываемой системы была спроектирован корпус (рисунок 4) на основе модели кисти компании "Youbionic" [71].

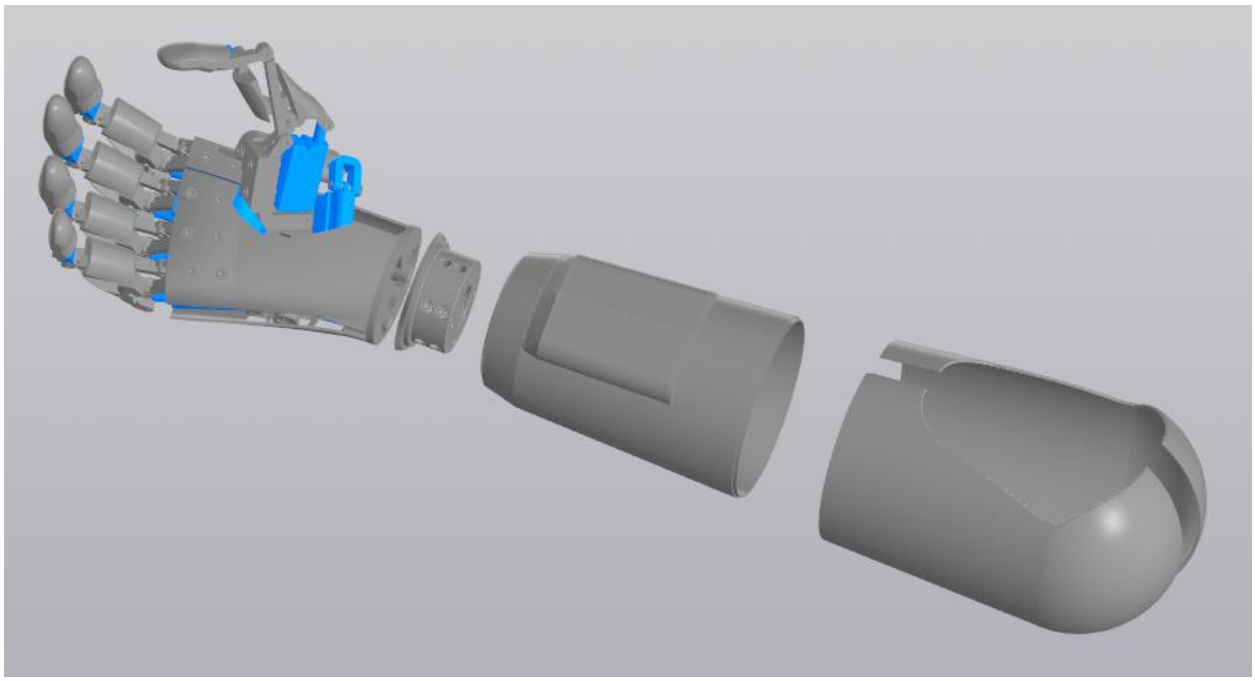

Рисунок 4 – Разработанная 3D модель протеза

Макет протеза можно представить как основную группу мышц и сухожилий (рисунок 5) необходимых для выполнения базовых функций (сжатия и разжатия), как всей руки, так и каждого отдельного пальца в целом.

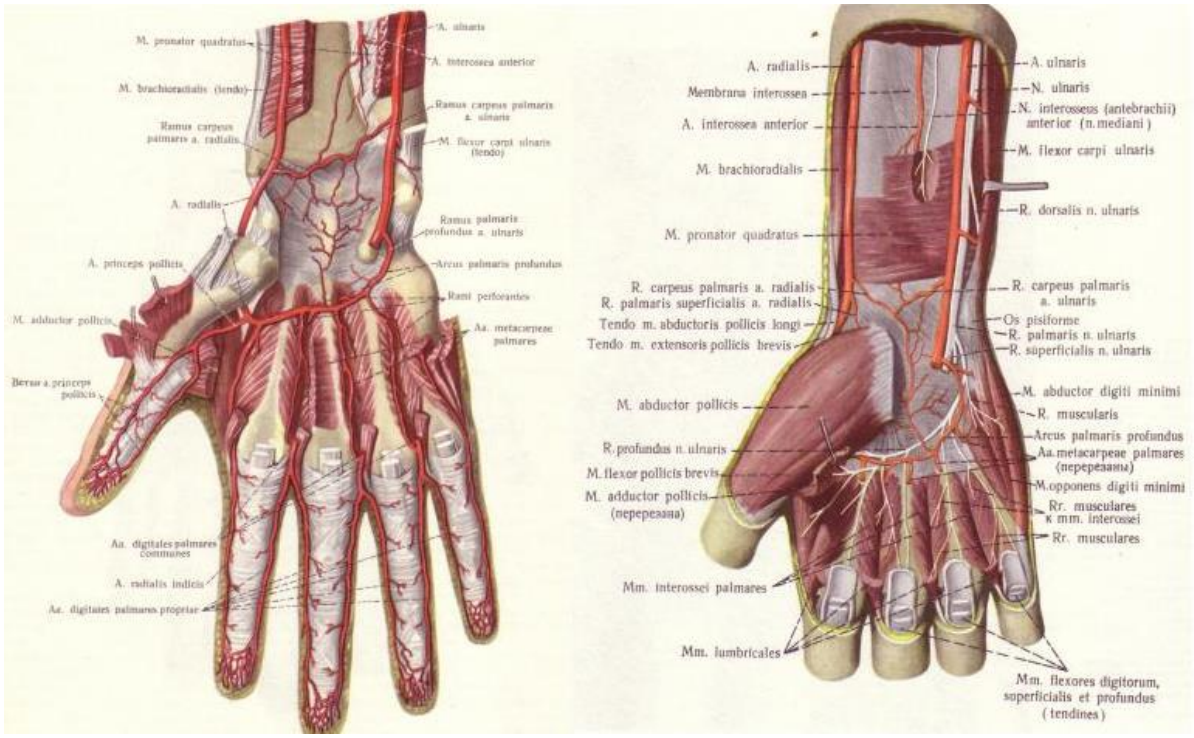

Рисунок 5 – Мышцы и сухожилия руки человека [21]

Модель представляет следующие группы мышц [21]. Переднюю группу мышц предплечья:

- поверхностный сгибатель пальцев - сгибает средние фаланги второго-пятого пальцев (первый считается большой палец, 5 мизинец);
- глубокий сгибатель пальцев - сгибает дистальные (конечные) фаланги второго-пятого пальцев;
- длинный сгибатель большого пальца кисти - сгибает большой палец и кисть.

Задняя группа мышц предплечья, разгибающая пальцы:

- разгибатель пальцев - разгибает второй-пятый пальцы кисти;
- разгибатель мизинца;
- короткий разгибатель большого пальца кисти - разгибает проксимальную (ближнюю) фалангу большого пальца;
- длинный разгибатель большого пальца - разгибает большой палец;
- разгибатель указательного пальца - разгибает большой палец.

Мышцы самой кисти:

- короткий сгибатель большого пальца;
- короткий сгибатель мизинца;
- червеобразные мышцы - четыре штуки для второго-пятого пальцев (сгибают их проксимальные фаланги и разгибают средние и дистальные).

Только благодаря столь сложному механизму осуществляется сжатие биологической руки человека (в учет не взяты мышцы, сухожилия и сочленения необходимые для выполнения более сложных действий/жестов). В макете количество элементов сведено к минимуму, задняя и передняя группа мышц была воспринята как единое целое и реализована в едином корпусном элементе (рисунок 6), а на основе мышц сгибателей и разгибателей была реализована шарнирная система (соединения элементов 1–3) с аналогом червячной мышцы (элемент 4). Аналогичным образом реализованы все

пальцы макета протеза, за исключение большого пальца (он имеет дополнительную ось вращения для реализации различного рода хватов). Благодаря подобной системе пальца имеется возможность принимать большое количество положений за счет размещения линейного привода с обратной связью в пространстве элемента 3 и прикрепления его движущего механизма к элементу 2. Так, в протезе используется 6 линейных приводов Actuonix PQ12-P с обратной связью в виде напряжения, пропорционального положению хода привода.

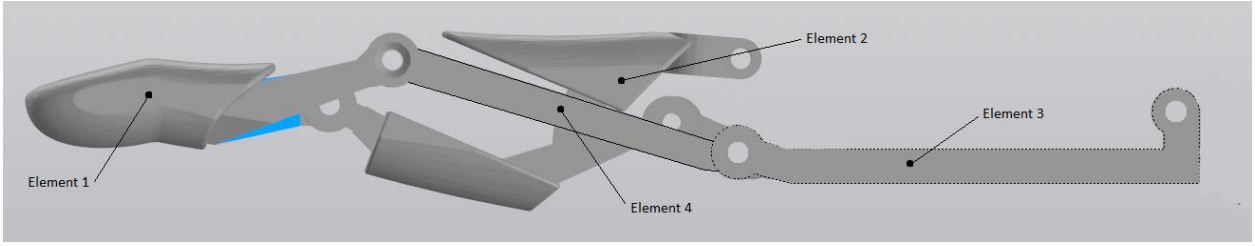

Рисунок 6 – Элементы пальца протеза

Важнейшим элементом макета протеза является область предплечья. Она позволяет заменять капсулу культиприемника индивидуально под каждого пользователя, что делает изделие доступным для всех. Проектирование модели производилось с использованием системы проектирование трехмерных объектов - "КОМПАС 3D v18".

Ранее приведенная модель была распечатана на 3D-принтере, а в основу печати был взят PLA пластик (полилактид). Он достаточно прост в обработке, материал не токсичен (в основе лежат натуральные, биоразлагаемые компоненты) и малоподвержен деформации (имеет плотность 1,25 г/см³).

Из основных настроек 3D-принтера можно выделить:

- скорость перемещения головки принтера - 60 мм/с;
- заполненность детали пластиком - 100%;
- высота слоя - 0.1 мм;
- толщина сопла - 0.4 мм;
- температура сопла - 230 С°;
- температура стола печати - 60 С°.

Именно они существенным образом влияют на время, затраченное на печать изделия. Суммарно печать модели составила 104.5 часа, а вес модели без электронных компонентов равен 800 граммам.

2.1.2. Структура исполняемых жестов

Для обеспечения возможности самостоятельно создания и настройки положений протеза требуется предоставить пользователю, либо реабилитологу, простую структуру настройки [4], поля которой он сможет настраивать для получения ожидаемого поведения протеза.

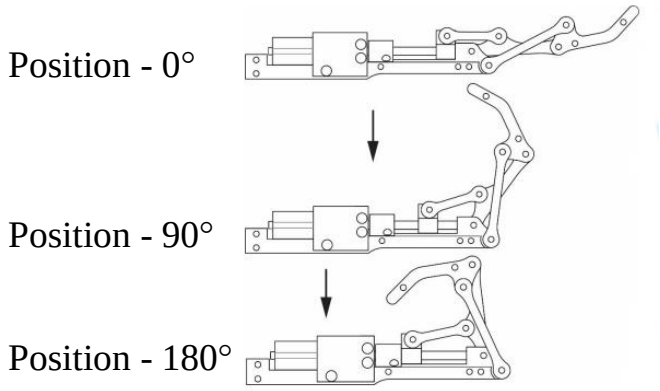

Рисунок 7 - Иллюстрация позиции пальца протеза в градусном представлении

Так, палец протеза может быть представлен в виде угла от 0 до 180 градусов, где 0° – выпрямленный палец, а 180° – согнутый (рисунок 7), а набор позиций пальцев и их комбинаций можно принять за жест руки. Подобных набор комбинаций позволяет реализовывать большинство положений руки, необходимых в повседневной жизни. Пользователь сможет самостоятельно одеваться, брать мелкие предметы, открывать двери, а также сможет выполнять наборы жестов, наподобие рукопожатия и приветствия.

Для обеспечения исполнения различных жестов составлена структура (рисунок 8), которая учитывает различные запросы пользователя к применяю жестов в повседневной жизни.

Структура жест протеза (Gesture) включает в себя уникальный идентификатор (id), информацию о жесте (info) и список положений пяти пальцев протеза в угловом соотношении (actions).

Информация о жесте (GestureInfo) позволяет пользователю настраивать: (1) названия жеста (name) для отображения в графическом интерфейсе и для исполнения с помощью голосового управления; (2) итеративность жеста

(iterable) – бесконечное повторение жеста до приема нового на исполнение; (3) количество повторений (repetitions) – ограниченное количество повторов жеста.

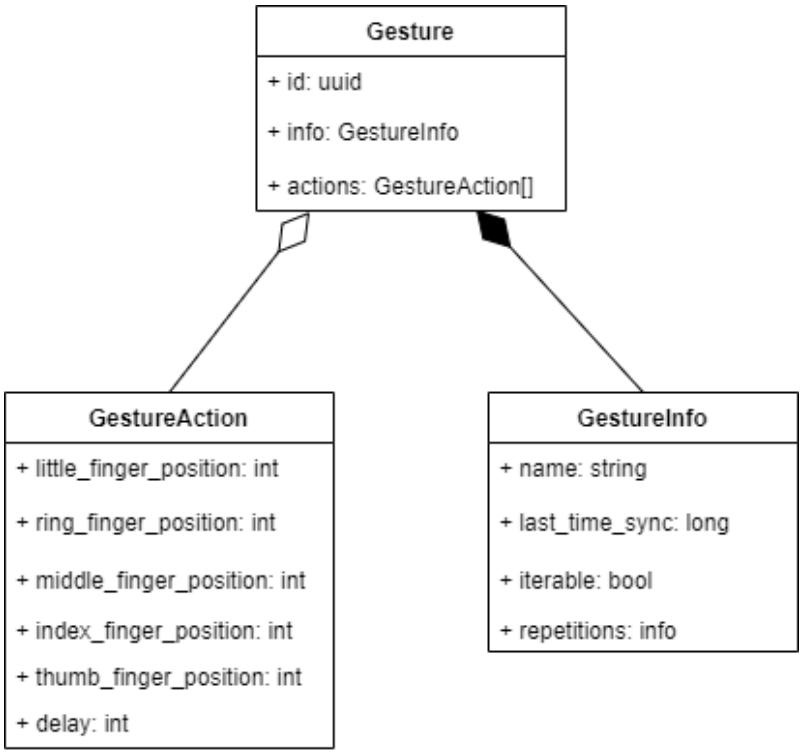

Рисунок 8 – Диаграмма структуры жеста протеза

Коллекций позиций жеста (GestureActions) позволяет выполнять установку набора положений пальцев протеза через временной промежуток (delay), хранимый в миллисекундах.

Используя данную структуру, пользователь сможет настраивать различные комбинации положений пальцев протеза под свои нужды.

2.1.3. Контроллер протеза

Основным компонентов в бионическом протезе является контроллер, который обеспечивает управление приводами протеза, обработку показаний датчиков и обмен данными с внешними устройствами. Ранее, в главе 1.2, был рассмотрен основной состав контроллера бионический протеза на примере проекта «EMG based Hand Gesture Recognition on Embedded Low Power Platforms», который отражает перечень основных аппаратно-программных компонентов, необходимый для обеспечения полноценной работы протеза, а именно:

1) Драйверы приводов для движения электрических моторов.
2) Микропроцессор для обработки данных.
3) ЭМГ электрод.
4) Модуль связи с персональными устройствами.

Для разрабатываемого бионического протеза была спроектирована близкая по концепции структура, за исключением некоторых компонентов. Структурная схема протеза приведена в Приложении 1, уменьшенную версию можно рассмотреть на рисунке 9.

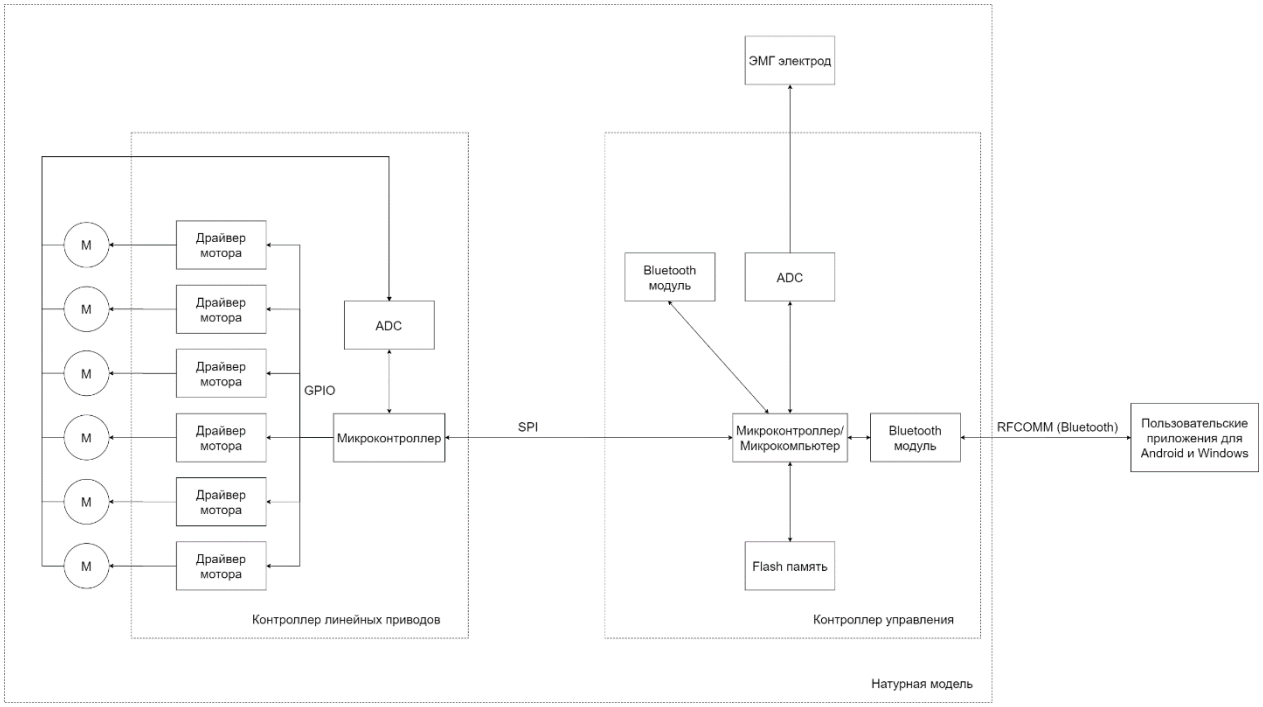

Рисунок 9 – Структурная схема бионического протеза руки

В первую очередь было принято решение разделить контроль над приводами и остальной логикой протеза (обработка ЭМГ показаний, взаимодействие с внешними устройствами) по причине того, что точечная установка положений с точностью до 5° на основании обратной связи приводов является ресурсоемкой операций, которая может значительно влиять на частоту дискретизации и обработки ЭМГ сигнала. В дополнении подобное решение позволяет сделать систему модульной и абстрагировать логику управления протезом от конкретных приводов, что позволит выполнять легкую замену механики устройства, например перейти с линейных приводов на сервоприводы, либо электрические моторы.

Для разделения управления линейными приводами система была разделена на два контроллера:

1) Контроллер линейных приводов – обеспечивает управление механической составляющей протеза в зависимости от используемых в протезе моторов. Так, для пяти пальцев кисти с механикой, построенной на линейных приводах, используются шесть Н-мостов (большой палец требует два привода) и шестиканальное АЦП для получения обратной связи в виде позиций приводов. Обработку обратной связи и управление приводами осуществляет микроконтроллер, принимающий внешние команды по SPI. Протокол, который должен поддерживать контроллер приводов для работы в составе системы рассмотрен в главе 2.3.2.

2) Контроллер управления – обеспечивает работу миоэлектрического управления на основании принимаемого ЭМГ сигнала с помощью АЦП. Беспроводной интерфейс в составе контроллера предназначен для взаимодействия с внешними устройствами, которые должны поддерживать протокол взаимодействия, описанный в главе 2.3.1. В дополнении к классическим компонентам контроллера протеза, на контроллере управления расположено постоянно запоминающее устройства для хранения создаваемых пользовательских жестов. Создание пользовательских жестов обеспечивается на внешних устройствах, которые в рамках системы представлены приложениями для персонального компьютера, либо мобильном телефона. В дополнении, мобильного приложение позволяет исполнять жесты с помощью графического, либо голосового управления. Так, максимальное количество исполняемых протезом жестов, ограничивается только объемом ПЗУ системы.

В последующих главах будет исследован и рассмотрен аппаратно-программный состав контроллера управления протеза и способы взаимодействия с ним с помощью программных протоколов. Также будет рассмотрено приложение для персонального компьютера, позволяющее выполнять конфигурацию жестов протеза.

2.2. Проектирование схемотехнической части центрального контроллера бионического протеза

Аппаратное и программное обеспечение – два основных элемента, составляющих вычислительную систему. Тенденции в области аппаратного обеспечения сильно влияет на приложение, которое будет построено на нем [42]. Производительность программного обеспечения может напрямую выиграть от выбора более мощного оборудования, но оно также ограничено его характеристиками. Следовательно, при разработке вычислительных систем приходится идти на компромиссы. Особенно это становится заметно при разработке встраиваемых систем.

Разработчики не могут изменять или обновлять уже готовую встроенную систему. Поэтому она должна быть предназначена для конкретной задачи и отличаться высокой стабильностью, надежностью и удовлетворять конструктивным ограничениям [57].

Для разрабатываемого бионического протеза аппаратное обеспечение должно обеспечивать:

1) Хранение пользовательских жестов протеза и конфигурации.
2) Получение и обработку миоэлектрических показаний с внешнего ЭМГ электрода.
3) Прием и обработку протокольных команд по Bluetooth.
4) Взаимодействие с контроллером линейных приводов по SPI.

Исходя из требований следует, что бионический протез должен содержать в себе следующие компоненты:

1) Постоянное запоминающее устройство для хранения конфигурации и команд.
2) Аналого-цифровой преобразователь для приема ЭМГ сигнала с электрода.
3) Модуль для приема и отправки данных по Bluetooth.
4) SPI модуль для взаимодействия с контроллером приводов.
5) Микропроцессор для обработки данных протезом.

При начале разработки системы следует тщательно подойти к выбору основного компонента – микропроцессора, так как он накладывает ограничения на остальные аппаратные устройства и программное обеспечение, которое будет написано под него.

Встраиваемые системы базируется на двух типах устройств, построенных на микропроцессорах, а именно на микроконтроллерах и микрокомпьютерах [45]. Различия между ними существенны как в производительности, так и в архитектуре.

Для разрабатываемого протеза следует рассмотреть возможность создания аппаратного обеспечения на основании наиболее яркий представителей каждой из отраслей. В качестве микроконтроллера был выбран STM32F407VET6 из семейства STM32, а в качестве микрокомпьютера был выбран Raspberry Pi Zero W. Каждый представитель имеет большое сообщество из разработчиков, внедряющих данные устройства в свои проекты, в том числе и в области протезирования [13, 34].

Если не брать в расчет архитектурные различные устройств, то, с точки зрения разработки бионического протеза, основные различия между ними заключаются в производительности, наличии периферийных устройств, наличии операционной системы и ограничениях на выбираемый язык программирования (таблица 1).

Таблица 1

Основные различия между STM32F407VET6 и Raspberry Pi Zero W при разработке бионического протеза

Характеристика	STM32F407VET6	Raspberry Pi Zero W
Частота процессора	168 MHz	1 Ghz
Объем оперативной памяти	192 kB	256Mb
Наличие операционной системы	-	Unix-like OS

Наличие Bluetooth модуля	-	+
Наличие SPI модуля	+	+
Поддержка файловой системы	Сложная интеграция с Flash память с помощью SDIO, либо SPI интерфейса и библиотеке FAT32	Простая интеграция с Flash памятью путем работы через операционную систему
Наличие встроенного АЦП	АЦП 12-бит	-
Ограничение по языкам программирования и фреймворкам	Язык: Assembler, C/C++ Фреймворки: имеются HAL библиотеки для работы с внутренними периферийным устройствам контроллера, но для реализации всей функциональности протеза потребуется написание большого количества библиотек	Язык: за счет Unix-like системы, большое количество языков, наиболее популярен Python Фреймворки: большое количество пакетов для работы с периферией на уровне микропрограмм и HAL, большое количеств фреймворков для написания ПО на уровне приложения

Можно заметить, что STM32 и Raspberry Pi имеют приблизительно равное число достоинств и недостатков. Так, например, наличие операционной системы на Raspberry Pi лишь отчасти является достоинством в упрощении написания ПО и работой с файловой системой, ведь обратной стороной является повышенное энергопотребление и расход процессорного времени и оперативной памяти на работу операционной системы. А на STM32

довольно малый объем оперативной памяти и отсутствие большого количества готового ПО для ускорения разработки.

Исходя из того, что явного кандидата путем прямого сравнения выявить сложно, то рассмотрим разработанные в рамках работы электрические принципиальные схемы контроллера управления при реализации и на STM32F407VET6 (Приложение 2) и на микрокомпьютере Raspberry Pi Zero W (Приложение 3).

STM32F407VET6 (D1) имеет встроенное АЦП и SPI, что позволяет подключить ЭМГ электрод 13E200 MyoBock (X2) и контроллер приводов напрямую (X1). Но отсутствие большого объема ПЗУ и встроенного Bluetooth модуля сильно сказывается на сложности схемы. Так, для увеличения объема ПЗУ используется Flash карта объемом 16GB (D4), которая соединяется с STM32 через схему электромагнитного фильтра CM1624 (D3). В качестве схемы Bluetooth модуля используется схема HC-06 (D2), которая соединяется с контроллером по UART. Таким образом, помимо заявленного SPI интерфейса, на STM32 задействуются дополнительно SDIO и UART интерфейсы. Дополнительной сложностью при разработке схемы является еще и большое количество компонентов для обеспечения стабильного питания, которым требуется выполнить монтаж на плате (C1-C29, R1-R9 и др.).

На фоне STM32F407VET6, принципиальная схема для системы на Raspberry Pi Zero W (D1) значительно проще в реализации. За счет того, что Raspberry Pi Zero W представлена уже готовым компонентом, содержащим цепи питания, разъем карты памяти, SPI и Bluetooth модуль, то для покрытия всех аппаратных требований необходимо только добавление АЦП модуля, который в данном случае представлен готовым модулем ADS1115 (D2).

Можно сделать вывод, что схема на базе микроконтроллера STM32F407VET6 будет иметь повышенную надежность за счет более габаритной схемы питания, но схема может быть изготовлена только в производственных условиях. Для начальных разработок эффективным

решением будет является использование схемы на базе микрокомпьютера Raspberry Pi Zero W, которое обеспечит и упрощенно создание схемы устройства и уменьшит затраты на разработку ПО.

2.3. Проектирование интерфейсов и протоколов взаимодействия устройств системы

2.3.1. Интерфейс взаимодействия внешних устройств с протезом

Взаимодействие пользователя с протезом осуществляется за счет использования двух приложения. Первым является десктоп-приложение HandControl, которое предоставляет пользователю возможность перепрограммирования жестов, осуществляемых протезом руки. Вторым – Android-приложение, которое выступает устройством управления протеза руки. В нем заложена отправка запроса на исполнение жестов посредством выбора в графическом интерфейса, а также реализация запросов посредствам голосового управления.

Для обеспечения взаимодействия приложений с контролером управления требуется обеспечить передачу исполнительных команд из пользовательских приложений на бионический протез. В связи с тем, что интерфейс передачи интегрируется в бионический протез, то на него накладываются следующие требования:

1) Интерфейс должен быть беспроводным, так как протез является носимым устройством и использование проводной связи будет доставлять дискомфорт пользователю.

2) Интерфейс должен быть доступен на мобильном телефоне и персональном компьютере, так как использование, либо разработка дополнительных модулей влекут за собой дополнительные временные и экономические расходы, а также усложняют взаимодействие пользователя с системой.

3) Интерфейс должен обеспечивать передачу данных на расстояние не менее пяти метров.

Проанализировав требования к интерфейсу передачи данных можно сразу отметить, что проводные интерфейсы, в частности USB и Ethernet интерфейсы, который доступны на заявленных устройствах не следует

использовать в протезировании, так как это повлечет серьезные неудобства пользователя при использовании протеза (наличие проводов на теле человека).

Беспроводными интерфейсами, доступными на устройствах являются Bluetooth и WiFi интерфейсы [76]. Адаптеры данных интерфейсов доступны на большинстве современных стационарных ПК [70], на ноутбуках [72] и на мобильных телефонах [54].

Для окончательного выбора стоит сравнить Wi-Fi и Bluetooth между собой по наиболее важным критериям, таким как скорость, потребляемая мощность и др. [55]. Сравнение приведено в таблице 2.

Таблица 2

Сравнение Bluetooth и WiFi интерфейсов

Критерий	Bluetooth	Wi-Fi
Топология	Двухточечная для Bluetooth Classic; Двухточечная, широковещательная, ячеистая для Bluetooth BLE;	Звездообразная
Потребляемая мощность	50μW на 10 сообщений для BLE	500μW на 10 сообщений
Скорость передачи	1 Мбит/с	более 100 Мбит/с
Расстояние передачи в условиях местности без преград	10м при минимальном потреблении, 100м при максимальном	До 150м

Как можно видеть из таблицы 2, Bluetooth обеспечивает более долгое время работы устройства без подзарядки. Именно поэтому, Bluetooth интерфейс часто используется в устройствах и IoT решениях с внешним аккумулятором, к которым и относится бионический протез. Так же, в связи с тем, что к протезу имеется необходимость подключатся с пользовательского устройства, то использование двухточечной топологии является лучшем

решением по причине отсутствия необходимости концентратора Wi-Fi. При наличии отдельного концентратора у пользователя была бы необходимость проводить его настройку и отсутствовала бы возможность перемещения на длительные расстояния. Либо, при использовании протеза в качестве концентратора, была бы необходимость входить в сеть протеза, что повлекло бы собой отключение от WAN сети при ее наличии.

Для реализации приемо-передачи на контроллере управления, представленном Raspberry Pi Zero W, используется Bluetooth v4.1 с поддержкой Low Energy, который работает в Bluetooth Dual Mode и обеспечивает надежность физического и логического соединения, а также непосредственную передачу, с помощью двух функциональных уровнях стека [53] (рисунок 10):

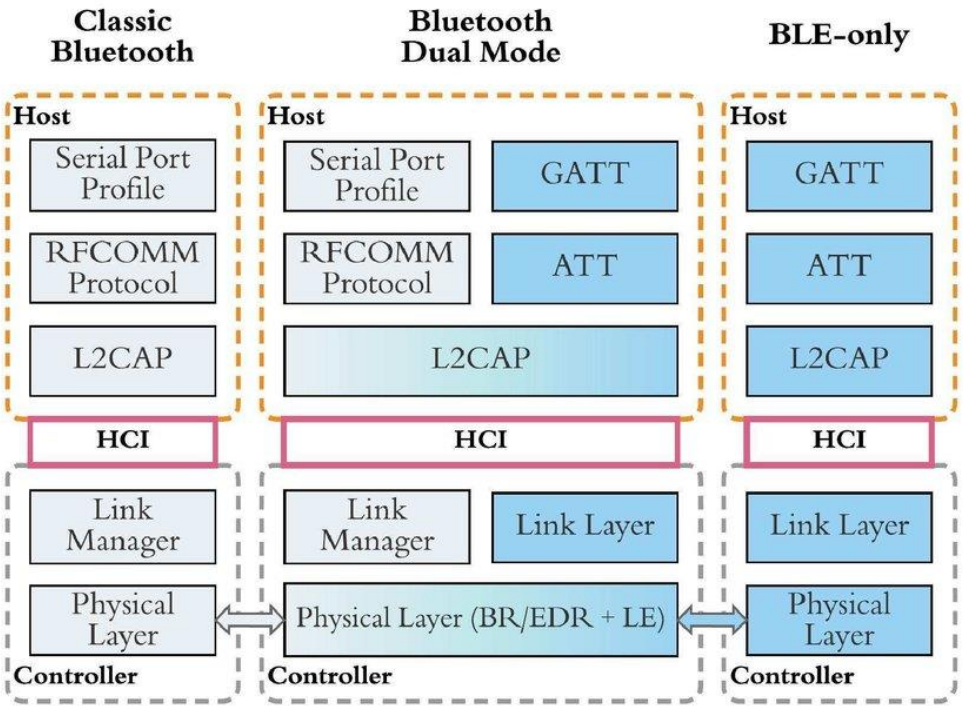

Рисунок 10 – Стек протоколов Bluetooth [53]

Нижние уровни — это основные базовые спецификации, описывающие, как работает Bluetooth. Основой стека протоколов Bluetooth является уровень радиосвязи. Уровень радиосвязи описывает физические характеристики трансивера. Он отвечает за модуляцию/демодуляцию данных для передачи или приема на радиочастотах в диапазоне 2,4 ГГц. Он разделяет полосу

передачи на 79 каналов и выполняет быстрое скачкообразное изменение частоты (1600 скачков в секунду) в целях безопасности.

Выше уровня радиосвязи находится протокол основной полосы частот и протокол контроллера/диспетчера каналов (Link Manager). Он транслирует команды интерфейса хост-контроллера (HCI) из верхнего стека, а также устанавливает и поддерживает связь. Он отвечает за управление соединением, обеспечение равноправия между ведомыми устройствами в пикосети и обеспечивает управление питанием.

Верхние уровни стека состоят из спецификаций профилей, которые фокусируются на том, как создавать устройства, которые будут взаимодействовать друг с другом, используя Bluetooth.

Интерфейс хост-контроллера (HCI) служит интерфейсом между программной частью системы и оборудованием (т. е. драйвером устройства).

Уровень L2CAP (протокол управления и адаптации логических каналов) находится выше HCI в верхнем стеке. Помимо взаимодействия с HCI, он играет центральную роль в связи между верхним и нижним уровнями стека Bluetooth. Он отслеживает, откуда приходят пакеты данных и куда они должны идти. Это обязательная часть каждой системы Bluetooth.

В связи с тем, что какого-то конкретного профиля для работы с протезами не существует и разрабатываемый протез, представленный контроллером управления, имеет собственный функционал, то оптимальный решением является разработка протокола программного уровня протеза поверх протокола RFCOMM, выполняющего эмуляцию последовательного порта в сети Bluetooth.

Был разработан протокол программного уровня на основе модели запрос-ответ. При реализации использовались такие понятия, как ведущее устройство, инициирующее отправку пакетов, и ведомое, которое отвечает на полученные данные. Также предусматривается возможность отправки с ведомого устройства данных без запроса - телеметрии протеза. Такой пакет имеет специальный тип и не должен восприниматься как ответ на запрос, а

должен обрабатываться отдельно. Передача пакетов осуществляется фреймами (рисунок 11) со следующим содержимым:

SFD	**Тип**	**Размер**	**Данные**	**CRC8**
8 байт	1 байт	2 байта	0–65536 байт	1 байт

Рисунок 11 - Формат передаваемых пакетов

- SFD (Start of Frame Delimiter) - разделитель пакетов, определяет начало нового пакета. Для определения начала фрейма используются 8 байт: {0xFD, 0xBA, 0xDC, 0x01, 0x50, 0xB4, 0x11, 0xFF} Для начала приема нового пакета следует определять SFD и начинать прием только после его успешной проверки.
- Тип – тип передаваемого пакета. Значение этого поля определяет, как должны обрабатываться данные пакета, и какая команда должна быть выполнена протезом.
- Размер – определяет размер поля данных пакета. В значение данного параметра не входят размеры других полей, кроме поля данных. Размер может быть нулевым, тогда поле данных отсутствует. Размер записывается с использованием little endian порядка байт.
- Данные – поле с данными пакета. Их формат зависит от типа передаваемого пакета. Если поле размер имеет нулевое значение, поле данных отсутствует.
- CRC8 – циклически избыточный код, рассчитываемый для всех полей. При приёме это поле должно соответствовать полученным данным, иначе пакет отбрасывается, а источнику направляется соответствующее сообщение.

В связи с тем, что контроллер управления представлен одноядерным микрокомпьютером, то введено ограничение на единовременную отправку только одного пакета в системе. Это значит, что ведущее устройство не должно отправлять новый пакет, пока не получит ответ от ведомого на

предыдущий запрос. Если ведущее устройство не получило ответ в течение 5 секунд, то считается, что ведомое устройство не способно обработать пакет и возвращается ошибка таймаута.

Для исполнения протокольных команд протеза было составлено API протеза, приведенное в таблице 3, со следующими полями:

- Запрос - название предоставляемой команды.
- Код запроса - идентификатор, размещаемый в поле типа. Код ответа должен соответствовать коду запроса.
- Данные - Полезная нагрузка запроса. Как правило является protocol buffer [66] структурой, исключением является пакет ERR.
- Ответ - данные, которые должны возвращаться в качестве ответа.
- Описание - описание команды.

Таблица 3

Описание API протеза

Запрос	**Код запроса**	**Данные**	**Ответ**	**Описание**
Telemetry	3	-	Telemetry	Телеметрия протеза. Содержит актуальную информацию о системе. Отправка команды начинается при подписке на телеметрию через **StartTelemetry**.

GetSettings	4	-	GetSettings	Получение текущих настроек протеза. Настройки содержат конфигурацию модулей системы.
SetSettings	5	SetSettings	ACK/ERR	Выполнить конфигурацию модулей протеза.
GetGestures	6	-	GetGestures	Актуальные жесты, хранимые на протезе.
SaveGesture	7	SaveGesture	ACK/ERR	Выполнить сохранение/обновление жеста протеза. После успешного завершения в телеметрии будет отправляться новое время синхронизации.
DeleteGesture	8	DeleteGesture	ACK/ERR	Выполнить удаление жеста протеза. После успешного завершения в телеметрии будет

				отправляться новое время синхронизации.
PerformGestureId	9	PerformGestureId	ACK/ERR	Выполнить жест по id. В телеметрии, в поле Gesture, будет установлено id жеста при успешном начале исполнения по id.
PerformGestureRaw	10	PerformGestureRaw	ACK/ERR	Выполнить жест, который не сохранен в системе. В телеметрии, в поле Gesture, будет установлено id жеста, который был отправлен на исполнение.
SetPositions	11	SetPositions	ACK/ERR	Установить положения пальцев протеза.
UpdateLastTimeSync	12	UpdateLastTimeSync	ACK/ERR	Обновить общее время синхронизации жестов системы.

GetTelemetry	13	-	GetTelemetry	Получить единичную телеметрию системы без подписки.
StartTelemetry	14	StartTelemetry	ACK/ERR	Подписаться на получение телеметрии устройством. После отключения устройства телеметрия прекратит отправку. Повторная отправка этого запроса невозможна до остановки телеметрии через **StopTelemetry**. При попытке начала телеметрии без ее завершения будет возвращена ошибка.

StopTelemetry	15	-	ACK/ERR	Остановить отправку телеметрии.
GetMioPatterns	16	-	GetMioPatterns	Получить набор паттернов системы.
SetMioPatterns	17	SetMioPatterns	ACK/ERR	Обновить паттерны системы.

Фреймы с типом ACK и ERR являются частными случаями ответов и приведены в таблице 4. Они имеют собственный код ответа, который записывается в поле Тип пакета. ACK должен возвращаться при успешном выполнении запроса, на который не подразумевается ответ в виде какой-либо структуры данных. Тип пакета равняется 0x01. Пакетом с типом ERR возвращается в качестве ответа, если система не смогла корректно обработать запрос. Тип пакета равняется 0x02. ERR в отличии от ACK может быть отправлен в качестве ответа на любой тип пакета, поэтому ведущее устройство всегда должно быть готово к получению данного фрейма. ERR может содержать дополнительную информацию об ошибке в Payload в виде protobuf сообщения Error.

Таблица 4

Описание частных случаев ответа

Тип ответа	**Код ответа**	**Данные**	**Описание**
Empty	0	-	Некорректное состояние. Не должно присутствовать в системе, свидетельствует о

			серьезном нарушении работоспособности протокола.
ACK	1	-	Запрос принят и исполнен.
ERR	2	Error	Ошибка выполнения запроса. Payload может содержать дополнительную информацию об ошибке в виде protobuf сообщения Error.

2.3.2. **Интерфейс взаимодействия центрального контроллера с драйвером линейных приводов**

Одной из причин разделения контроллера бионического протеза на контроллер управления из Raspberry Pi Zero W и контроллер линейных приводов являлось требование большого количество контактов для отдельного мотора (5 контактов на каждый привод [63], на 6 приводов 30 контактов) и отдельное машинное время для обработки обратной связи. В связи с этим появились дополнительные трудности, свойственные многоконтроллерными системам, а именно организация обмена данными между контроллерами.

Контроллеру управления необходимо выполнять обмен конфигурацией (положения приводов) и состояниями (телеметрией) с контроллером линейных приводов протеза. Драйвер линейных приводов располагается на расстоянии до 10см и имеет пропускной способностью не более 100Hz для обмена телеметрией устройства. Для взаимодействия с драйвером могут

использоваться проводные интерфейсы, такие как UART, I2C, SPI, CAN и Ethernet [56]. Для реализуемой системы оптимальным интерфейсом является интерфейс SPI. Он является быстрым, надежным и зарекомендовавшим себя полнодуплексным синхронным интерфейсом [12] и не требует дополнительных сложных модулей и модуляторов в отличие от CAN и Ethernet. Так же выбору послужило то, что данный интерфейс был практически единственными интерфейсом, контакты которого были не задействованы ни на контроллере управления (Raspberry Zero W), ни на драйвере линейных приводов (STM32F103C8T6).

Для покрытия требуемой функциональности используется следующая концепция: в системе инициализатором отправки команды (spi master) является контроллер управления протеза, который передает протокольные команды на драйвер линейных приводов (spi slave). Между контроллерами имеются команды установки положения и получения телеметрии. Протокольные команды имеют фиксированный размер в 9 байт и содержат (рисунок 12):

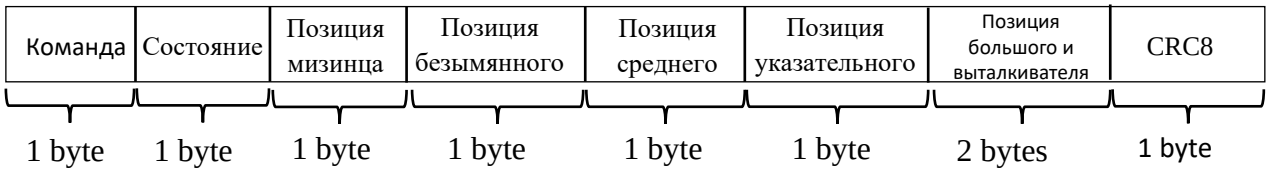

Рисунок 12 - Формат передачи состояния драйвера моторов

- Команду – протокольная команда, которая закодирована следующими значениями:
  - 0x0 – запрос/получение телеметрии устройства.
  - 0x1 – запрос установки положений протеза.
- Состояние протеза – текущий режим работы драйвера протеза, который закодирован следующими значениями:
  - 0x0 – режим инициализации. Драйвер находится в момент старта системы и до окончания калибровки положения всех пальцев;

    - 0x3 – режим ожидания. Драйвер ожидания запроса выполнения установки положений.
    - 0x4 – режим установки положений. Драйвер выполняет установку новых положений на каждый палец.
    - 0x5 – режим ошибки. Драйвер находится в ошибке. Требуется привлечение специалиста.
- Позицию мизинца – значение от 0 до 180 градусов, где 180 – согнутое состояние.
- Позицию безымянного – значение от 0 до 180 градусов, где 180 – согнутое состояние.
- Позицию среднего – значение от 0 до 180 градусов, где 180 – согнутое состояние.
- Позицию указательного – значение от 0 до 180 градусов, где 180 – согнутое состояние.
- Позицию большого и выталкивателя – первый байт определяет положение большого пальца, второй байт отвечает за позицию выдвижения большого пальца, так же представлены значениями от 0 до 180 градусов, где 180 – согнутое состояние.
- CRC8 – циклический избыточный код с полиномом 0x07.

Master устройство выполняет отправку выбранной команды по SPI размером 9 байт, ожидает 2мс (время обработки команды) и отправляет еще 9 байт пустого заполнителя (0xFF) для установки на линии CLK синхросигнала, чтобы slave устройство могло отправить ответ на команду. Диаграмма последовательности протокола представлена на рисунке 13.

Рисунок 13 – Диаграмма последовательности протокола между контроллером управления и драйвером линейных приводов

3. РАЗРАБОТКА АППАРАТНО-ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ БИОНИЧЕСКОГО ПРОТЕЗА РУКИ

3.1. Разработка программного обеспечения центрального контроллера системы

3.1.1. Архитектура аппаратно-программного обеспечения центрального контроллера

Разработка программного обеспечения для встраиваемых и киберфизических имеет значительные различия с классической разработкой ПО, так как встраиваемым системам свойственно устаивать в процессе технологического развития оборудования и средств разработки [46]. Так, передовые процессы производства интегральных схем подвергаются изменениям каждые 12–18 месяцев [43], однако жизненный цикл некоторых продуктов, таких как автомобильная промышленность, военная промышленность и медицина (к которой и относится разрабатываемый бионический протез), значительно длиннее, чем цикл устаревания и внедрения компонентов.

Данная проблема сказывается на программном обеспечении встраиваемых систем, так как, зачастую, оно сильно завязано на аппаратном обеспечении системы и при любой замене аппаратных компонентов, либо внесении дополнительных требований, может потребовать такое количество изменений, что стоимость доработки существующей системы окажется значительно выше, чем разработка ее с нуля.

В связи с тем, что бионический протез так же относится к системам с долгим жизненным циклом, то архитектура программного обеспечения протеза должен обеспечивать минимизацию стоимости внесения изменений при создании новой версии протеза, исправлении багов ПО, либо замене компонентов в процессе разработки.

В первую очередь стоит выделить аппаратные компоненты, взаимодействие с которыми осуществляется программным обеспечением протеза и определить возможные изменений ПО, которые могут

потребоваться при замене устройства. Для центрального контроллера такими устройствами являются:

1) Встроенный Bluetooth модуль. Обеспечивает соединение и обмен с внешними подсистемами протеза по RFCOMM соединению. Для данного компонентам присутствует риск перехода на другую модель модуля, либо на другие интерфейсы, которых с каждым годом становится все больше [23]. Из этого следует, что изменение в модуле подключения повлекут изменения в процессе передачи протеза, а также в работе с модулем на уровне оборудования. Решением риска изменения модуля является написание высокоуровневого драйвера для работы с модулем. Решением риска перехода на другой интерфейс является разделение бизнес-логики взаимодействия с внешними устройствами от работы с драйвером модуля.

2) Внешний АЦП ADS1115 и электрод 13E200 MyoBock. Компоненты обеспечивают получение ЭМГ данных с мышц человека. Смена каждого из компонентов может повлечь изменения принимаемого ЭМГ сигнала. Так, замена АЦП повлечет изменение способа взаимодействия с модулем и изменения минимальных и максимальных оцифрованных показаний принимаемого сигнала при различных параметрах АЦП [74]. Замена электрода с активного на пассивный может потребовать дополнительных шагов в предобработке сигнала для устранения шумов. Решением является создание общей абстракции работы с датчиком, которая будет предоставлять неизменный диапазон значений, частоты и качества сигнала.

3) Драйвер линейных приводов. Смена интерфейса взаимодействия с SPI на другой интерфейс может повлечь изменение способа взаимодействие с устройством. Протокол, описанный в пункте 2.3.2 уже предоставляет абстракцию для взаимодействия с положениями протеза, но для уменьшения риска внесения изменений, взаимодействие по SPI должно происходить в виде отдельного компонента, и бизнес-логика системы не должна знать о выбранном интерфейсе.

Для исключения тесной зависимости программного обеспечения бионического протеза от выделенных аппаратных компонентов следует разделить программное обеспечение на различные архитектурные уровни. Так, согласно Роберту Мартину, эффективным решением для встраиваемых систем является разделение программного обеспечения на следующие уровни [3]:

1) Уровень оборудования – непосредственное аппаратное обеспечение системы.

2) Уровень микропрограмм – уровень, отвечающий за непосредственную работу с аппаратным обеспечением. На данном уровне находятся драйвера низкоуровневых устройств.

3) Уровень HAL (Hardware Abstraction Layer; HAL) – слой аппаратных абстракций. Уровень, приспосабливающий микропрограммы под потребности программного обеспечения.

4) Уровень программного обеспечения – уровень, содержащий программное обеспечение предметной области (бизнес-логику) и высокоуровневые абстракции.

Наложив требования и аппаратные компоненты программного обеспечения центрального контроллера на архитектурные уровни была разработана архитектура аппаратно-программного обеспечения центрального контроллера протеза. Разработанная архитектура приведена на рисунке 14. Рассмотрим каждый уровень подробнее.

В качестве языка программирования для программного обеспечения протеза был использован Python версии 3.7, как оптимальный язык для работы с Raspberry Pi имеющий большое сообщество и поддержку производителем.

Разработанное программное обеспечение находится в Приложении 4.

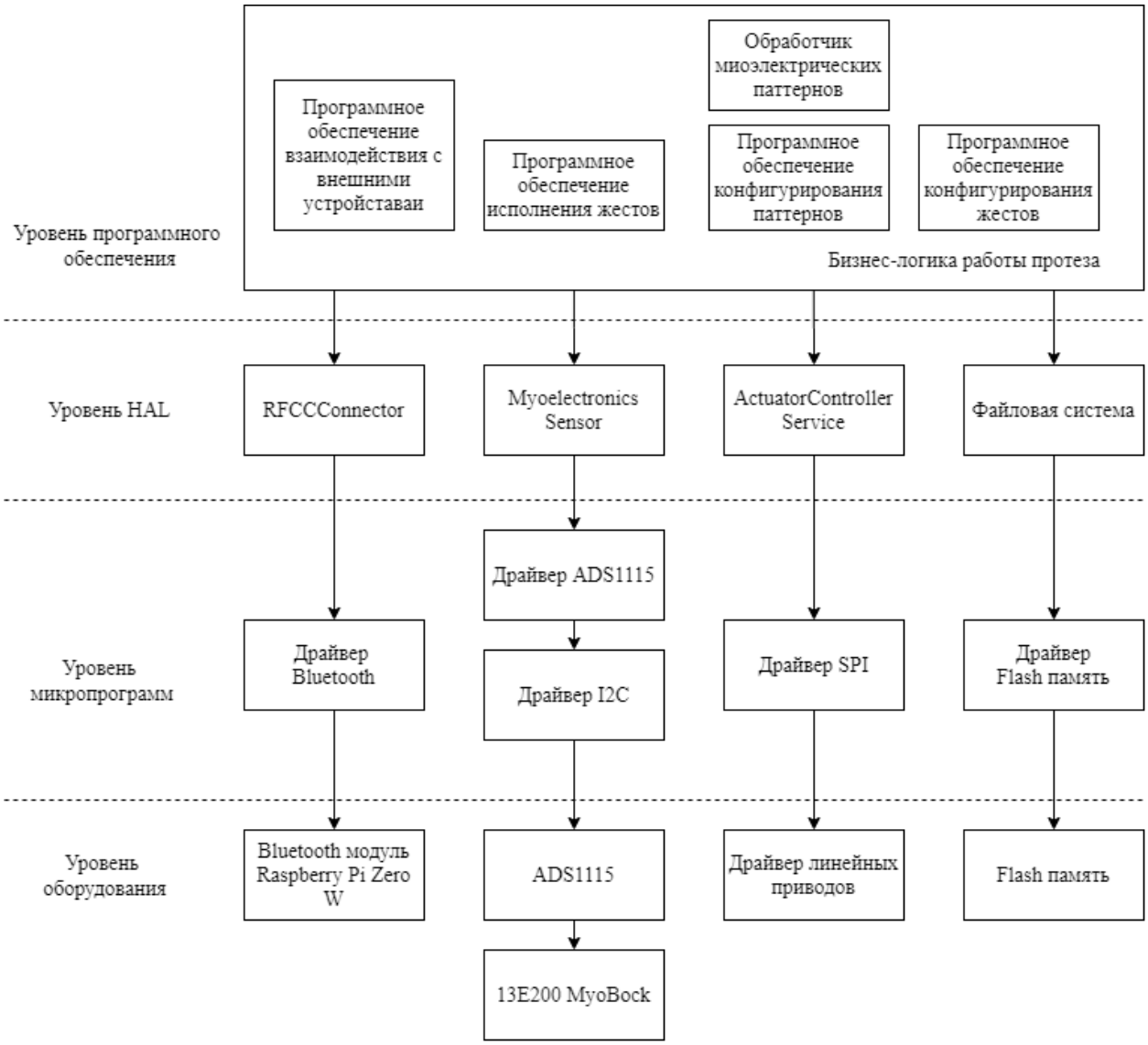

Рисунок 14 – Архитектура уровней аппаратно-программного обеспечения центрального контроллера бионического протеза

3.1.2. **Уровень оборудования**

Уровень оборудования представлен аппаратными компонентами системы, описанными в пункте 2.2, к которым относятся:

1) Встроенный Bluetooth модуль Raspberry Pi Zero W
2) АЦП ADS1115
3) Драйвер линейных приводов
4) Flash память

Благодаря тому, что в качестве основы центрального контроллера был выбран микрокомпьютер Raspberry Pi Zero W, обеспечивать взаимодействие с такими компонентами как Bluetooth модуль и Flash память напрямую не

требуется. За каждый из этим компонентов отвечает отдельный драйвер в составе операционной системы Raspberry Pi OS [67] и внутреннее устройство и конфигурирование компонента не важно для написания программного обеспечения.

Идентичная ситуация обстоит и с драйвером линейных приводов. Между центральным контроллером и драйвером приводов предполагается связь по SPI интерфейсу и протоколу, описанному в пункте 2.3.2, который не зависит от конкретной организации драйвера и используемых приводов, а обеспечивает установку угловых положений пальцев.

Но совершенно другим случаем является внешнее АЦП ADS1115, которое предназначено для считывания ЭМГ показаний. Устройство поддерживает различный набор конфигураций и отдаваемых параметров, которые должны настраиваться на уровне микропрограммы и уровне HAL. Для определения возможных конфигураций и параметров следует рассматривать Datasheet устройства ADS1115 [50]. Так, наиболее важными параметрами ADS1115, которые требуют настройку в регистре конфигурации (рисунок 15), являются настройки усилителя, частоты дискретизации и режима работы.

15	14	13	12	11	10	9	8
OS	MUX[2:0]			PGA[2:0]			MODE
R/W-1h	R/W-0h			R/W-2h			R/W-1h
7	6	5	4	3	2	1	0
DR[2:0]			COMP_MODE	COMP_POL	COMP_LAT	COMP_QUE[1:0]	
R/W-4h			R/W-0h	R/W-0h	R/W-0h	R/W-3h	

Рисунок 15 – регистр конфигурации ADS1115 [50]

ADS1115 имеет программный усилитель, обеспечивающий диапазон входных напряжений преобразования. Настройка усилитель осуществляется в битах 11:9 (поле PGA[2:0]) и поддерживает диапазоны напряжений, приведенные в таблице 5.

Таблица 5

Конфигурация диапазона входных напряжений ADS1115

Конфигурация поля PGA[2:0]	Полный диапазон входных напряжений	Единиц напряжения на один разряд АЦП
000	±6.144В	187.5мВ
001	±4.096В	125 мВ
010 (по умолчанию)	±2.048В	62.5мВ
011	±1.024В	31.25мВ
100	±0.512В	15.625мВ
101	±0.256В	7.8125мВ
110	±0.256В	7.8125мВ
111	±0.256В	7.8125мВ

Диапазонами напряжения электрода 13E200 MyoBock являются 0-5В и единственной допустимой конфигураций поля PGA[2:0] являются биты 000.

Электрод 13E200 MyoBock имеет ширину полосы частот 90-450Гц, поэтому для поля DR[2:0], которое отвечает за частоту дискретизации, значение по умолчанию в 128Гц (конфигурация бит 100) не удовлетворяет допустимым критериям. Для достижения требуемой частоты дискретизации следует установить в биты 7:5 значение 110, равное частоте дискретизации 475Гц.

Заключительным параметром настройки является режим работы. ADS1115 предоставляет два режима работы:

1) режим непрерывного преобразования – АЦП выполняет преобразование непрерывно. Когда преобразование завершено, ADS1115 помещает результат в регистр преобразования и немедленно начинает новое преобразование.

2) однократное считывание (значение по умолчанию) – АЦП выполняет единичное преобразование при установке бита 1 в поле OS регистр конфигурации.

В связи с тем, что обработки миоэлектрических показаний должна проводиться с максимальной частотой и на постоянной основе, то следует использовать режим непрерывного преобразования путем установки бита 0 в поле MODE регистра конфигурации.

После завершения конфигурации значения по каналу преобразования могут быть получены из регистра преобразования по I2C интерфейсу в непрерывном режиме.

3.1.3. **Уровень микропрограмм**

Для взаимодействия с Bluetooth модулем и Flash картой используется предоставляемое программное обеспечение в составе операционной системы Raspberry Pi Zero W и пакетов языка Python.

Так, для конфигурирования Bluetooth модуля был использован CLI инструмент bluetoothctl. Устройство было включено и ему было задано имя для обнаружения и осуществления подключения с внешних устройств. Для отправки и приема данных по RFCOMM протоколу Bluetooth из Python приложения протеза был взят пакет для Raspberry Pi - Blue Dot [52], который позволяет открывать RFCOMM соединение, записывать в него поток байт и выполнять прием в обратных вызовах.

Для работы с Flash картой дополнительных поисков не потребовалось. Raspberry Pi позволяет работать с Flash картой как с обычном файловой системой (при ее наличии на карте). Исходя из этого, приложение протеза может получить полный доступ к файловой системе карты из Python пакета os, который предоставляет доступ к записи, чтению директорий и файлов, а также другим возможностям работы с файловой системой.

Для работы с ADS1115 по I2C интерфейсу и для работы с SPI интерфейсом также были использованы сторонние пакеты Python для Raspberry Pi, обеспечивающие необходимый уровень конфигурации устройств и интерфейсов. Так, для взаимодействия с ADS1115 был использован пакет Adafruit_ADS1x15, обеспечивающий конфигурирование диапазона напряжении, частоты дискретизации и режима работы. Код

конфигурирования и начала непрерывного преобразования приведен на рисунке 16.

```
def __init__(self):
    # Create an ADS1115 ADC (16-bit) instance.
    self._adc = Adafruit_ADS1x15.ADS1115()
    self._adc.stop_adc()

    self._CHANNEL = 0
    self._GAIN = self._ADS1x15_CONFIG_GAIN[0]
    self._RATE = 475

    # For 5 volts - maximum value is:
    if self._ADS1x15_CONFIG_GAIN[0] == self._GAIN:
        self._maximum_value_for_16_bit_and_5v = int(5000000 / 187.5)
    else:
        self._logger.info("Conversation for sensor not supported.")
        raise Exception('Conversation not supported')

    self._coefficient_conversation = self._MAXIMUM_VALUE_FOR_12BIT / self._maximum_value_for_16_bit_and_5v

def start_sensor(self):
    self._logger.info("Sensor try to start.")
    self._adc.start_adc(self._CHANNEL, self._GAIN, self._RATE)
    self._logger.info("Sensor running.")
```

Рисунок 16 – Код работы с пакетом Adafruit_ADS1x15

Для обеспечения передачи данных по SPI интерфейсу был использован пакет spidev, который является драйвером работы с SPI интерфейсом для Raspberry Pi. Пакет позволяет задавать настройки внутреннего SPI модуля Raspberry Pi и обеспечивает передачу массива байт.

3.1.4. **Уровень HAL**

Для минимизации зависимостей между уровнем программного обеспечения и уровнем микропрограмм были написаны дополнительные сервисы для каждого устройства, которые позволяют отделить логику работы с конкретными устройствами от бизнес-логики системы.

Сервисом для работы с Bluetooth модулем является RFCCConnector, который имплементирует два интерфейса: IResponseWriter и IPackageReceiver. Имплементация от интерфейсов позволяет использовать Bluetooth соединение через инверсию зависимостей. Когда уровню программного обеспечения потребуется осуществить прием, либо передачу данных на внешние подсистемы, то ПО уровня программного обеспечения не будет информировано о том, что он работает с Bluetooth протоколом RFCOMM и передает массивы байт, то есть не будет зависеть от интерфейса по причине

того, что на уровне HAL реализуется контракт передачи и приема пакетов системы из пункта 2.3.1. Уровню программного обеспечения только потребуется сформировать полезную нагрузку пакета и установить тип пакета.

Для реализации исполнения протокольных команд при запуске приложения вызывается метод run у класса RFCCConnector, при вызове метода выполняется создание Bluetooth-сервера, который разрешает соединения и принимает входящие RFCOMM последовательные данные. Когда данные получены сервером, они передаются функции обратного вызова _data_received_handler. Определить наличие подключенного внешнего устройства можно в свойстве connected.

При приеме данных в функцию обратного вызова происходит обработка входящих данных парсером протокола, представленным классом ProtocolParser. Метод update реализует прослушивание входного потока байт с RfccConnector на наличие SDF, типа пакета и данных, а также проверку целостности пакета. В методе предусмотрена обработка ошибок таймаута, благодаря чему протез сможет принимать следующий пакет, даже если предыдущий не был полностью доставлен из-за ошибок, связанных с неожиданным разрывом соединения. Парсер осуществляет свою работу на основании машины состояний, где каждый пришедшее поле фрейма ознаменует переход в следующее состояние. По завершению сборки входного пакета происходит передача пакета в метод receive_package у RFCCConnector, который объявлен в интерфейсе IPackageReceiver.

ActuatorControllerService используется для взаимодействия с контроллером приводов протеза. На уровне микропрограмм взаимодействие с драйвером осуществляется путем отправки потока байт по шине SPI. ActuatorControllerService предоставляет работу с контроллером приводов как с конечным высокоуровневым устройством, поддерживающим установку позиций в угловом соотношении и получение телеметрии путем вызова соответствующих методов. Для установки позиций в сервисе имеется метод set_new_positions, который выполняет формирование и отправку команды

установки позиция, согласно контракту из пункта 2.3.2. Телеметрия драйвера линейных приводов доступна в непрерывном режиме при вызове метода enable_telemetry.

Абстракцией над работой с электродом и АЦП с помощью пакета Adafruit_ADS1x15, является класс MyoelectronicsSensor. Так как изменение конфигурации АЦП не требуется, то класс имеет только методы начала (start_sensor) и завершения (stop_sensor) получения значений с электрода в непрерывном режиме через метод get_value. Также класс обеспечивает конвертацию неполного диапазона 16-битных значений с АЦП (максимальным значением для 5В является 26666) в диапазон 12-битных значений от 0 до 4095, при этом напряжение 5В будет равно 4095.

3.1.5. **Уровень программного обеспечения**

Уровень программного обеспечения содержит код, отвечающий за поведение системы в рамках предметной области (бизнес-логику). Для разрабатываемого бионического протеза код логики предметной области должен обеспечивать:

1) Исполнение протокольный команд
2) Распознавание миоэлектрических паттернов
3) Хранение конфигурации и настроек паттернов
4) Хранение пользовательских жестов
5) Исполнение жестов протеза по протокольным командам и распознанным миоэлектрическим паттернам

Каждое поведение задействует один или несколько компонентов уровня HAL. Так, для исполнение протокольных команд требуется использование RFCCConnector, для распознавания паттернов – MyoelectronicsSensor, для хранения настроек, жестов и паттернов – файловая система, а для непосредственного исполнения жестов задействуются сразу все компоненты уровня HAL.

В первую очередь были введены модели данных, которые существуют в системе и являются сущностями для работы с протезом на уровне бизнес-логики:

1) Gesture - шаблон жеста.
2) GestureAction - действие в рамках жеста.
3) MotorPositions - позиции моторов в угловом соотношении.
4) Positions - позиции пальцев протеза.
5) MioPattern – конфигурация паттерна протеза.
6) Settings – конфигурация протеза.

Также для внешних взаимодействий по протоколу взаимодействия из пункта 2.3.1 и работы с интерфейсами IResponseWriter и IPackageReceiver были введены следующие классы:

1) Request;
2) Response;
3) CommandType;
4) DTO объекты для всех типов команд.

Классы Request, Response содержат тип команды и полезную нагрузку.

В проекте используются DTO (Data Transfer Objects) — это объекты, передающие данные. Они не имеют поведения, в них есть только поля, конструктор и геттеры/сеттеры и методы сериализации и десериализации в структуры данных Protocol Buffer. Такие объекты создаются для каждой команды, которая есть в API протеза.

Каждая функция протеза должна выполняться параллельно, поэтому приложение центрального контроллера протеза построено по модели вычислений Кана [73] и представляет собой 6 поток приложения, обеспечивающих передачу данных с помощью очередей (Python Queue). Для обеспечения работоспособности бионического протеза введены потоки со следующим назначением (рисунок 17):

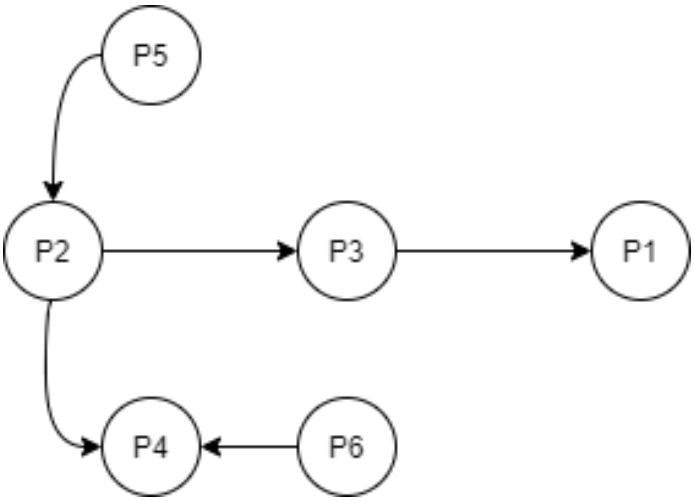

Рисунок 17 –DFD модель центрального контроллера протеза

1) Поток для класса ActuatorControllerService (P1) – обеспечивает отправку и прием положений протеза на контроллер приводов бионического протеза по SPI.

2) Поток для класса Communication(P2) – обеспечивает управление протезом по Bluetooth командам и миоэлектрическим паттернам.

3) Поток для класса HandController (P3) – обеспечивает исполнение жеста протеза.

4) Поток для отправки телеметрии (P4) – обеспечивает отправку телеметрии минимум каждую секунду (настраиваемый параметр) при наличии подключения.

5) Поток для MyoelectronicsService (P5) – обеспечивает получение и обработку ЭМГ сигнала.

6) P6 – поток отправки команды запроса телеметрии драйвера линейных приводов в очередь потока P1.

За обработку пакетов из очереди отвечает класс Communication. Непосредственно через него предоставляется доступ к репозиторию жестов, настройкам протеза, телеметрии и к классу исполнения жестов протеза. В бесконечном цикле отдельного потока опрашивается очередь запросов в блокирующем режиме. При поступлении нового запроса поток просыпается и начинается его обработка. Проверяется тип команды и в соответствии с ним выполняется действие и отправляется ответ об успешном выполнении, либо

ответ, содержащий данные. Пример для протокольной команды установки позиции пальцев изображен на рисунке 18.

```
if request.command_type == CommandType.SetPositions:
    self.handle_set_positions_request(request.payload)
    request.response_writer.write_response(Response(CommandType.Ok, None))
    return
```

Рисунок 18 – Пример обработки протокольной команды установки позиции пальцев в классе Communication

Функция handle_set_positions_request(request.payload) десериализует payload запроса через соответствующий SetPositionsDto объект, после чего отправляет команду установки положений в протез.

Если команда требует ответа, а не подтверждения получения, то в качестве ответа посылается сериализованный объект. Это делается путем вызова функции serialize() у DTO объекта. Пример для команды получения жеста с протеза изображен на рисунке 19.

```
if request.command_type == CommandType.GetSettings:
    settings_dto = self.handle_get_settings()
    request.response_writer.write_response(Response(CommandType.GetSettings,
                                                    settings_dto.serialize()))
    return
```

Рисунок 19 – Пример обработки команды с сериализуемым ответом в классе Communication

Непосредственное сохранение новых жестов обеспечивается протокольной командой SaveGesture и не отличается от приема и исполнения остальных команд за исключением вызова метода add_gesture у репозитория жестов в обработчике команды, приведенном на рисунке 20.

```
def handle_save_gesture(self, payload: bytes):
    logging.info(f'Start handling save gesture')
    save_gesture_dto = SaveGestureDto()
    save_gesture_dto.deserialize(payload)

    self._gesture_repository.add_gesture(save_gesture_dto.time_sync, save_gesture_dto.gesture_dto)
```

Рисунок 20 – Пример команды сохранения жеста

Жесты хранятся в классе GestureRepository, имплементирующем паттерн репозиторий. Он выполняет следующие функции:

- add_gesture - добавление жеста в память;
- delete_gesture - удаление жеста из памяти;
- get_gesture_by_id - поиск жеста по id (например, как часть команды исполнения);
- get_all_gestures - возвращает список всех сохраненных жестов.

Как было отмечено, на Raspberry Pi стоит unix-like ОС, поэтому все реализовано через обращения к файловой системе через стандартные средства питона по работе с файловой система из пакета os. Рассмотрим реализацию метода add_gesture в качестве примера (рисунок 21):

```
def add_gesture(self, current_time: int, new_gesture: GestureDto):
    if new_gesture.id in self._gestures_dictionary:
        self._logger.info(f'Update gesture {new_gesture.id}')
    else:
        self._logger.info(f'Adding new gesture {new_gesture.id}')

    with open(os.path.join(self._path_to_gestures, new_gesture.id + self._gestures_file_extension),
              'wb') as gesture_file:
        gesture_file.write(new_gesture.serialize())

    self._gestures_dictionary[new_gesture.id] = new_gesture
    self.update_time_sync(current_time)
```

Рисунок 21 – Код метода add_gesture

1) Выполняется открытие файла “{id}.gesture” в директории жестов _path_to_gesture с режимом на запись двоичного файла.

2) GestureDto сериализуется в бинарное представление структуры protocol buffer.

3) Сериализованный байтовый массив записывается в открытый файл.

4) Файл закрывается.

5) Кеш жестов, хранимых в оперативной памяти, обновляется.

6) Выполняется обновление последнего времени синхронизации жестов для последующей синхронизации.

В связи с тем, что контроллер обеспечивает кеширование, то введено ограничение на максимальное количество жестов – 5000. Так, при размере одного жеста в 5Кб будет задействовано 25Мб оперативной памяти, что допустимо для Raspberry Pi Zero W с объемом в 512Мб.

После выполнения описанной последовательности жест считается сохраненным в системе и может быть использован для исполнения на протезе, а в ответе на запрос вернется ACK. При возникновении ошибок во время сохранения выброшенное исключение будет обработано в классе Communication и в ответе на запрос вернется ERR.

Исполнение жестов также происходит в отдельном потоке в классе HandController и алгоритм заключается в следующем:

1) Стоим в блокировке потока на чтении очереди жестов на исполнение.

2) В очередь поступает жест путем вызова метода perform_gesture у HandController из класса Communication (обработчик протокольных команд), либо MyoelectronicsService (обработчик распознавания миоэлектрических паттернов).

3) Жест появился - начинаем его исполнение.

4) Узнаем какие действия есть в жесте и повторяющийся ли жест.

5) Пока в очереди нет новых жестов, исполняем текущий жест.

6) Отправляем позиции на драйвер моторов, через ActuatorControllerService.

7) Если есть ещё действия в жесте, то переходим к пункту 4.

8) Если жест повторяющийся, то сбрасываем счетчик действий и переходим к 4 пункту.

9) Жест завершен, переходим на пункт 1.

3.2. **Разработка алгоритмов распознавания миоэлектрических паттернов**

3.2.1. **Анализ и выбор алгоритмов распознавания миоэлектрических паттернов**

Распознавание миоэлектрических паттернов можно представить как набор следующих этапов [29, 33, 22] (рисунок 22):

1) Прием сигнала
2) Предобработка
3) Извлечение признаков
4) Классификация

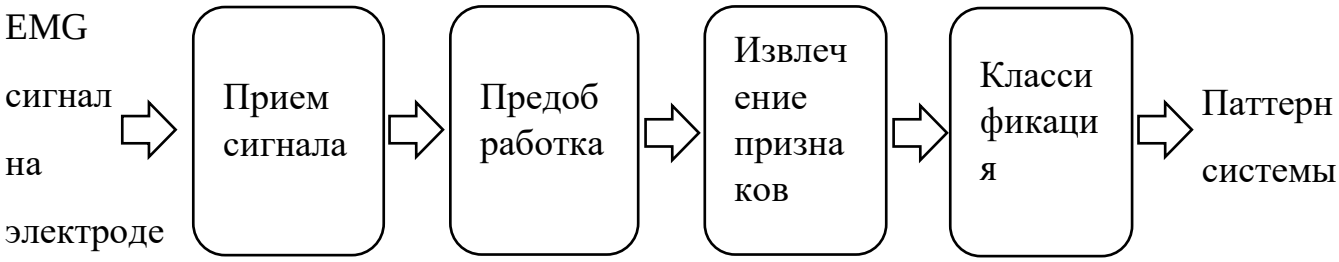

Рисунок 22 – Блок схема метода распознавания жестов

Прием сигнала

Данный этап сильно зависит от электрода, прием которого обеспечивается, а также от количества электродов в системе. Электроды регистрирует мышечную активность руки и подают аналоговое значение данной активности на вывод. С вывода значение может быть считано с помощью аналогов цифрового преобразователя. От количества электродов зависит объем жестов, которые могут быть распознаны системой, так как с помощью большего количества электродов можно охватить несколько мышц руки. Качество электрода определяет качество сигнала, которое находится на выводе. На пассивных электродах будет присутствовать большое количество помех, которые потребуют значительную предобработку.

Предобработка

На этапе предобработки сигнала происходит фильтрация сигнала с целью исключить помехи электрода и помехи линий питания. На этот этап значительно сказывается качество выбранного электрода. На рынке представлены электроды разной степени доступности, но наиболее важная классификация электродов, которая влияет на качество сигнала – активные и пассивные электроды [14]. Различие заключается в объеме необходимой предобработки. Пассивные электроды предоставляют нефильтрованный сигнал с шумами (рисунок 23), который требует выполнение фильтрации.

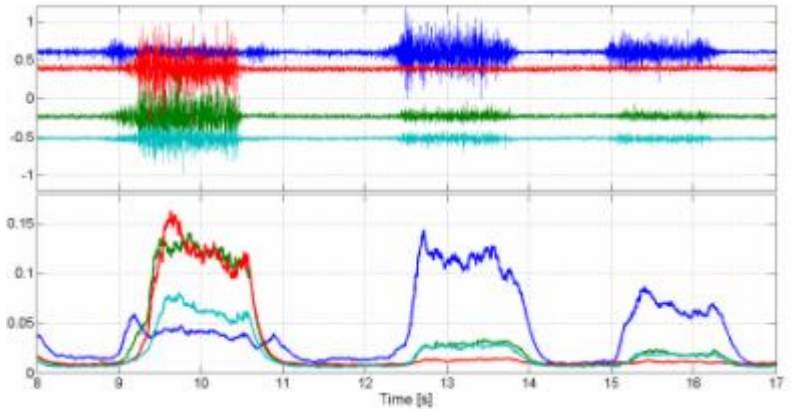

Рисунок 23 – Исходный сигнал пассивного электрода сверху и снизу сигнал с применением режекторного фильтра на 50Гц и ФНЧ [4]

В это время, активные электроды, к которому и относится 13E200 MyoBock, значительно выигрывают, так как не требуют отдельной фильтрации, она уже осуществляется в самом электроде аппаратно. Для подтверждения данной факта были сняты показания с электрода 13E200 MyoBock, показания отображены на рисунке 24.

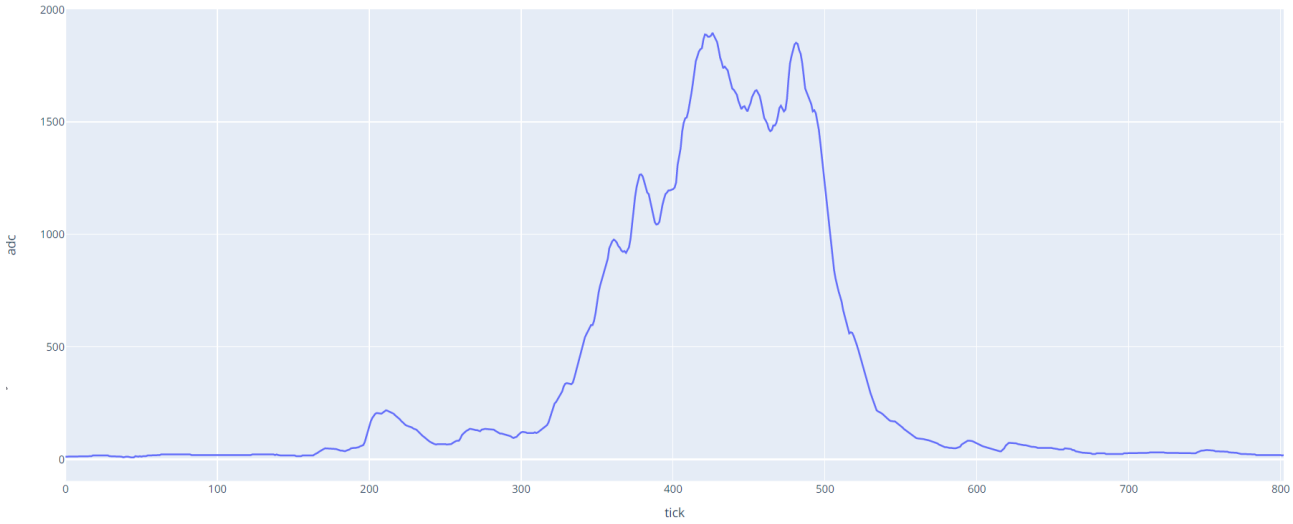

Рисунок 24 – Полученный сигнал на выходе электрода 13E200 MyoBock при сжатии руки

Как можно видеть, сигнал не требует дополнительной фильтрации на стороне приема, что уменьшает кол-во операций, которые требуется выполнять Raspberry Pi Zero W.

Помимо фильтрации на данном этапе так же происходит выделение сигнала для последующего извлечения признаков. Для определения алгоритма выделения жестов были собраны данные по двум жестам в количестве 30 штук, которые должны распознаваться системой:

1) Сжатие руки – сжатие пальцев кисти в среднем темпе. Показания жеста приведены на рисунке 24.

2) Резкое сгибание руки – резкое сгибание кисти руки. Показания приведены на рисунке 25.

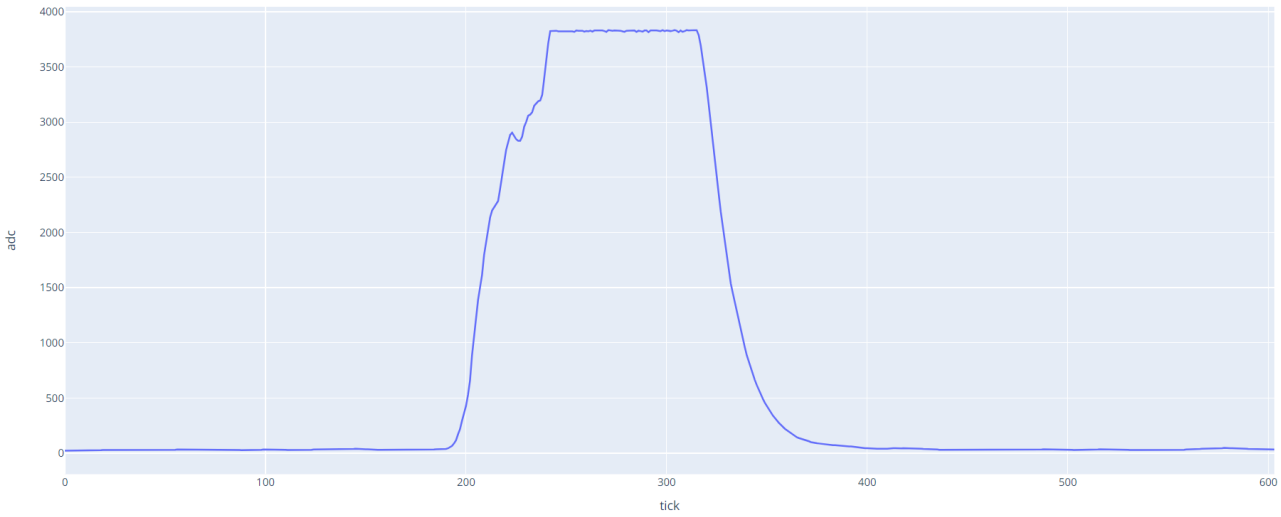

Рисунок 25 – Полученный сигнал на выходе электрода 13E200 MyoBock при резком сгибании руки

Данное количество жестов было выбрано после исследовании статьи Single channel surface EMG control of advanced prosthetic hands: A simple, low cost and efficient approach [36]. В статье рассматривалось распознавание трех жестов на основании одного канала ЭМГ. Авторы статьи получили положительные результаты распознавания трех жестов, но отметили, что тренировка мышц даже для трех жестов требует значительных усилий. В связи с этим, для разрабатываемой системы были выбраны только два разнородных жеста, которые сможет повторить практически любой человек без значительных тренировок мышц руки и которые будет довольно легко отличить классификатору друг от друга.

Для извлечения сигнала может использоваться запись на протяжении константного времени при достижении порогового уровня в принимаемом сигнале [22], но более оптимальным решением для систем с невысокой производительностью является выделение сигнала жеста из общего потока с помощью расчета среднего по окну [36], что уменьшит влияние помех при единоразовом достижении порога. Для выделения сигнала жеста из общего потока ЭМГ используется среднее на окно из пяти элементов. Если среднее окна будет больше значения 400 (12-битное АЦП с максимальным значением 4095), то система будет считать, что принимает сигнал паттерна. Так, сигнал, изображенный на рисунке 25, после обработки примет вид из рисунка 26.

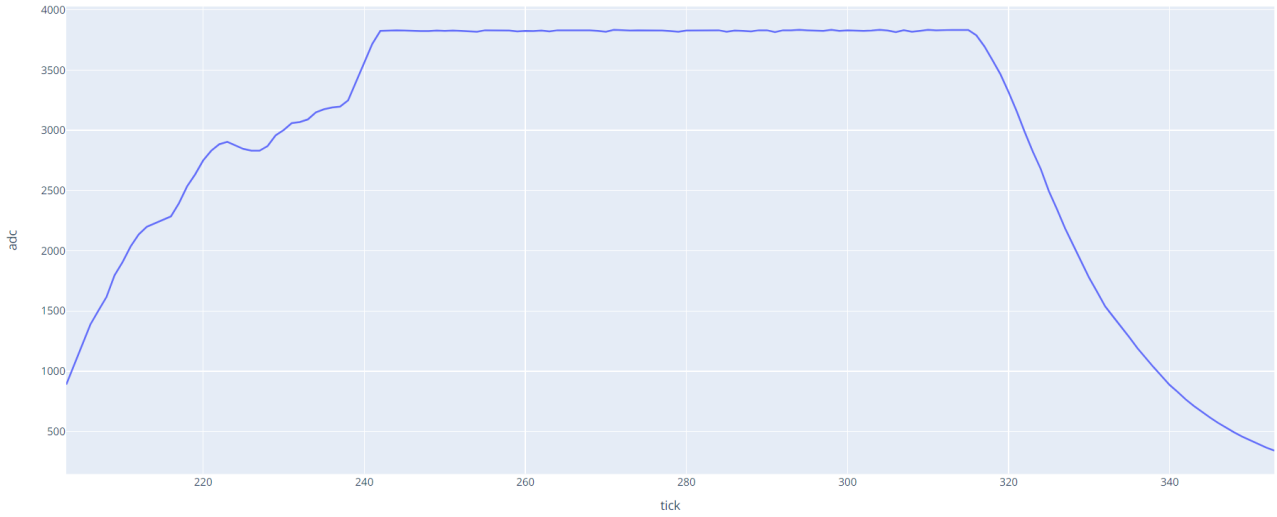

Рисунок 26 – Извлеченный сигнал жеста с рисунка 25

Извлечение признаков и классификация

Затем сигнал ЭМГ разбивается на окна (семплы) и рассчитывается функция по каждому окну, которая представляет собой характеристику (признаки) окна. Характеристики сигнала можно сгруппировать в три категории:

1) Временная область (TD).
2) Частотная область (FD).
3) Частотно-временная область (TFD).

Каждая категория описывает разные компоненты сигнала и требует различных подходов. Для определения наиболее подходящего алгоритма выделения признаков была исследована статья Evaluation of surface EMG-based recognition algorithms for decoding hand movements [15], в которой выполнено сравнение всех категорий выделения признаков в сочетании с классификаторами по эффективности и времени обработки. Стоит оговорится, что FD и TFD алгоритмы были бы неприменимы к разрабатываемой системе, так как частота дискретизации довольно низкая и невозможно обеспечить постоянную частоту, так как Raspberry Pi Zero W работает на операционной системе Unix-like и драйверы устройств не смогут поддерживать постоянную и высокую частоты дискретизации. В связи с этим основной упор был сделан на исследование TD признаков. Выбор алгоритма извлечения признаков был

основан на результатах исследования рассматриваемой статьи и рекомендациях автора статьи. Результаты для TD признаков приведены на рисунке 27.

TDF		Average classification accuracy (%) ± STD					
		LDA	KNN	DT	MLE	SVM	MLP
1	MAV	$84.65^{c} \pm 7.73$	$93.17^{ab} \pm 4.60$	$90.05^{abc} \pm 4.92$	$89.93^{bcd} \pm 6.21$	$84.02^{bc} \pm 8.47$	$91.14^{bcdef} \pm 4.69$
2	STD	$84.66^{c} \pm 7.95$	$93.52^{a} \pm 3.56$	$89.75^{abcd} \pm 5.01$	$89.45^{bcd} \pm 6.05$	$84.62^{bc} \pm 7.62$	$91.00^{cdef} \pm 4.20$
3	Var	$71.21^{f} \pm 11.35$	$90.42^{abc} \pm 4.94$	$89.95^{abc} \pm 4.68$	$79.19^{gh} \pm 8.40$	$72.83^{e} \pm 10.14$	$90.51^{cdefg} \pm 4.11$
4	WL	$84.71^{c} \pm 7.43$	$93.16^{ab} \pm 4.38$	$90.18^{abc} \pm 5.47$	$90.44^{bc} \pm 5.88$	$84.86^{bc} \pm 8.08$	$91.21^{bcde} \pm 4.69$
5	ZC	$55.48^{mn} \pm 8.45$	$47.26^{lm} \pm 9.63$	$47.69^{pq} \pm 8.69$	$53.70^{nop} \pm 8.10$	$49.50^{mn} \pm 8.14$	$52.12^{rs} \pm 8.55$
6	NP	$57.92^{klmn} \pm 8.53$	$51.43^{jkl} \pm 10.42$	$51.34^{op} \pm 8.80$	$55.44^{mno} \pm 8.29$	$54.55^{ijkl} \pm 9.20$	$55.81^{pqr} \pm 8.90$
7	MPV	$83.49^{cd} \pm 7.81$	$92.29^{abc} \pm 4.36$	$88.83^{abcd} \pm 5.01$	$87.81^{bcde} \pm 6.64$	$84.14^{bc} \pm 7.11$	$89.85^{defg} \pm 4.69$
8	MFV	$19.97^{r} \pm 1.99$	$19.65^{q} \pm 1.69$	$18.43^{v} \pm 1.68$	$20.41^{tu} \pm 1.64$	$19.71^{s} \pm 1.94$	$19.93^{w} \pm 1.79$
9	SSC	$59.19^{jklm} \pm 9.08$	$50.01^{klm} \pm 9.61$	$51.39^{op} \pm 9.56$	$58.08^{lmno} \pm 8.97$	$56.61^{hij} \pm 8.79$	$55.42^{qr} \pm 9.28$
10	Cor	$62.89^{hij} \pm 9.07$	$57.46^{ghi} \pm 10.05$	$52.98^{no} \pm 9.56$	$61.43^{kl} \pm 9.35$	$58.18^{ghi} \pm 9.43$	$54.98^{rs} \pm 10.51$
11	DAMV	$84.85^{c} \pm 7.45$	$93.29^{ab} \pm 4.50$	$90.29^{abc} \pm 5.40$	$90.42^{bc} \pm 5.84$	$84.01^{bc} \pm 8.11$	$91.18^{bcdef} \pm 4.72$
12	FDim	$37.41^{pq} \pm 8.30$	$34.31^{n} \pm 8.21$	$34.14^{rst} \pm 8.99$	$34.73^{r} \pm 8.45$	$33.73^{pq} \pm 7.11$	$37.64^{uv} \pm 8.99$
13	MFL	$84.88^{c} \pm 7.47$	$93.95^{a} \pm 4.41$	$90.16^{abc} \pm 5.50$	$90.4^{bc} \pm 5.98$	$84.24^{bc} \pm 7.73$	$91.46^{abcde} \pm 4.41$
14	HFD	$41.07^{op} \pm 7.29$	$33.42^{no} \pm 6.84$	$33.33^{rst} \pm 6.75$	$40.17^{q} \pm 7.20$	$38.36^{op} \pm 6.27$	$38.15^{uv} \pm 7.24$
15	Skew	$35.84^{q} \pm 6.88$	$28.90^{op} \pm 5.93$	$29.77^{tu} \pm 6.52$	$33.78^{r} \pm 6.40$	$27.40^{r} \pm 5.51$	$34.08^{v} \pm 6.46$
16	IAV	$84.57^{c} \pm 7.75$	$93.23^{ab} \pm 4.49$	$90.01^{abc} \pm 5.13$	$89.97^{bc} \pm 6.04$	$83.97^{bc} \pm 8.08$	$91.62^{abcde} \pm 4.46$
17	HMob	$53.78^{n} \pm 10.05$	$45.60^{m} \pm 11.24$	$46.12^{q} \pm 10.71$	$53.27^{op} \pm 10.56$	$48.73^{mn} \pm 10.46$	$50.61^{s} \pm 9.52$
18	HCom	$68.72^{fg} \pm 8.71$	$66.18^{f} \pm 11.38$	$63.02^{jk} \pm 10.03$	$67.45^{ij} \pm 8.74$	$59.74^{fgh} \pm 11.63$	$66.63^{l} \pm 9.18$
19	ER	$57.85^{lmn} \pm 9.55$	$76.47^{e} \pm 5.34$	$80.84^{gh} \pm 6.10$	$69.34^{i} \pm 9.73$	$54.87^{hijkl} \pm 8.46$	$83.14^{ij} \pm 5.16$
20	DASDV	$83.95^{cd} \pm 7.95$	$92.26^{abc} \pm 4.50$	$88.94^{abcd} \pm 5.31$	$88.25^{bcde} \pm 5.98$	$85.01^{bc} \pm 7.44$	$89.75^{defg} \pm 4.29$
21	WAM	$71.11^{f} \pm 14.75$	$75.24^{e} \pm 14.43$	$78.82^{h} \pm 11.28$	$60.07^{klm} \pm 28.44$	$77.40^{de} \pm 13.88$	$79.57^{jk} \pm 11.11$
22	MAVS	$11.52^{s} \pm 2.89$	$32.82^{no} \pm 5.67$	$32.21^{st} \pm 5.56$	$26.23^{s} \pm 3.68$	$12.45^{t} \pm 1.40$	$18.52^{w} \pm 5.10$
23	Kurt	$35.80^{q} \pm 6.72$	$35.84^{n} \pm 8.16$	$35.08^{rs} \pm 8.09$	$32.76^{r} \pm 6.58$	$29.97^{qr} \pm 6.53$	$39.09^{u} \pm 7.97$
24	Perc	$81.36^{cde} \pm 8.16$	$89.60^{abc} \pm 6.15$	$86.32^{bcde} \pm 6.68$	$85.81^{cdef} \pm 7.43$	$80.15^{cd} \pm 8.82$	$87.73^{efgh} \pm 5.38$
25	Hist	$42.04^{op} \pm 6.97$	$46.42^{m} \pm 5.81$	$34.35^{rst} \pm 6.87$	$40.57^{q} \pm 6.72$	$41.91^{o} \pm 6.39$	$40.25^{tu} \pm 7.67$

[a,b] Means within a column not sharing a common superscript are significantly different ($p < 0.0001$)

Рисунок 27 – Результаты классификации жестов по TD признакам [15]

Автор отметил, что из 25 функций TD, девять функций - MAV, STD, WL, MPV, DAMV, MFL, IAV, DASDV и Perc - показали лучшую производительность для шести классификаторов. Классификаторы KNN и MLP показали численно более высокие показатели точности. Комбинируя классификатор KNN с каждым из вышеупомянутых функций, обеспечили точность с в среднем 93,17%, 93,52%, 93,16%, 92,29%, 93,29%, 93,95%, 93,23%, 92,26% и 89,60% соответственно. В Комбинация MAVS / LDA получила самую низкую скорость со средней точностью 11,52%.

Исходя из результатов автора Evaluation of surface EMG-based recognition algorithms for decoding hand movements [15], был выбран алгоритм извлечения признака MAV (среднее значение по модулю):

$$MAV = \frac{1}{N}\sum_{i=1}^{N}|x_i|,$$

где x_i-данные ЭМГ, N – количество отсчетов в окне.

Для классификации был выбран алгоритм KNN (алгоритм ближайших соседей), так как автор рекомендовал его как достаточно точный и быстродействующий алгоритм, который может быть применен в системах реального времени для распознавания жестов. Исходя из рисунка 27, он имеет довольно высокую точность. При этом автор статьи использовал K равным 2.

Таким образом, для исходного оборудования в виде микрокомпьютера Raspberry Pi Zero W и электрода 13E200 MyoBock, можно выделить следующий алгоритм распознавания миоэлектрических паттернов (рисунок 28):

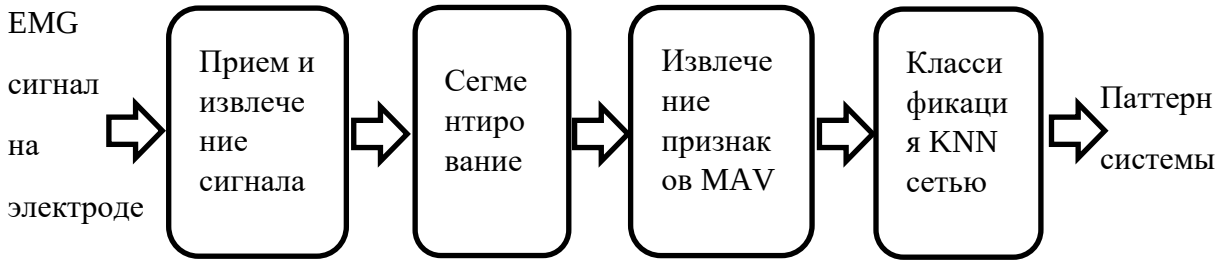

Рисунок 28 – Алгоритм распознавания паттернов для Raspberry Pi Zero W и электрода 13E200 MyoBock

Входным сигналом системы распознавания является значение ЭМГ сигнала, поступающего в систему путем считывания аналогового сигнала одноканального электрода 13E200 MyoBock аналогово-цифровым преобразователем (АЦП). Таким образом, для извлечения сигнала требуется обрабатывать N-битные значения, полученные с АЦП.

Сбор ЭМГ данных может быть осуществлен без предобработки в виде фильтрации, что значительно уменьшает нагрузку на микрокомпьютер. После приема сигнала и выделения паттерна, должно происходить разбиение (сегментирование) сигнала жеста на 10 окон, нахождение MAV признака для

каждого окна и классификация на основе полученных признаков KNN моделью с K равным 2.

Результатом работы алгоритма будет являться паттерн протеза, представленный значением от 0 до 1, где 0 - сжатие руки, а 1 – резкое сгибание руки.

3.2.2. **Реализация алгоритма распознавания миоэлектрических паттернов**

В первую очередь требовалось обеспечить прием данных с электрода 13E200 MyoBock на Rasbperry Pi Zero W. Прием данных с выхода контроллера может быть обеспечен только с АЦП. Согласно схеме из пункта 2.2 был использован внешний АЦП для Raspberry Pi Zero W - ADS1115, который поставляется в виде готового модуля и обеспечивает частоту дискретизации до 860Гц. Максимальное напряжение питание 5.5В. Данные характеристики полностью удовлетворяют характеристикам напряжения и ширины полосы частот 13E200 MyoBock.

По схеме из пункта 2.2 был собран макетный образец, представленный на рисунке 29. Макетный образец включает подключенный и закрепленный на руке электрод 13E200 MyoBock.

Электрод был закреплен эластичной повязкой для более плотного прилегания к мышце. Raspberry Pi, для исключения помех сети, питается от модуля Raspberry Pi Zero W UPS.

Взаимодействие с электродом осуществляется с помощью класса MyoelectronicsSensor, описанного в главе 3.1.5. Используя его, были собраны по 30 образов каждого паттерна. Примеры данных были приведены на рисунке 24 и рисунке 25. Сбор данных в таком количестве проводился для последующей тренировки KNN модели.

После сбора данных по жестам можно приступать к предобработке данных, получению признаков и классификации согласно определенному алгоритму.

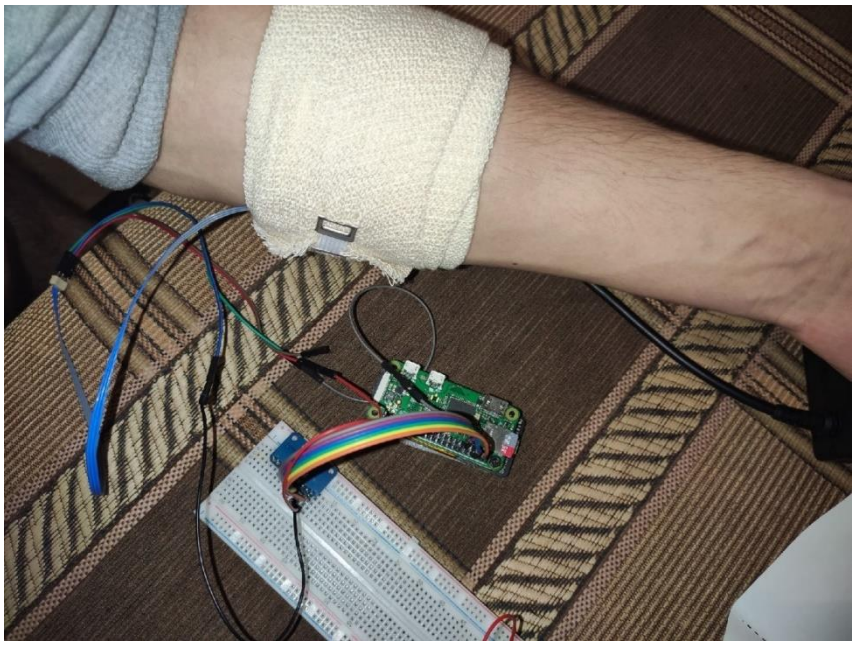

Рисунок 29 – Макетный образец MPR системы на Raspberry Pi Zero W и электроде 13E200 MyoBock

Для обучения KNN модели по каждому из полученных сигналов были извлечены паттерны путем прохода по каждому из записанных csv файлов с окном в 5 элементов и пороговым значением 400. На каждом шаге рассчитывается среднее по окну и извлекается сигнал при необходимости. После полного прохода по исходному сигналу, извлеченный сигнал жеста записывается в выходной файл.

Для непосредственной классификации используется модель KNN из пакета scikit-learn [69]. Во время обучения происходит загрузка размеченных извлеченных жестов на этапе предобработки. Загруженные жесты подвергаются разбиению на окна и извлечению признаков MAV с помощью статических методов разработанных классов FeatureExtractor и SignalSampler. SignalSampler обеспечивает разбиение сигнала на 10 окон одинакового размера. Количество окон может быть изменено передачей параметра count_of_samples. FeatureExtractor реализует расчет MAV характеристики по полученным окнам.

На выходе метода FeatureExtractor.extract_mav содержится коллекция MAV характеристик по 10 окнам сигнала. Данный выход используется в качестве входных данных для обучения KNN модели.

Для проверки точности по распознаванию жестов, перед обучением KNN, исходная коллекция MAV характеристик разбивается на обучающую и тестовую выборку в соотношении 4 к 5. Таким образом, из 30 наборов данных по каждому жесту, 6 из них будут использованы для валидации модели.

После обучения KNN модели была проведена валидация на основе RMSE (среднеквадратичная ошибка). RMSE и для тренировочной, и для тестовой выборки показала 0%, что является хорошим показателем. Это означает, что были выбраны жесты, которые пользователь протеза не сможет перепутать друг с другом и классификатор с высокой долей будет распознавать их корректно. В то же время, стоит подумать, как об увеличении количества жестов системы как минимум на один.

Стоит отметить, что в текущей версии scikit-learn не поддерживает переход моделей между архитектурами и выдает исключение при загрузке таких моделей. Так, модель обученная на x64 архитектуре не может быть использована на raspberry pi, которая базируется на x32 архитектуре. В связи с этим обучение стоит выполнять непосредственно на Raspberry Pi Zero W. Для 60 экземпляров данных это не создало проблем производительности, но для большего размера данных стоит попытаться решить данную проблему.

Таблица 6

Оценка результатов разработанной MPR системы

Точность классификации	100%
Время загрузки модели на Raspberry Pi Zero W	0.005мс
Время извлечения признаков и классификации на Raspberry Pi Zero W	0.016мс

После переобучения модели на Raspberry Pi Zero W был проведен замер временных характеристик разработанного алгоритма классификации жестов (таблица 6). Результаты полностью удовлетворяют требованиям систем реального времени.

Для извлечения и последующей обработки сигнала в реальном времени был разработан класс EmgSignalHandler, который включает в себя извлечение сигнала по окну в методе handle в отдельном потоке приложения. При достижении порога по окну происходит сегментирование сигнала и извлечение MAV признаков по сегментам. MAV признаки поступают в уже обученную KNN сеть и выполняется классификация. На выходе метода handle возвращается паттерн системы, либо None, если ни один из жестов не был найден.

Результаты работы в реальном времени неотличимы от оффлайн режима. Система распознавание выполняет классификацию жестов в реальном времени с точностью 100% параллельно с другими процессами протеза при подключенном ЭМГ канале.

3.3. Разработка программного обеспечения конфигурирования жестов протеза

3.3.1. Пользовательский интерфейс конфигурирования жестов протеза приложения для персонального компьютера

Для создания и изменения жестов на протезе предполагается использование приложения для персонального компьютера, которое должно обеспечивать изменение и заполнение полей структуры жеста, изображенного на рисунке 8.

Проанализировав структуру жеста, были сделаны следующие выводы о полях и соотнесенных им графических элементов:

1) id – уникальный идентификатор жеста. Не должен быть доступен пользователю для изменения и создания. Создание идентификатора должно проводиться автоматически. Отображаться элемент не должен.

2) name – имя жеста для голосового управления на мобильном приложении. Может изменяться пользователем и представлять из себя элемент TextBox в WPF.

3) last_time_sync – время последнего сохранения жеста в unix timestamp. Не должно изменяться пользователем, должно автоматически заполняться в приложении при сохранении жеста. Отображаться пользователю не должно.

4) iterable – флаг неограниченного повторения жеста. Может быть представлен как CheckBox в WPF.

5) repetitions – количество повторений жеста. Может быть представлено как TextBox с ограничением на положительные числа.

6) actions – коллекция действий в жесте. Может быть изменяема пользователями. Пользователю требуется предоставлять доступ к добавлению и удалению из коллекции. В WPF может быть представлено элементов ListBox, добавление и удаление может осуществляться с помощью элементов Button.

7) finger_positions – позиции пальцев протеза. Ограничены значениям от 0 до 180. Могут быть отображены в TextBox с ограничением на положительное числа от 0 до 180.

8) delay – задержка между установками положений. Может быть представлена как время в секундах в TextBox с ограничением на число с плавающей запятой.

Исходя из сделанных выводов был разработан графический интерфейс пользователя, представленный на рисунке 30, который состоит из следующих основных частей:

1) Список жестов системы.
2) Настройка количество итераций жеста.
3) Список жестов протеза.
4) Настройка положений пальцев протеза.

В программе реализована архитектура паттерна MVVM [64], которая используется как основная архитектура для приложений WPF. В связи с этим приложения, в общем случае, можно описать с помощью трех компонентов: моделей (Model), которые представляют бизнес-логику приложений, представлений пользовательского интерфейса на языке XAML (View) и представлений-моделей (ViewModel), в которых содержится вся логика построения графического интерфейса и ссылка на модели.

Исходя из архитектуры был разработан класс MainWindowViewModel, который содержит модель-представление каждой из основных частей интерфейса. Разметка интерфейса находится в MainWindow.xaml.

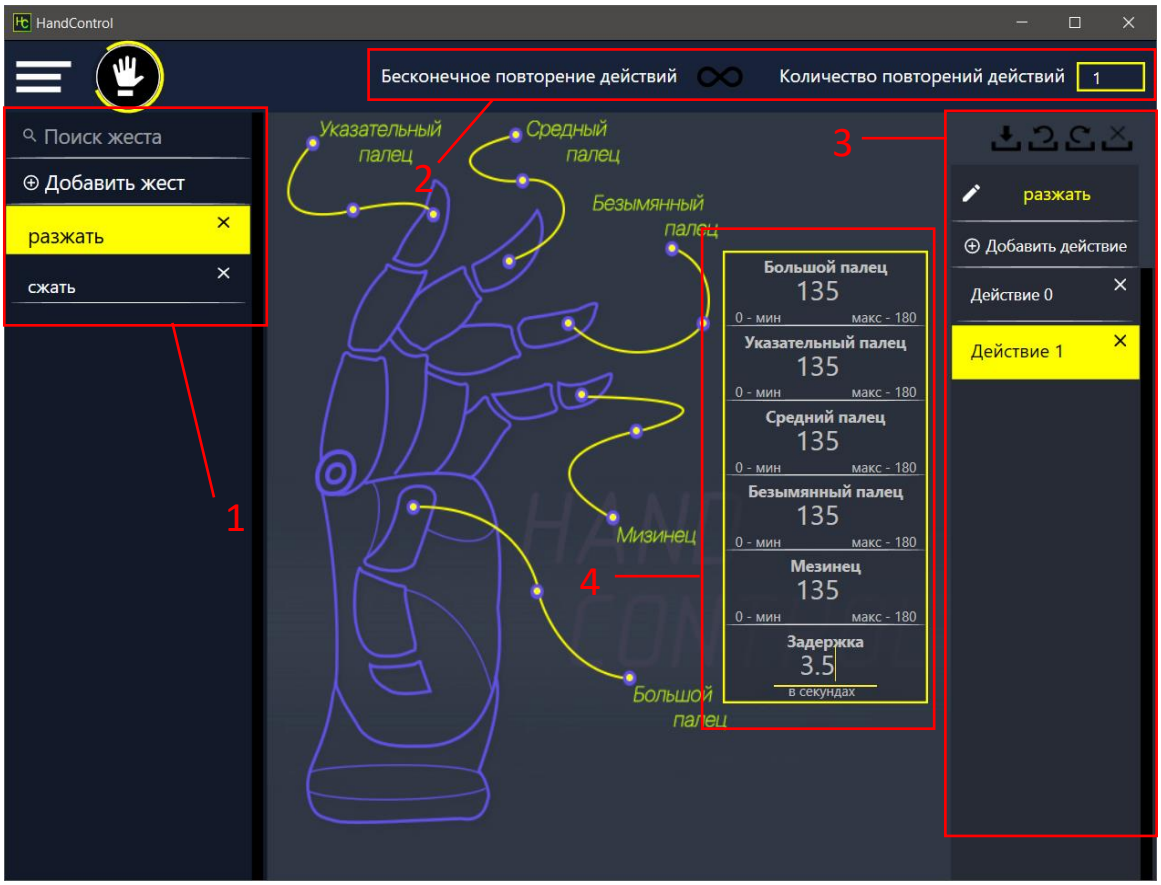

Рисунок 30 – Графический интерфейс пользователя для конфигурирования жестов с обозначением основных частей

Для обеспечения изменения жеста в интерфейсе во View, а соответственно и во ViewModel хранится полный список жестов ListGesture, который представляет из себя экземпляр коллекции с оповещением ObservableCollection для автоматического обновления во View при выполненной привязке (binding) к ListBox. Код привязки приведен на рисунке 31.

```
<ListBox
    Grid.Column="0"
    Background="Transparent"
    BorderThickness="0"
    ItemContainerStyle="{StaticResource StyleListBox}"
    ItemsSource="{Binding ListGesture}"
    ScrollViewer.VerticalScrollBarVisibility="Visible"
    SelectedItem="{Binding SelectedGesture, Mode=TwoWay}">
```

Рисунок 31 – Код привязки коллекции жестов к ListBox

Добавление жестов в коллекцию осуществляется по подписке на наблюдаемую коллекцию жестов из предварительно разработанного сервиса GestureService, который испускает элементы при загрузке жестов с жесткого диска, либо при приходе их с протеза во время синхронизации. Код инициализации изображен на рисунке 32.

```
0 references
public MainWindowViewModel(IProstheticManager prostheticManager, IGestureService gestureService)
{
    Prosthetic = prostheticManager ?? throw new ArgumentNullException(nameof(prostheticManager));
    Prosthetic.IsConnectionChanged.Subscribe(ProstheticConnectionChangeHandler);
    _gestureService = gestureService ?? throw new ArgumentNullException(nameof(gestureService));

    gestureService.Gestures.ObserveOn(Application.Current.Dispatcher).Subscribe(GestureAddOrUpdateHandler);

    ListGesture = new ObservableCollection<GestureModel>();
    ListGestureView = CollectionViewSource.GetDefaultView(ListGesture);
}

1 reference
private void GestureAddOrUpdateHandler(GestureModel gesture)
{
    var gestureInUiResult :IEnumerable<GestureModel> = ListGesture.Where(x :GestureModel => x.Id == gesture.Id);
    var inUiResult :GestureModel[] = gestureInUiResult as GestureModel[] ?? gestureInUiResult.ToArray();
    if (inUiResult.Any())
    {
        var gestureInUi :GestureModel = inUiResult.Single();
        ListGesture.Remove(gestureInUi);
    }

    ListGesture.Add(gesture);
    SortGestures(ListGesture, start:0, end:ListGesture.Count() - 1);
}
```

Рисунок 32 – Код инициализации и добавления списка жестов

После того, как какой-либо жест будет добавлен в коллекцию ListGesture, то во View пользователь увидит элемент, изображенный на рисунке 30 под пунктом 1, который отображает названия жестов в коллекции ListGesture. Пользователь может выполнить удаление элемента коллекции нажатием на крестик в интерфейсе, либо добавить новый жест кнопкой “Добавить жест”. После того как жест был отображен в интерфейсе пользователю предоставляется доступ к изменению его содержимого по нажатию на элемент списка в интерфейсе.

По нажатию на жест в пункте 1 из рисунка 30 пользователю отображаются элементы из пункта 2 и 3 для изменения кол-ва повторений жеста, его имени и списка действий жеста. Коллекция действий жеста

реализовано аналогично коллекции жестов, за исключением того, что инициализация коллекций действий происходит при выборе жеста в интерфейсе, а не по получению от сервисов, так как информация о действиях уже хранится в самом жесте. По нажатию, либо добавлению действия, пользователю отображается элемент из пункта 4 на рисунке 30 путем установки свойства SelectedAction.

Для валидации полей на минимальные и максимальные значения и тип в WPF не оказалось специальный элементов, удовлетворяющих видению удобного UX, поэтому для валидации на тип была написана обработка событий PreviewTextInput и LostFocus в которых осуществляется проверка типа и пустой строки. Код валидации строки во View изображен на рисунке 33.

```
6 references
private void NumberPreviewTextInput(object sender, TextCompositionEventArgs e)
{
    if (sender is TextBox textBox)
    {
        // Use SelectionStart property to find the caret position.
        // Insert the previewed text into the existing text in the textbox.
        var fullText:string = textBox.Text.Insert(textBox.SelectionStart, e.Text);

        // If parsing is successful, set Handled to false
        e.Handled = !int.TryParse(fullText,
            style:NumberStyles.AllowDecimalPoint | NumberStyles.AllowLeadingSign,
            CultureInfo.InvariantCulture, out var val:int);
    }
}

7 references
private void Number_OnLostFocus(object sender, RoutedEventArgs e)
{
    if (sender is TextBox textBox)
    {
        if (string.IsNullOrEmpty(textBox.Text))
        {
            textBox.Text = "0";
        }
    }
}
```

Рисунок 33 – Код валидации числовых полей

Таким образом ввод некорректных символов и пустой строки будет блокироваться во View, и пользователь не увидит некорректных символов при вводе значений в TextBox.

Валидация же минимальных и максимальных значений была написана во ViewModel в сеттере у числовых полей. При попытке пользователем установить значение меньше или больше, чем допустимое в поле будет установлено соответствующее пороговое значение. Пример кода валидации пороговых значений приведен на рисунке 34. MinPosition задано значением 0, MaxPosition значением 180, как и было описано исходя из структуры жеста.

```
/// <summary>
///     Gets or sets положение указательного пальца в градусах.
/// </summary>
12 references | 2/2 passing
public int PointerFinger
{
    get => _pointerFinger;
    set
    {
        if (value < MinPosition)
        {
            _pointerFinger = MinPosition;
            return;
        }

        if (value > MaxPosition)
        {
            _pointerFinger = MaxPosition;
            return;
        }

        _pointerFinger = value;
        OnPropertyChanged(nameof(PointerFinger));
    }
}
```

Рисунок 34 – Код валидации числовых полей на пороговые значения

По завершению настройки жеста пользователь может сохранить жест выполнив нажатие на кнопку сохранения из пункта 3 на рисунке 30. После чего жест будет сохранен локально и синхронизирован с подключенным протезом по протоколу передачи.

3.3.2. **Имплементация протокола передачи**

Коммуникация с контроллером бионического протеза осуществляется по протоколу программного уровня, заявленного в системе разрабатываемого бионического протеза в пункте 2.3.1.

Контроллером управления накладывается следующие ограничения: подключаемое устройство должно подключаться по Bluetooth интерфейсу по

RFCOMM протоколу и выступать master устройством, самостоятельно выполняя максимум один запрос за раз, но параллельно ожидая приход пакетов телеметрии.

Исходя из ограничений был составлен следующий список задач, для которых требуется написать программное обеспечение:

1) Обеспечить подключение к контроллеру управления по Bluetooth интерфейсу RFCOMM протоколу.

2) Обеспечить формирование и прием пакетов согласно формату протокола.

3) Поддержать протокольные команды с необходимой полезной нагрузкой.

В первую очередь был найден фреймворка для работы с Bluetooth - 32feet.NET [48], который может обеспечивать подключение как к BLE, так и к Bluetooth Classic. Для взаимодействия с Bluetooth устройствами через фреймворка был написан класс BluetoothService, который обеспечивает подключение конечной точки контролера протеза при вызове метода ConnectAsync. Метод ConnectAsync (см. рисунок 35) получает информации о Bluetooth устройстве, которое было предварительно ему передано вышестоящим классом DeviceBluetooth и выполняет подключение к устройству. После чего метод выполняет получение потока передачи для этого устройства, из которого можно читать и записывать данные. Если данные операции прошли успешно, то начинается асинхронный прием данных с устройства вызовом BeginRead на потоке. Когда данные будут приняты, то они будут направлены в обработчик BluetoothStreamReadHandler. Если подключение прошло успешно, то из метода возвращается true, иначе false.

```
2 references
public async Task<bool> ConnectAsync(Device device)
{
    var task = Task.Run( function: () =>
    {
        try
        {
            if (_bluetoothStream != null) return false;

            if (device == null) throw new ArgumentNullException(nameof(device));

            var bluetoothClient = new BluetoothClient();
            var ep = new BluetoothEndPoint(device.DeviceInfo.DeviceAddress, _serviceClassId);

            // connecting
            bluetoothClient.Connect(ep);

            // get stream for send the data
            _bluetoothStream = bluetoothClient.GetStream();

            _bluetoothStream.BeginRead(_readBuffer,  offset: 0,  size: _readBuffer.Length, BluetoothStreamReadHandler,
                _bluetoothStream);

            return true;
        }
        catch
        {
            return false;
        }
    });

    return await task;
}
```

Рисунок 35 – Код метода ConnectAsync

После того как устройство будет успешно подключено оно начнет принимать данные асинхронно и вызывать обработчик асинхронного события по приему. По приему в обработчик данные переводятся на асинхронную модель программирования Reactive Extension [68] для более удобной работы. Для этого в классе имеется Subject, который испускает приходящие данные. Стоит отметить, что прямое использование Subject в качестве Observable стоит исключить и заменить его на создание Observable через FromAsyncPattern [65].

После того, как из сервиса была получена наблюдаемая коллекция ReceivedDataObservable для приема входных данных, она используются в вышестоящем классе DeviceBluetooth. Для обработки принятых данных с Bluetooth устройства в конструкторе класса выполняется создание экземпляра парсера протокола ProtocolParser, который обеспечивает разбор входящего потока байт с целью получить пакет протокола.

При получении потока байт из наблюдаемой коллекции ReceivedDataObservable происходит их обработка парсером протокола. Метод Update в парсере реализует ‘прослушивание’ входного потока байт на наличие SDF, приём типа пакета и данных, а также проверку целостности пакета. В методе предусмотрена обработка ошибок таймаута, благодаря чему протез

сможет принимать следующий пакет, даже если предыдущий не был полностью доставлен из-за ошибок, связанных с неожиданным разрывом соединения. Парсер осуществляет свою работу на основании машины состояний, где каждый пришедшее поле фрейма ознаменует переход в следующее состояние. По завершению сборки входного пакета происходит испускание его в наблюдаемую коллекцию принятых пакетов ReceivedPackagesObservable.

Для отправки пакетов на котроллер управления был разработан метод SendToDeviceAsync. Код метода изображен на рисунке 36.

```
9 references
public async Task<byte[]> SendToDeviceAsync(CommandType command, byte[] payload)
{
    await _semaphoreSendPackage.WaitAsync();

    try
    {
        if (!IsBluetoothConnected)
        {
            throw new InvalidOperationException(message: "Device not connected");
        }

        var package :byte[] = _protocolParser.CreatePackage(command, payload);
        await _service.SendAsync(package);

        _packageResponseCompletionSource = new TaskCompletionSource<byte[]>();
        var cancellationTokenSource = new CancellationTokenSource();
        cancellationTokenSource.CancelAfter(delay: TimeSpan.FromSeconds(10));

        using (cancellationTokenSource.Token.Register(() =>
        {
            // this callback will be executed when token is cancelled
            _packageResponseCompletionSource.TrySetCanceled();
        }))
        {
            var response :byte[] = await _packageResponseCompletionSource.Task;
            return response;
        }
    }
    finally
    {
        _semaphoreSendPackage.Release();
    }
}
```

Рисунок 36 – Код отправки пакетов

В связи с ограничениями протокола передачи на контроллере управления метод организован следующим образом: при входе в метод выполняется взятие семафора для предотвращения отправки нескольких запросов за раз. После того, как семафор будет взят выполняется формирование пакета системы, вызовом CreatePackage и его отправка на

протез вызовом SendAsync у BluetoothService. В связи с тем, что протокол является запрос/ответ, то код отправки SendToDeviceAsync был организован оптимальный образом и представляет собой асинхронную задачу из модели асинхронного программирования TAP [59], так как эта модель наиболее удобна для ожидания ответа на запросы в отличии от реактивной модели, либо событийной. Для ожидания ответа создается TaskCompletionSource, результат которого устанавливается при асинхронном приеме следующего пакета. Код установки результата задачи приведен на рисунке 37. Так же отправка поддерживает отмену по истечению времени передачи. Время передачи ограничено 10 секундами.

```
1 reference
private void DataReceivedHandler(PackageDto package)
{
    if (_packageResponseCompletionSource == null)
    {
        throw new InvalidOperationException(message:"The package was not expected to be received.");
    }

    if (package.Command == CommandType.Error)
    {
        _packageResponseCompletionSource.SetException(new Exception(message:"Prosthetic error code"));
    }

    if (package.Crc != package.ReceivedCrc)
    {
        _packageResponseCompletionSource.SetException(new Exception(message:"Crc not equals"));
    }

    _packageResponseCompletionSource.SetResult(package.Payload);
}
```

Рисунок 37 – Код установки результата отправки пакета

После того как сервисы для отправки данных по Bluetooth были написаны они могут довольно легко быть использованы для передачи протокольных команд протезу. Для этого был написан класс ProstheticConnector, который является высокоуровневой абстракцией для работы с протезом.

Для примера рассмотрим сохранение/обновление жеста протеза. Отправка данной команды осуществляется в методе SaveGesturesAsync (рисунок 38). Метод принимает на входе экземпляр data transfer object и с помощью фреймворка AutoMapper конвертирует его экземпляр, являющийся

сгенерированным классом из сообщения Protocol Buffer - SaveGestures, который был описан на контроллере управления в proto-файле. После этого экземпляр сериализуется в массив байт (полезную нагрузку) и отправляется на протез с соответствующим типом команды 0x07. Ответ на запрос может быть получен вызовом await на результат вызова метода SaveGesturesAsync.

```
public Task SaveGesturesAsync(SaveGestureDto saveGestures)
{
    var protobufModel = _mapper.Map<SaveGestureDto, SaveGesture>(saveGestures);
    return _prostheticDevice.SendToDeviceAsync(CommandType.SaveGestures, protobufModel.ToByteArray());
}
```

Рисунок 38 – Код сохранение/обновление жеста протеза

3.3.3. **Конфигурирование и синхронизация жестов системы**

Выполнил разработку интерфейса пользователя и коммуникацию между протезом и десктоп приложением осталось выполнить их объединения для предоставления пользователю возможности сохранение и получения жестов протеза.

Для данной функции был написан класс GestureService. Данный сервис обеспечивает работу с жестами, хранимыми в системе. При включении приложения выполняется получение списка жестов из файловой системы компьютера. Жесты при этом хранятся в виде файлов с расширением «.gesture» и содержат сериализованные сообщения protocol buffer.

После загрузки жестов из файловой системы приложение может быть использовано для конфигурирование жестов. При сохранении жеста из пользовательского интерфейса будет вызван метод AddGestureAsync (см. рисунок 39), который обеспечивает сохранение жеста локально и на протез через вызов ранее рассмотренного метода SaveGesturesAsync.

```
2 references
public async Task AddGestureAsync(GestureModel gesture)
{
    try
    {
        var gestureDto = _mapper.Map<GestureModel, GestureDto>(gesture);
        var saveGesture = new SaveGestureDto()
        {
            Gesture = gestureDto,
            TimeSync = DateTime.Now
        };

        await _prostheticConnector.SaveGesturesAsync(saveGesture);
        _gesturesLocalStorage.Add(saveGesture);
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
        throw;
    }
}
```

Рисунок 39 – Код добавления жеста в систему

В случае, если протез не подключен во время сохранения жеста, то отправка происходить не будет и жест будет синхронизирован при следующем подключении протеза. Для этого была спроектирована и разработана специальная надстройка над работой с жестами протеза, которая должна поддерживаться в каждой подсистеме протеза, обеспечивающей локальное хранение жестов.

Для выполнения синхронизации жестов между устройствами было введено время последней синхронизации жестов, которое хранится на контроллере управления как для каждого жеста индивидуально, так и для всего списка жестов (общее последнее время синхронизации).

При подключении к протезу данное время принимается в пакете телеметрии и сравнивается с локальным временем синхронизации. Если присутствует десинхронизация жестов, то выполняется расчет и синхронизация жестов в методе SyncGesturesAsync. Код метода изображен на рисунке 40. Код обеспечивает определение устаревших, либо отсутствующих жестов на протезе и в локальном хранилище и при их наличии выполняет соответствующую отправку протокольной команды, либо сохранение на жесткий диск.

```
public async Task SyncGesturesAsync()
{
    var getGesturesRemote = await _prostheticConnector.GetGesturesAsync();
    var getLocalGestures = _gesturesLocalStorage.GetGestures();

    var newTimeSync = DateTime.Now;
    newTimeSync = new DateTime(
        ticks: newTimeSync.Ticks - (newTimeSync.Ticks % TimeSpan.TicksPerSecond),
        newTimeSync.Kind
    );

    if (getGesturesRemote.LastTimeSync != getLocalGestures.LastTimeSync)
    {
        var (absentLocal:IEnumerable<GestureDto>, outdatedLocal:IEnumerable<GestureDto>) =
            GetAbsentAndOutDatedGestures(source: getGesturesRemote.Gestures, filter: getLocalGestures.Gestures);

        var (absentRemote:IEnumerable<GestureDto>, outdatedRemote:IEnumerable<GestureDto>) =
            GetAbsentAndOutDatedGestures(source: getLocalGestures.Gestures, filter: getLocalGestures.Gestures);

        foreach (var gesture in absentRemote.Concat(outdatedRemote))
        {
            await _prostheticConnector.SaveGesturesAsync(new SaveGestureDto()
                { Gesture = gesture, TimeSync = newTimeSync }); // Task
        }

        foreach (var gesture in absentLocal.Concat(outdatedLocal))
        {
            _gesturesLocalStorage.Add(new SaveGestureDto()
                {Gesture = gesture, TimeSync = newTimeSync});

            var gestureModel = _mapper.Map<GestureDto, GestureModel>(gesture);
            _gestureReplaySubject.OnNext(gestureModel);
        }
    }

    var updateLastTimeSyncDto = new UpdateLastTimeSyncDto()
    {
        LastTimeSync = newTimeSync
    };

    await _prostheticConnector.UpdateLastTimeSyncAsync(updateLastTimeSyncDto);
    _gesturesLocalStorage.UpdateLastTimeSync(updateLastTimeSyncDto);
}
```

Рисунок 40 – Код синхронизации жестов в системе

3.3.4. Конфигурирования паттернов протеза в интерфейсе приложения

Пользователю предоставляется возможность связывать паттерны протеза, классифицируемые системой распознавания, с существующими жестами на основании интерфейса, приведенного на рисунке 41. Интерфейс представляет собой ItemsControl, который содержит коллекцию конфигураций паттернов, полученных командой GetMioPatterns. Стоит отметить, что пользователю отображаются не идентификаторы жестов, а их названия в привычном ему представлении. По нажатию кнопки сохранить, на протез отправляется команда SetMioPatterns, которая выполняет обновление конфигурации паттернов.

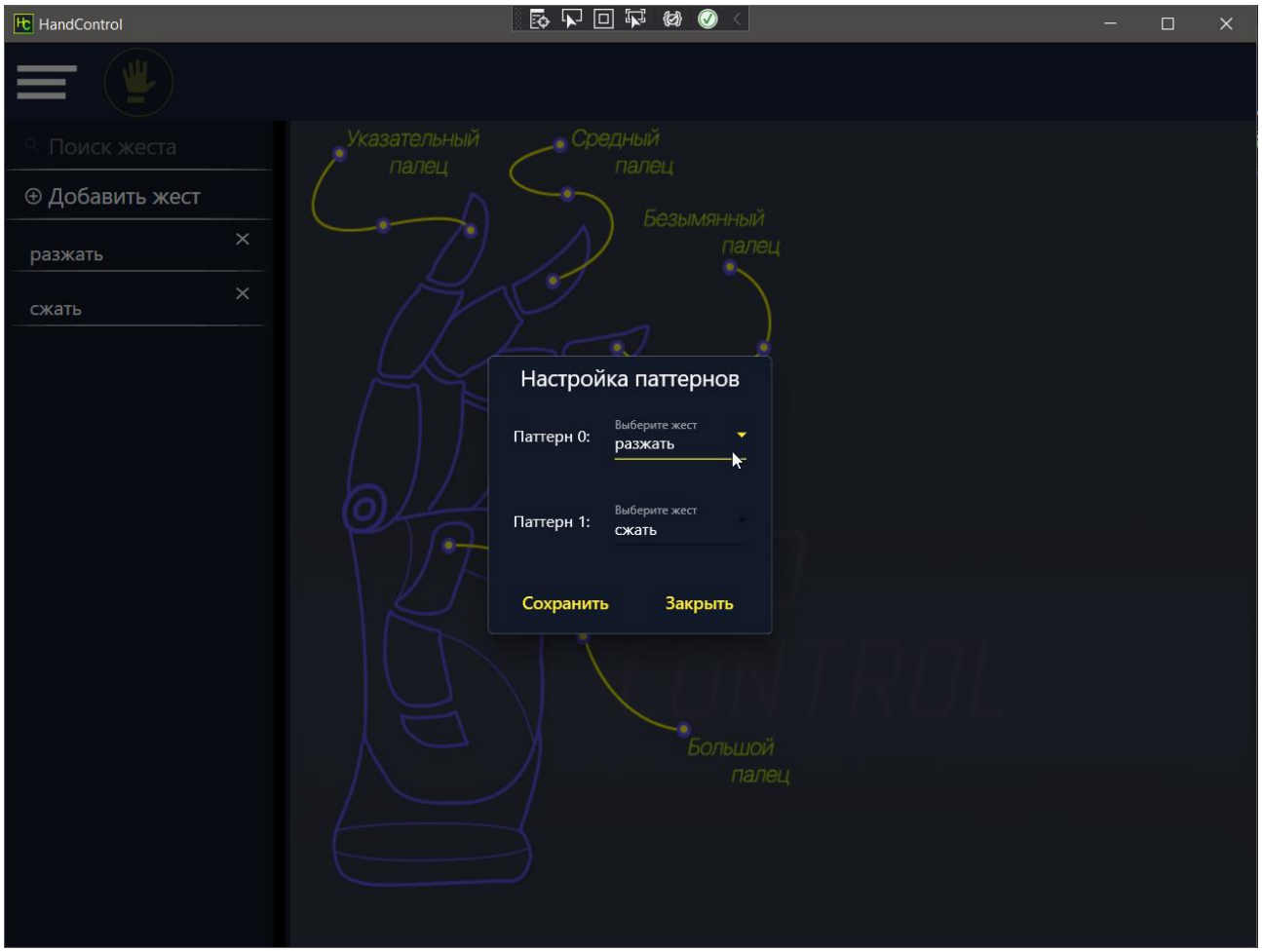

Рисунок 41 – Пользовательский интерфейс конфигурирования паттернов

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы проведено исследование состояния сферы протезирования верхних конечностей и бионических протезов, как наиболее эффективного решения для людей с ампутациями. Установлено, что одними из проблем бионических протезов являются ограниченность количества принимаемых положений и долгий процесс обучения использования. Данные проблемы сводятся к повышенным требованиям к тренировке мышц и концентрации пользователя при использовании миоэлектрического управления, что приводит к ограничениям на максимальное количество из 20 степеней свобод протеза и обучению использования до полу года.

Для решения данной проблемы была разработана аппаратно-программная система бионического протеза руки человека на базе Raspberry Pi Zero W, позволяющая самостоятельно создавать, конфигурировать и исполнять жесты протеза за счет подключаемых по Bluetooth пользовательских устройств, поддерживающих программный протокол протеза.

Проведен выбор и состав основных компонентов системы. Для контроллера управления протезом разработана структурная и принципиальная схемы, программное обеспечение и алгоритм классификации паттернов в реальном времени. Представлена 3D-модель протеза руки с исполняющим механизмом на линейных приводах. Для конфигурирования жестов и паттернов протеза разработан пользовательский интерфейс для персональных компьютеров на операционной системе Windows в виде .Net приложения на языке C#.

Разработанный бионический протез значительно расширяет спектр принимаемых протезом положений. Результаты показывают, что система, способна обеспечить пользователя исполнением индивидуально сконфигурированных жестов через:

1) Пользовательские устройства, подключенные по Bluetooth, и обеспечивающие возможность исполнения жестов в количестве 5000. Ограничение взято со значительным запасом и связано с объемом RAM при кешировании жестов на Raspberry Pi Zero W. В действительности в данном количестве нет необходимости и планируется использование пользователем максимального количества в 50–100 жестов.

2) Миоэлектрическое управление на основании алгоритма классификации паттернов через MAV признаки и KNN сеть. Для одноканального ЭМГ электрода и двух паттернов система выполняет распознавание с точностью более 95% при времени классификации 0.016мс.

Результатом работы является собранный экспериментальный образец системы (рисунок 42) в виде натурной модели бионического протеза с культеприемником и аппаратно-программным обеспечением. Экспериментальный образец готов к использованию в демонстрационных целях, в качестве выставочного экспоната, либо в качестве основы промышленного образца.

Рисунок 42 - Экспериментальный образец разработанного бионического протеза руки

По завершению разработки следует отметить направления, которые следует улучшить в системе:

1) Увеличить количество ЭМГ каналов системы до 4, чтобы обеспечить классификацию большого количество паттернов при миоэлектрическом управлении.

2) Ввести паттерн разблокировки миоэлектрического управления. Если система заблокирована, никакие другие паттерны не будут классифицированы, пока не будет обнаружен паттерн разблокировки. Дополнительный паттерн позволит избегать ненамеренных подергиваний мышц пользователя и, соответственно, неожиданного для него распознавания и исполнения жеста.

3) Разработать общую схему питания протеза. В текущем варианте Raspberry Pi Zero W питается от отдельного модуля UPS в связи с тем, что резкое отключение питания может повредить файлы операционной системы. Данное решение является неэффективным. Следует разработать схему питания, которая не будет разрывать цепь при нажатии на кнопку выключения, а будет выполнять отключение с задержкой. Подобное решение позволит безопасно обеспечивать Raspberry Pi питанием от аккумулятора протеза и исключить UPS модуль, что увеличит время работы протеза и упростит процесс зарядки.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Специальная литература

1. Васяева Н.С., Маркин К.А., Рожкин П.А. Разработка аппаратно-програмной системы автоматизированного перепрограммируемого протеза руки // International Journal of Advanced Studies - 2018. - Т. 8. - № 1-2. - С. 29-47

2. Ильиных А.Р., Салодкина П.С., Чигринова М.С. Бионические протезы: история, принцип работы и новейшие достижения // ЗАКОНОМЕРНОСТИ И ТЕНДЕНЦИИ ИННОВАЦИОННОГО РАЗВИТИЯ ОБЩЕСТВА сборник статей Международной научно-практической конференции.- 2020 -. С. 205-209

3. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения // Питер – 2019–410 с.

4. Рожкин П.А., Васяева Н.С. Разработка принципов надстройки для перепрограммируемой системы управления бионическим протезом // Глобализация и национальная безопасность: человек и общество в меняющемся мире. Двадцать вторые Вавиловские чтения: материалы международной междисциплинарной научной конференции (Йошкар-Ола, 6-7декабря 2018г.) - 2019. - Т. Часть 2. - С. 184-189

5. Рожкин П.А., Маркин К.А. Разработка автономного роботизированного протеза с возможностью перепрограммирования // Инженерные кадры - будущее инновационной экономики России: материалы Всероссийской студенческой конференции - 2018. - № 4. - С. 113-116

Иностранные источники

6. Advanced Amputee Solutions LLC. Amputee statistics you ought to know, 2012. http://www.advancedamputees.com/amputee-statistics-you-ought-know (date last accessed 9 December 2019).

7. Aszmann OC, Roche AD, Salminger S, et al. Bionic reconstruction to restore hand function after brachial plexus injury: a case series of three patients. Lancet 2015;385:2183–2189.

8. Blum, J.; Byram Hills, H. Using Force Sensors to Effectively Control a Below-Elbow Intelligent Prosthetic Device; Student Science: Washington, DC, USA, 2008.

9. Cipriani, C.; Controzzi, M.; Carrozza, M.C. The SmartHand transradial prosthesis. J. NeuroEng. Rehabil. 2011, 8, 1–14.

10. Cipriani, C.; Zaccone, F.; Micera, S.; Carrozza, M.C. On the shared control of an EMG-controlled prosthetic hand: Analysis of user–prosthesis interaction. IEEE Trans. Robot. 2008, 24, 170–184.

11. Cristina Piazza, Ann M. Simon, Kristi L. Turner, Laura A. Miller, Manuel G. Catalano, Antonio Bicchi, Levi J. Hargrove. Exploring augmented grasping capabilities in a multi-synergistic soft bionic hand. Journal of NeuroEngineering and Rehabilitation volume 17, Article number: 116 (2020)

12. Dentgen M., Renner S., Mottok J.: Equally Distributed Bus-Communication Access Rights for Inter MCU Communication Using Multimaster SPI. // Architecture of Computing Systems – ARCS – 2020 – 200-212p

13. Dr. Yu Wang, Warren Hunter, Mr. XiaoLin Chen, Housney Ahmed, Haneefah Safo. Impoved Hardware Design of IoT Prosthetic Device. 2018 ASEE Mid-Atlantic Fall Conference, October 26-27, 2018 – Brooklyn Technical High School

14. EMG-based Hand Gesture Recognition WithFlexible Analog Front End / S.Benatti∗, B.Milosevic†, F.Casamassima∗, P.Sch¨onle‡, P.Bunjaku‡, S.Fateh‡, Q.Huang‡, L.Benini // Biomedical Circuits and Systems Conference (BioCAS), 2014At: LausanneVolume: pp.57,60

15. Evaluation of surface EMG-based recognition algorithms for decoding hand movements / Sara Abbaspour, Maria Lindén, Hamid Gholamhosseini, Autumn Naber, Max Ortiz-Catalan // Medical & Biological Engineering & Computing (2020) 58:83–100

16. Farina D, Aszmann O. Bionic limbs: clinical reality and academic promises. Sci Transl Med 2014;6:257ps12.

17. Hafner BJ, Sanders JE. Considerations for development of sensing and monitoring tools to facilitate treatment and care of persons with lower limb loss. J Rehabil Res Dev. 2014; 51(1):1-14.

18. Herr HM, Grabowski AM. Bionic ankle-foot prosthesis normalizes walking gait for persons with leg amputation. Proc Biol Sci 2012;279:457–464.

19. Jennifer E Cheesborough, Lauren H Smith. Targeted Muscle Reinnervation and Advanced Prosthetic Arms. February 2015. Seminars in Plastic Surgery 29(1):62-72

20. J-R. R. Diego, Dan William C. Martinez*, Gerald S. Robles, and John Ryan C. Dizon. Development of Smartphone-Controlled Hand and Arm Exoskeleton for Persons with Disability. Open Engineering Volume 11 Issue 1:161-170.

21. Kalmin O.V., Kalmina O.A.: Miologiya: Uchebno-metodicheskoe posobie [Myology: Study guide]. In: Penza, IIC PGU, (2003)

22. Low-cost wearable Multichannel Surface EMG Acquisition for Prosthetic Hand Control / Davide Brunelli and Andualem Maereg Tadesse

23. Mahmoud Elkhodr, Seyed Shahrestani and Hon Cheung, EMERGING WIRELESS TECHNOLOGIES IN THE INTERNET OF THINGS: A COMPARATIVE STUDY, International Journal of Wireless & Mobile Networks (IJWMN) Vol. 8, No. 5, October 2016

24. Markin K., Rozhkin P., Platunov A. Development of a hardware-software system for a bionic prosthesis of a human hand // CEUR Workshop Proceedings - 2020, Vol. 2590, pp. 1-8

25. Marko Bumbaširević, Aleksandar Lesic, Tomislav Palibrk, Darko Milovanovic, Milan Zoka, Tamara Kravić-Stevović, and Stanisa Raspopovic. The current state of bionic limbs from the surgeon’s viewpoint. EFORT Open Rev. 2020 Feb; 5(2): 65–72.

26. Mastinu E, Doguet P, Botquin Y, Håkansson B, Ortiz-Catalan M. Embedded system for prosthetic control using implanted neuromuscular interfaces

accessed via an osseointegrated implant. IEEE Trans Biomed Circ Syst. 2017; 11(4):867–77.

27. Matthew Dyson, Jessica Barnes, Kianoush Nazarpour. Myoelectric control with abstract decoders. Journal of Neural Engineering, Volume 15, Number 5. 2 July 2018.

28. Ortiz-Catalan M, Håkansson B, Brånemark R. An osseointegrated human-machine gateway for long-term sensory feedback and motor control of artificial limbs. Sci Transl Med 2014;6:257re6.

29. Prosthetic Finger Movement Controller Based on EMG Signals using Statistical Feature and KNearest Neighbors / Attika Puspitasari, Achmad Rizal, Husneni Mukhtar // INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 9, ISSUE 03, MARCH 2020

30. Raspopovic S, Capogrosso M, Petrini FM, et al. Restoring natural sensory feedback in real-time bidirectional hand prostheses. Sci Transl Med 2014;6:222ra19.

31. Raspopovic S, Petrini FM, Zelechowski M, Valle G. Framework for the development of neuroprostheses: from basic understanding by sciatic and median nerves models to bionic legs and hands. Proc IEEE 2017;105:34–49.

32. Roshan James, Cato T. Laurencin. Regenerative Engineering and Bionic Limbs. Rare Metals. 2015 Mar 1; 34(3): 143–155.

33. SEMG Feature Extraction Based on Stockwell Transform Improves Hand Movement Recognition Accuracy / Haotian She, Jinying Zhu,Ye Tian, Yanchao Wang,Hiroshi Yokoi, Qiang Huang// Sensors, 14 October 2019

34. Shobha S, N Manu M, Santhosh Gagan T, S Shashank B, Ashwin Jayan. Implementation OF Prosthetic Robotic ARM Using Additive Manufacturing. International Journal of Progressive Research in Science and Engineering. Volume-1, Issue-3, June-2020. P91-99

35. Simone Benatti, Bojan Milosevic,Elisabetta Farella, Emanuele Gruppioni, Luca Benini. A Prosthetic Hand Body Area Controller Based on Efficient Pattern Recognition Control Strategies. Sensors 2017, 17(4), 869

36. Single channel surface EMG control of advanced prosthetic hands: A simple, low cost and efficient approach / Mahmoud Tavakoli, Carlo Benussi, Joao Luis Lourenco // Expert Systems With Applications Expert Systems with Applications, Volume 79, 15 August 2017, Pages 322-332

37. Sliman J. Bensmaia. Biological and bionic hands: natural neural coding and artificial perception. 2015 IEEE World Haptics Conference (WHC)

38. Sudarsan, S.; Sekaran, E.C. Design and Development of EMG Controlled Prosthetics Limb. Procedia Eng. 2012, 38, 3547–3551.

39. Tan DW, Schiefer MA, Keith MW, Anderson JR, Tyler J, Tyler DJ. A neural interface provides long-term stable natural touch perception. Sci Transl Med 2014;6:257ra138.

40. Van der Riet D, Stopforth R., Bright G., Diegel O., An Overview and Comparison of Upper Limb Prosthetics. In Proceedings of the 2013 AFRICON, Pointe-Aux-Piments, Mauritius, 9–12 September 2013; pp. 1–8.

41. Ventimiglia, P. Design of a Human Hand Prosthesis. Ph.D. Thesis, Worcester Polytechnic Institute, Worcester, MA, USA, 2012.

42. Wei Pan, Zhanhuai Li, Yansong Zhang, Chuliang Weng. The New Hardware Development Trend and the Challenges in Data Management and Analysis. Data Science and Engineering volume 3, pages263–276 (2018)

43. Xiaozhou Meng; Thornberg, B.; Olsson, L., Component obsolescence management model for long life cycle embedded system; 2012 IEEE AUTOTESTCON Proceedings AUTOTESTCON, 2012 IEEE. :19-24 Sep 2012

44. Yamakawa, S.; Nojima, T. A Proposal for a MMG-Based Hand Gesture Recognition Method. In Proceedings of the 25th annual ACM symposium on User interface software and technology, Cambridge, MA, USA, 7–10 October 2012; pp. 89–90.

45. Yılmaz GÜVEN, Ercan COŞGUN, Sıtkı KOCAOĞLU, Harun GEZİCİ, Eray YILMAZLAR. Understanding the Concept of Microcontroller Based Systems To Choose The Best Hardware For Applications. International Journal of Engineering And Science Vol.6, Issue 9 (September 2017), PP -38-44

46. Zachary A. Collier, James H. Lambert, Managing obsolescence of embedded hardware and software in secure and trusted systems, Frontiers of Engineering Management volume 7, pages172–181 (2020)

47. Zollo, L.; Roccella, S.; Guglielmelli, E.; Carrozza, M.C.; Dario, P. Biomechatronic design and control of an anthropomorphic artificial hand for prosthetic and robotic applications. IEEE/ASME Trans. Mechatron. 2007, 12, 418–429.

Электронные ресурсы

48. 32feet.NET - Personal Area Networking for .NET [Электронный ресурс]. Режим доступа: https://github.com/inthehand/32feet

49. A prosthetic arm for contactless payments could turn you into a superhuman [Электронный ресурс]. Режим доступа: https://www.rbth.com/science-and-tech/326477-prosthetic-arm-for-contactless-payments

50. ADS111x Datasheet [Электронный ресурс]. Режим доступа: https://www.ti.com/lit/ds/symlink/ads1113.pdf?ts=1621494284474&ref_url=https%253A%252F%252Fwww.google.com%252F

51. BeBionics Hand [Электронный ресурс]. Режим доступа: http://www.bebionic.com

52. Blue Dot [Электронный ресурс]. Режим доступа: https://bluedot.readthedocs.io/en/latest/index.html

53. Bluetooth 101 – Part VI – Bluetooth Architecture [Электронный ресурс]. Режим доступа: https://hearinghealthmatters.org/waynesworld/2014/bluetooth-101-part-vi/#:~:text=Bluetooth%20protocol%20stack%20consists%20of,sits%20on%20top%20of%20this.,

54. Bluetooth overview [Электронный ресурс]. Режим доступа: https://developer.android.com/guide/topics/connectivity/bluetooth

55. Bluetooth vs Wifi for the Internet of Things (IoT) [Электронный ресурс]. Режим доступа: https://www.quicsolv.com/blog/internet-of-things/bluetooth-vs-wifi-comparison-iot-solutions/

56. ElPROCUS: Overview on electronic communication protocols [Электронный ресурс]. Режим доступа: https://www.elprocus.com/communication-protocols/

57. Embedded System – Characteristics, Types, Advantages & Disadvantages [Электронный ресурс]. Режим доступа: https://electricalfundablog.com/embedded-system-characteristics-types-advantages-disadvantages/

58. i-LIMB [Электронный ресурс]. Режим доступа: http://www.touchbionics.com

59. Interop with Other Asynchronous Patterns and Types [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/interop-with-other-asynchronous-patterns-and-types

60. John M. Miguelez, MacJulian Lang, Robert Dodson, Mariya Cameron, Cullen Hays. BUILDING A BETTER ARM: DESIGN AND FABRICATION IN UPPER-LIMB PROSTHETICS [Электронный ресурс]. Режим доступа: https://opedge.com/Articles/ViewArticle/2016-07-01/2016-07_02

61. Limb Loss Statistics [Электронный ресурс]. Режим доступа http://www.amputee-coalition.org/limb-loss-resource-center/resources-by-topic/limb-loss-statistics/limb-loss-statistics

62. Michelangelo Hand [Электронный ресурс]. Режим доступа: http://www.ottobockus.com/prosthetics/upper-limb-prosthetics/

63. Miniature Linear Motion Series PQ12-P [Электронный ресурс]. Режим доступа: https://actuonix.s3.amazonaws.com/Actuonix+PQ12+Datasheet.pdf

64. MVVM: полное понимание (+WPF) Часть 1 [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/338518/

65. Observable.FromAsyncPattern [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/en-us/previous-versions/dotnet/reactive-extensions/hh229052(v=vs.103)

66. Protocol Buffers [Электронный ресурс]. Режим доступа: https://developers.google.com/protocol-buffers

67. Raspberry Pi OS [Электронный ресурс]. Режим доступа: https://www.raspberrypi.org/documentation/raspbian/

68. ReactiveX [Электронный ресурс]. Режим доступа: http://reactivex.io/

69. The k-Nearest Neighbors (kNN) Algorithm in Python [Электронный ресурс]. Режим доступа: https://realpython.com/knn-python/#fit-knn-in-python-using-scikit-learn

70. TUF GAMING X570-PLUS (WI-FI) Specifications [Электронный ресурс]. Режим доступа: https://www.asus.com/us/Motherboards/TUF-GAMING-X570-PLUS-WI-FI/specifications/

71. YOUBIONIC ARM [Электронный ресурс]. Режим доступа: https://www.youbionic.com/arm

72. Какие беспроводные интерфейсы есть в ноутбуках [Электронный ресурс]. Режим доступа: https://faqhard.ru/base/16/16.php

73. Модель вычислений Кана [Электронный ресурс]. Режим доступа: https://studwood.ru/1042173/informatika/model_vychisleniy_kana

74. Раскладываем по полочкам параметры АЦП [Электронный ресурс]. Режим доступа: https://habr.com/ru/company/milandr/blog/528164/

75. Счетная палата раскритиковала систему помощи инвалидам в России [Электронный ресурс]. Режим доступа: https://www.rbc.ru/society/20/09/2019/5d836e2a9a794741bd7bec2f

76. Технологии беспроводной передачи данных ZigBee, BlueTooth, Wi-Fi [Электронный ресурс]. Режим доступа: https://wireless-e.ru/standarty/tehnologii-besprovodnoj-peredachi-dannyh/

77. Частный бизнес и рынок протезирования [Электронный ресурс]. Режим доступа: https://www.if24.ru/chastnyj-biznes-i-rynok-protezirovaniya/