**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc167839912)

[1 ОБЗОР ЛИТЕРАТУРЫ 8](#_Toc167839913)

[1.1 Обзор аналогов 8](#_Toc167839914)

[1.2 Обзор протоколов беспроводной связи 11](#_Toc167839919)

[1.3 Обзор интерфейсов 14](#_Toc167839923)

[1.4 Обзор библиотеки OpenCV 15](#_Toc167839926)

[2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ 17](#_Toc167839927)

[2.1 Описание основных блоков устройства управления 17](#_Toc167839928)

[2.2 Описание основных блоков программы 20](#_Toc167839937)

[3 ФУНКЦИОНАЛНОЕ ПРОЕКТИРОВАНИЕ 23](#_Toc167839946)

[3.1 Блоки устройства управления 23](#_Toc167839947)

[3.2 Описание функций прошивки устройства управления 32](#_Toc167839956)

[3.3 Описание функционала программной части 34](#_Toc167839963)

[3.4 Описание диаграммы классов 34](#_Toc167839964)

[4 ПРИНЦИПИАЛЬНОЕ ПРОЕКТИРОВАНИЕ 40](#_Toc167839975)

[4.1 Компоненты устройства 40](#_Toc167839976)

[4.3 Разработка схемы питания 48](#_Toc167839984)

[5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 50](#_Toc167839985)

[5.1 Разработка пользовательского интерфейса 50](#_Toc167839986)

[5.2 Функция распознавание мяча 54](#_Toc167839990)

[5.3 Функция connectToPlayer 55](#_Toc167839991)

[6 СБОРКА УСТРОЙСТВА 56](#_Toc167839992)

[6.1 Разработка печатной платы 56](#_Toc167839993)

[6.2 Разработка корпуса устройства 59](#_Toc167839996)

[6.3 Сборка устройства 61](#_Toc167839997)

[7 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 63](#_Toc167839998)

[7.1 Объекты тестирования 63](#_Toc167839999)

[7.1 Порядок проведения тестов для аппаратной части 63](#_Toc167840000)

[7.2 Подготовка тестовой среды 63](#_Toc167840001)

[7.3 Методика тестирования подачи сигнала передвижения модели игрока 64](#_Toc167840005)

[7.4 Методика тестирования подачи сигнала отталкивания мяча 65](#_Toc167840006)

[7.5 Тестирование подачи сигнала восстановления жизней игрока 65](#_Toc167840007)

[7.6 Тестирование отображения информации на экране 65](#_Toc167840008)

[7.7 Тестирование воспроизведения звукового сигнала 66](#_Toc167840009)

[7.8 Порядок проведения тестов для программной части 66](#_Toc167840010)

[7.9 Подготовка тестовой среды 67](#_Toc167840011)

[7.10 Тестирование Wi-Fi соединения 67](#_Toc167840015)

[7.11 Тестирование подачи сигнала управления моделью игрока 68](#_Toc167840016)

[7.12 Тестирование управления и отображения количества жизней игрока 68](#_Toc167840017)

[7.13 Тестирование отображения изображения с камеры модели игрока 69](#_Toc167840018)

[7.14 Тестирование обнаружения мяча 69](#_Toc167840019)

[8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 70](#_Toc167840020)

[8.1 Руководство пользователя аппаратной части 70](#_Toc167840021)

[8.2 Руководство пользователя программной части 76](#_Toc167840027)

[9 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И РЕАЛИЗАЦИИ АППАРАТНО-ПРОГРАММНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ СИМУЛЯТОРА КОМПЬЮТЕРНОЙ ИГРЫ «ПИНГ-ПОНГ» 80](#_Toc167840030)

[9.1 Характеристика аппаратно-программной системы 80](#_Toc167840031)

[9.2 Расчёт экономического эффекта от производства аппаратно-программной системы 80](#_Toc167840032)

[9.3 Расчёт инвестиций в проектирование и производство аппаратно-программного комплекса 85](#_Toc167840033)

[9.4 Расчёт показателей экономической эффективности инвестиций в проектирование и производство аппаратно-программного комплекса 87](#_Toc167840037)

[ЗАКЛЮЧЕНИЕ 88](#_Toc167840038)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 89](#_Toc167840039)

[ПРИЛОЖЕНИЕ А 90](#_Toc167840040)

[ПРИЛОЖЕНИЕ Б 91](#_Toc167840041)

[ПРИЛОЖЕНИЕ В 100](#_Toc167840042)

[ПРИЛОЖЕНИЕ Г 114](#_Toc167840043)

[ПРИЛОЖЕНИЕ Д 115](#_Toc167840044)

[ПРИЛОЖЕНИЕ Е 116](#_Toc167840045)

# ВВЕДЕНИЕ

Симуляторы компьютерных игр являются популярным и востребованным видом развлечения, предоставляющим возможность пользователям погрузиться в виртуальный мир и испытать ощущение участия в различных игровых событиях. Одним из таких симуляторов является симулятор компьютерной игры «Пинг-Понг». Компьютерная игра «Пинг-Понг» является классической аркадной игрой, в которой игроки управляют «ракетками» для отбивания мяча и попытки забить его в ворота противника. Игра обычно проходит на двумерном поле, разделенном сеткой на две половины. Каждый игрок контролирует свою «ракетку», перемещая ее вверх и вниз по полю, чтобы отразить мяч и вернуть его противнику. Цель игры состоит в том, чтобы набрать больше очков, чем противник, забивая мяч в ворота соперника и предотвращая сопернику сделать то же самое.

Правила игры в Пинг-Понг достаточно просты и понятны. Игра начинается с подачи мяча одним из игроков. Мяч должен пересечь сетку и должен быть отражен «ракеткой» противника. Если игрок не смог отразить мяч, и он попадает в его ворота, то соперник получает очко. Игра продолжается до определенного количества очков, обычно установленного заранее, или до того момента, когда один из игроков набирает достаточное количество очков для победы.

В рамках данного дипломного проекта симулятор компьютерной игры Пинг-Понг представляет собой прямоугольное поле, на котором расположены друг напротив друга модели игроков. Модели игроков представляют собой подвижные платформы, осуществляющее передвижение влево и вправо по плоской горизонтальной поверхности, с толкающим механизмом, которые управляются удалённо. Это обеспечит равные условия для всех игроков, поскольку скорость передвижения и сила удара по мячу будут одинаковыми.

Целью данного дипломного проекта является разработка и проектирование аппаратно-программной системы управления для симулятора компьютерной игры «Пинг-Понг».

В соответствии с поставленной целью были определены следующие

задачи:

* разработка структуры аппаратного и программного обеспечения;
* разработка функциональных узлов;
* выбор и обоснование элементной базы устройства;
* разработка компонентов устройства;
* разработка программного обеспечения для корректной обработки информации;
* настройка компонентов проекта;
* тестирование и проведение испытаний разработанной системы.

Проект будет включать в себя разработку пульта управления. Пульт управления представляет собой следующий функционал:

* подключение пульта управления к игровой платформе по радиоканалу;
* возможность персонализации, которая представляет собой осуществление выбора игрока;
* передача сигналов управления движением игрока с помощью джойстика или кнопок, расположенных на боковой части пульта управления;
* передача сигнала отталкивания мяча при нажатии определённой кнопки пульта управления с последующей активации отталкивающего механизма, расположенного на модели игрока;
* отображение информации о подключении пульта управления к игровой платформе, а также информации о состоянии зарядки и разрядки.

Также проект предполагает разработку приложения, совместимого с операционной системой Android, которое также включает в себя функции подключения к платформе, управление движением модели игрока и отталкивание мяча. Отличительный функционалом приложения является захват видеоданных с модели игрока, а также последующем распознаваниям мяча в потоке видеоданных для определения его положения на игровой платформе.

# 1 ОБЗОР ЛИТЕРАТУРЫ

## Обзор аналогов

На одном из начальных этапов разработки, обзор аналогов, рассматриваются существующие решения и проекты, связанные с аналогичными функциональными возможностями и целями разрабатываемой аппаратно-программной системы. На данном этапе изучается и анализируется различные аналоги, чтобы получить представление о том, какие технологии, методы и подходы уже существуют на рынке и как они могут быть применены к разрабатываемой аппаратно-программной системе. Это позволит использовать лучшие практики и идеи из уже существующих решений, а также определить уникальные особенности и преимущества разрабатываемой программно-аппаратной системе.

### 1.1.1 Wii Remote

Wii Remote представляет собой беспроводной контроллер, который подключается к игровой платформе через Bluetooth. Он имеет удобную форму, которая легко помещается в руке игрока. Контроллер оснащен различными кнопками, включая кнопку "A" и "B", кнопки управления вверх/вниз/влево/вправо, а также кнопку "Home" для возврата в главное меню.

Однако главной особенностью Wii Remote является наличие встроенного акселерометра и гироскопа. Эти сенсоры позволяют контроллеру определять движение и ориентацию в пространстве. Игрок может управлять моделью игрока, двигая пульт вправо и влево, а также отталкивать мяч, совершая движения рукой.

На рисунке 1.1 представлено изображение данного устройства управления.



Рисунок 1.1 – Игровой контроллер Wii Remote [1]

В устройстве нет дисплея для отображения информации и джойстика, что отличает его от разрабатываемой системы.

### 1.1.2 PlayStation Move

PlayStation Move подключается к игровой платформе через Bluetooth, обеспечивая беспроводное соединение. Контроллер имеет эргономичную форму, которая лежит удобно в руке игрока. Он оснащен кнопками "Move", "Triangle", "Circle", "Cross" и "Square", а также кнопкой "PS" для возврата в главное меню.

Главной особенностью PlayStation Move является его способность точно отслеживать движения игрока в трехмерном пространстве. Контроллер оснащен гироскопом, акселерометром и магнитометром, которые в совокупности позволяют определить его положение и ориентацию в пространстве. Благодаря этому, игрок может управлять моделью игрока в, совершая жесты и движения рукой.

На рисунке 1.2 представлено изображение данного устройства управления.



Рисунок 1.2 – Игровой контроллер PlayStation Move [2]

В устройстве нет дисплея для отображения информации, что отличает его от разрабатываемой системы.

### 1.1.3 Joy-Con

Joy-Con контроллеры являются частью игровой консоли Nintendo Switch. Каждый Joy-Con состоит из двух отдельных контроллеров, правого и левого, которые могут быть использованы вместе или отдельно.

Каждый Joy-Con имеет встроенный акселерометр и гироскоп, что позволяет отслеживать движения и ориентацию контроллера в пространстве. Они также имеют вибромоторы для тактильной обратной связи.

Правый Joy-Con включает в себя кнопки A, B, X, Y, плюсовой джойстик, кнопку HOME для возврата в главное меню, а также кнопки для управления звуком и захвата экрана. Он также имеет встроенный инфракрасный датчик движения, который позволяет контроллеру распознавать движения в пространстве и взаимодействовать с другими Joy-Con контроллерами.

Левый Joy-Con включает в себя кнопку D-пада, минусовой джойстик, кнопку захвата видео и кнопку для синхронизации с консолью Nintendo Switch. Он также имеет встроенный инфракрасный датчик движения и может использоваться как независимый контроллер или соединяться с правым Joy-Con для создания полноценного геймпада.

На рисунке 1.3 представлено изображение данного устройства управления.



Рисунок 1.3 – Игровые контроллер Joy-Con [3]

В устройстве нет дисплея для отображения информации и звуковой индикации, что отличает его от разрабатываемой системы.

### 1.1.4 Steam Controller

Steam Controller – это контроллер, разработанный компанией Valve, который предназначен для использования с играми на платформе Steam. Он предлагает уникальный подход к управлению играми и имеет ряд особенностей.

Steam Controller имеет два симметрично размещенных джойстика, расположенных на верхней части контроллера. Они могут быть использованы как традиционные джойстики или как большие трекпады для точного контроля курсора или камеры в играх. Это обеспечивает гибкость в выборе способа управления в зависимости от игры и предпочтений игрока.

Контроллер также оснащен множеством дополнительных кнопок, включая кнопки A, B, X, Y, кнопку "гриппер" (grip) на каждой стороне для дополнительных команд, а также кнопки для управления звуком и захвата экрана. Он также имеет встроенный акселерометр и гироскоп для отслеживания движений контроллера в пространстве.

Одной из особенностей Steam Controller является его настраиваемость. Пользователи могут создавать и настраивать свои собственные профили управления для каждой игры, определяя функции кнопок и параметры управления. Это позволяет игрокам создавать оптимальные настройки для различных жанров игр или своих индивидуальных предпочтений.

На рисунке 1.4 представлено изображение данного устройства управления.



Рисунок 1.4 – Игровые контроллер Steam Controller [4]

В устройстве нет дисплея для отображения информации и звуковой индикации, что отличает его от разрабатываемой системы.

## 1.2 Обзор протоколов беспроводной связи

Взаимодействие программной части с игровой платформой реализуется с помощью технологий Wi-Fi и Bluetooth. Взаимодействие аппаратной части с игровой платформой реализуется по радиоканалу с помощью радиочастотного модуля.

### 1.2.1 Обзор протокола Wi-Fi

Wi-Fi (Wireless Fidelity) [5] – это беспроводной протокол, который позволяет устройствам подключаться к беспроводным сетям и обмениваться данными. Этот протокол использует радиочастоты для передачи данных между устройствами.

Основные характеристики протокола Wi-Fi:

1. Скорость передачи данных: Wi-Fi обеспечивает высокую скорость передачи данных, которая может достигать нескольких гигабит в секунду. Это позволяет пользователям быстро загружать и скачивать файлы, стримить видео, играть в онлайн-игры и многое другое.
2. Диапазоны частот: Wi-Fi работает в различных диапазонах частот, таких как 2,4 ГГц и 5 ГГц. Частота 2,4 ГГц обеспечивает большую покрытие и проникновение через стены, но может быть подвержена помехам от других устройств, таких как микроволновые печи и беспроводные телефоны. Частота 5 ГГц обеспечивает более высокую скорость и меньше помех, но имеет более ограниченную область покрытия.
3. Безопасность: Протокол Wi-Fi предоставляет различные методы безопасности, такие как WEP, WPA и WPA2, для защиты от несанкционированного доступа к сети. Это позволяет пользователям создавать пароли и шифровать свои беспроводные сети, чтобы предотвратить несанкционированное использование и защитить свои данные.
4. Различные режимы работы: Wi-Fi поддерживает различные режимы работы, включая точку доступа, клиент-сервер и мост. Это позволяет устройствам работать в разных режимах в зависимости от их функциональности и потребностей.

### 1.2.2 Обзор протокола Bluetooth

Bluetooth [6] представляет собой беспроводной протокол, который позволяет устройствам на короткие расстояния (обычно до 10 метров) обмениваться данными. Этот протокол был разработан для обеспечения низкопотребляющего подключения переносных устройств без необходимости использования проводов.

Основные характеристики протокола Bluetooth:

1. Удобство подключения: Bluetooth предоставляет простой и удобный способ подключения устройств друг к другу. С помощью этого протокола вы можете легко подключить наушники, клавиатуру, мышь, акустическую систему и другие устройства к своему компьютеру, смартфону или планшету без использования проводов.
2. Низкая потребляемая мощность: Bluetooth разработан с учетом энергосбережения, что делает его идеальным для переносных устройств, таких как наушники или умные часы, которые работают на батарейках. Протокол Bluetooth потребляет меньше энергии, чем Wi-Fi, что позволяет устройствам работать дольше без необходимости зарядки.
3. Различные профили: Bluetooth поддерживает различные профили, которые определяют типы устройств и их функциональность. Например, есть профиль для передачи аудио (A2DP), профиль для передачи файлов (OBEX), профиль для гарнитуры (HSP), профиль для клавиатуры и мыши (HID) и многое другое. Это позволяет устройствам быть совместимыми и выполнять различные функции.
4. Безопасность: Протокол Bluetooth обеспечивает безопасность данных, передаваемых между устройствами, путем использования шифрования. Это предотвращает несанкционированный доступ к передаваемым данным и защищает личную информацию пользователей.
5. Совместимость: Bluetooth является стандартом, поддерживаемым большинством устройств. Это означает, что вы можете легко подключить Bluetooth-устройство к другому совместимому устройству без проблем с совместимостью.

### 1.2.3 Обзор протокола ESB

Протокол ESB [7] (Enhanced ShockBurst) – это беспроводной протокол, разработанный компанией Nordic Semiconductor, который используется в их беспроводных радиочипах, таких как серия NRF24 и NRF52. Этот протокол предоставляет надежное и эффективное беспроводное соединение для передачи данных на короткие расстояния.

Основные характеристики протокола Enhanced ShockBurst:

1. Высокая надежность: ESB использует усовершенствованный метод передачи данных, называемый Enhanced ShockBurst, который обеспечивает высокий уровень надежности передачи данных. Этот метод включает в себя проверку целостности данных, повторную передачу в случае потери пакетов и механизмы обнаружения и исправления ошибок, что делает передачу данных стабильной и надежной.
2. Низкая задержка: Протокол ESB обеспечивает низкую задержку передачи данных. Это особенно важно для приложений, где требуется быстрая передача данных, например, в системах мониторинга здоровья, беспроводных устройствах управления и других приложениях реального времени.
3. Низкое энергопотребление: ESB разработан с учетом энергосбережения, что делает его идеальным для батарейных устройств. Протокол использует различные методы оптимизации энергопотребления, такие как режимы сна и быстрое переключение между активным и неактивным состояниями, чтобы продлить время работы устройств на батарейках.
4. Простота использования: ESB предоставляет простой и удобный способ использования для разработчиков. Он имеет простой API и набор команд, которые позволяют разработчикам легко настроить и управлять беспроводным соединением.
5. Гибкость: ESB поддерживает различные режимы работы, включая однонаправленную и двунаправленную передачу данных. Он также поддерживает различные режимы энергосбережения, которые позволяют устройствам работать в разных режимах в зависимости от их функциональности и требований.

## 1.3 Обзор интерфейсов

Интерфейсы I2C и SPI являются двумя широко используемыми последовательными интерфейсами для связи между микроконтроллерами и периферийными устройствами. Оба интерфейса обеспечивают передачу данных, но имеют свои особенности и преимущества.

### 1.3.1 Обзор интерфейса I2C

I2C [8] (Inter-Integrated Circuit) – это последовательный интерфейс, который широко используется во многих электронных устройствах для связи между микроконтроллерами и периферийными устройствами.

Основные характеристики интерфейса I2C:

1. Мультиплексирование: I2C позволяет подключать несколько устройств к одной шине, используя только две линии - линию данных (SDA) и линию тактирования (SCL). Это позволяет экономить пины микроконтроллера и упрощает подключение устройств.

2. Простота использования: I2C обладает простым и понятным протоколом, что делает его легким в использовании и программировании. Он использует адресацию, чтобы идентифицировать каждое устройство на шине, и поддерживает механизмы для обнаружения ошибок и повторной передачи данных.

3. Скорость передачи данных: I2C обычно работает на низких скоростях передачи данных, обычно до 100 кбит/с (стандартный режим) или до 400 кбит/с (расширенный режим). Это может быть недостатком в случае, когда требуется высокоскоростная передача данных.

4. Гибкость: I2C поддерживает различные режимы работы, включая однонаправленную и двунаправленную передачу данных. Он также поддерживает различные режимы энергосбережения, которые позволяют устройствам работать в разных режимах в зависимости от их функциональности и требований.

5. Распространенность: I2C является широко используемым интерфейсом и поддерживается большинством микроконтроллеров и периферийных устройств. Это обеспечивает совместимость и удобство при интеграции различных устройств.

### 1.3.2 Обзор интерфейса SPI

SPI [9] (Serial Peripheral Interface) – это последовательный интерфейс, который широко используется для связи между микроконтроллерами и периферийными устройствами.

Основные характеристики интерфейса SPI:

1. Высокая скорость передачи данных: SPI предоставляет высокоскоростную передачу данных, что делает его идеальным для приложений, где требуется быстрая обработка данных. Скорость передачи может достигать нескольких мегабит в секунду, и в отличие от других интерфейсов, SPI не имеет ограничений на скорость передачи.

2. Параллельная связь: SPI использует отдельные линии для каждого устройства, включая линии тактирования (SCLK), передачи данных (MOSI) и приема данных (MISO). Это позволяет более надежное и параллельное взаимодействие с несколькими устройствами одновременно.

3. Гибкость: SPI предоставляет большую гибкость в настройке и передаче данных. Он поддерживает разные режимы передачи данных, такие как полный дуплекс, полудуплекс и однонаправленная передача данных. Также возможно подключение нескольких устройств на одной шине SPI.

4. Простота использования: В отличие от других интерфейсов, SPI имеет простой и понятный протокол передачи данных. Он основан на мастер-слейв архитектуре, где микроконтроллер выступает в роли мастера, а периферийные устройства – в роли слейвов. Это делает его легким в использовании и программировании.

5. Распространенность: SPI является одним из самых распространенных последовательных интерфейсов и поддерживается большинством микроконтроллеров и периферийных устройств. Это обеспечивает совместимость и удобство при интеграции различных устройств.

Интерфейс SPI является мощным и гибким способом связи между микроконтроллерами и периферийными устройствами. Его высокая скорость передачи данных, параллельная связь и гибкость делают его предпочтительным выбором для многих приложений, таких как сенсорные экраны, датчики, флэш-память и другие устройства.

## 1.4 Обзор библиотеки OpenCV

OpenCV [10] (Open Source Computer Vision Library) – это библиотека с открытым исходным кодом, которая предоставляет широкий спектр функций для обработки изображений и компьютерного зрения. Она была разработана с целью помочь разработчикам создавать приложения, связанные с компьютерным зрением, такие как распознавание объектов, трекинг, сегментация и многое другое.

Основные возможности библиотеки OpenCV:

1. Обработка изображений и видео: OpenCV предоставляет набор функций для чтения, записи, обработки и анализа изображений и видео. Она поддерживает различные форматы файлов и позволяет применять различные операции обработки, такие как фильтрация, морфологические операции, изменение размера и многое другое.
2. Распознавание объектов: OpenCV предлагает алгоритмы компьютерного зрения, которые позволяют распознавать и классифицировать объекты на изображениях. Она поддерживает методы машинного обучения, такие как метод опорных векторов (SVM), а также алгоритмы, основанные на особенностях (features), например, гистограммы направленных градиентов (HOG) и каскады Хаара.
3. Трекинг объектов: OpenCV предоставляет алгоритмы для отслеживания движущихся объектов на видео. Они позволяют определить позицию и перемещение объекта в каждом кадре видео. Это полезно для различных приложений, включая системы слежения за объектами и анализа движения.
4. Сегментация изображений: OpenCV позволяет разделять изображения на отдельные объекты или регионы на основе их свойств и характеристик. Это полезно для задач сегментации, таких как выделение объектов на фоне, разделение изображений на различные регионы и другие.
5. Машинное обучение: OpenCV включает в себя поддержку множества алгоритмов машинного обучения, таких как классификация, кластеризация, регрессия и т.д. Это позволяет разработчикам создавать модели машинного обучения для различных задач компьютерного зрения.

Распознавание объектов является одним из ключевых применений OpenCV. Библиотека предоставляет мощные инструменты и алгоритмы для обнаружения и классификации объектов на изображениях. Это может быть полезным для различных задач, таких как распознавание лиц, детектирование автомобилей, определение жестов и многое другое. OpenCV предоставляет гибкие возможности для создания собственных моделей и обучения их на основе размеченных данных.

В целом, OpenCV является мощным инструментом для разработки приложений, связанных с компьютерным зрением. Она предоставляет широкий спектр функций и алгоритмов, которые могут быть использованы для различных задач обработки изображений и распознавания объектов.

# 2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

Для того, чтобы составить структуру разрабатываемой системы, необходимо выделить функции, которые будет выполнять система, затем определить компоненты и связь между ними исходя из данных функций.

В рамках данного дипломного проекта необходимо разработать устройство управление, которое подключается к игровой платформе по радиоканалу и позволяет управлять моделью игрока, а также программу под операционную систему Android, которая кроме того, что позволяет подключаться и управлять моделью игрока, но ещё и позволяет наблюдать за моделью игрока от первого лица, а также распознавать положение мяча на платформе. Структурная схема устройства представлена на чертеже ГУИР.400201.025 Э1. Структурная схема программной части представлена на чертеже ГУИР.400201.025 C1.

## 2.1 Описание основных блоков устройства управления

В результате разделения устройства управления были определены следующие блоки:

* блок управления;
* блок элементов управления перемещением;
* блок элементов управления отталкиванием мяча;
* блок приёма и передачи радиосигнала;
* блок отображения информации;
* блок заряда элемента питания;
* блок звуковой индикации;
* блок питания.

### 2.1.1 Блок управления

Блок управления играет важную роль в функционировании устройства управления. Его основная задача - обеспечить связь и взаимодействие между всеми блоками.

Блок управления отвечает за управление игровым процессом и взаимодействие с игроками. Он считывает данные, полученные c элементов управления, анализирует и обрабатывает полученные данные и отправляет соответствующие сигналы блоку приёма и передачи радиосигнала. Также блок управления отвечает за считывание информации о состоянии подключения устройства управления к платформе и о состоянии зарядки и разрядки аккумулятора.

### 2.1.2 Блок элементов управления перемещением

Блок элементов управления перемещением представляет собой органы управления, которые позволяют игрокам контролировать движение модели игрока.

Органы управления включают в себя различные кнопки, переключатели и джойстики, которые позволяют игрокам передвигать модель игрока по игровому полю. С помощью кнопок или джойстика игрок может управлять направлением движения модели игрока влево или вправо, а также активировать отталкивающий механизм. Эти органы управления позволяют игрокам маневрировать и реагировать на движение шайбы, обеспечивая более точное и эффективное управление игровым процессом.

### 2.1.3 Блок элементов управления отталкиванием мяча

Блок элементов управления отталкиванием мяча представляет собой органы управления, которые позволяют игрокам подавать сигнал отталкивания мяча.

Органы управления включают в себя кнопки, которые позволяют игрокам подавать сигнал отталкивания мяча. При подаче сигнала отталкивания мяча на игровую платформу активируется механизм отталкивания мяча, который расположен на модели игрока.

### 2.1.4 Блок приёма и передачи радиосигнала

Блок приёма и передачи радиосигнала играет важную роль в передаче данных от блока управления на модель игрока.

После обработки данных блоком управления, эти данные необходимо передать на модель игрока. Для этого используется блок приёма и передачи радиосигнала. Этот блок обеспечивает беспроводную связь между блоком управления и моделью игрока, что позволяет передавать данные без необходимости физического подключения.

Блок управления передает данные на блок приёма и передачи радиосигнала, который затем кодирует и модулирует информацию для передачи по радиоканалу. После этого, блок приёма и передачи радиосигнала отправляет данные на модель игрока.

Модель игрока, в свою очередь, получает данные от блока приёма и передачи радиосигнала и использует их для управления игровым процессом.

### 2.1.5 Блок отображения информации

Блок отображения выводит визуальное представление заряда аккумулятора, информацию о состоянии зарядки и разрядки, а также состояние подключения к игровой модели игрока. Этот блок получает питание от схемы и отображает соответствующую информацию в удобном визуальном формате для пользователя.

Основной функцией этого блока является отображение уровня заряда аккумулятора, который питает устройство. Это может быть представлено в виде графической шкалы, числового значения или иконки на дисплее. Пользователь может видеть текущий уровень заряда и решать, требуется ли зарядка аккумулятора.

Кроме того, этот блок также выводит информацию о состоянии зарядки и разрядки аккумулятора. Например, он может указывать, аккумулятор заряжается, разряжается или находится в режиме ожидания. Это позволяет пользователю контролировать процесс зарядки и разрядки аккумулятора и принимать необходимые меры в зависимости от текущего состояния.

Кроме того, этот блок также может отображать информацию о состоянии подключения к игровой модели игрока. Например, он может указывать, установлено ли подключение или происходит ли передача данных. Это позволяет игроку контролировать соединение и убедиться, что устройство готово к игре.

### 2.1.6 Блок заряда элемента питания

Основная функция блока заряда элемента питания – контроль и регулирование процесса зарядки аккумулятора, чтобы обеспечить оптимальное и безопасное заполнение его энергией.

Блок заряда элемента питания обеспечивает надежную и безопасную зарядку аккумулятора, контролируя напряжение, ток и другие параметры. Он также может предоставлять информацию о состоянии зарядки с помощью индикаторов. Важно выбирать надежный и эффективный блок управления зарядкой, чтобы обеспечить оптимальную работу аккумулятора и продлить его срок службы.

### 2.1.8 Блок звуковой индикации

Блок звуковой индикации предназначен для оповещения пользователя о состоянии подключения модели игрока к игровой платформе. Он обеспечивает аудиоинформацию о процессе подключения устройства управления к игровой платформе.

Блок звуковой индикации предоставляет различные звуковые сигналы, такие как мелодии или гудки, которые информируют пользователя о различных этапах подключения модели игрока. Звуковой сигнал указывает на успешное подключение, неудачную попытку или требовать внимания пользователя для выполнения определенных действий. Важно отметить, что блок звуковой индикации должен быть проектирован и настроен таким образом, чтобы звуковые сигналы были понятными и различимыми для пользователя.

### 2.1.7 Блок питания

Блок питания устройства управления, который представляет собой аккумулятор, обеспечивает энергией все необходимые компоненты и функции устройства. Он является источником питания, который может быть переносным, и обеспечивать независимую работу устройства от внешней электрической сети.

## 2.2 Описание основных блоков программы

В результате разделения программы были определены следующие блоки:

* блок пользовательского интерфейса;
* блок подключения к модели игрока;
* блок обработки пользовательского ввода;
* блок отправки данных;
* блок захвата видеоданных с модели игрока;
* блок обработки видеоданных;
* блок получения данных;
* блок обработки полученных данных.

### 2.2.1 Блок пользовательского интерфейса

Блок пользовательского интерфейса отвечает за взаимодействие пользователя с приложением. Он включает в себя различные элементы, такие как кнопки, ползунки, текстовые поля и т.д., которые позволяют пользователю управлять игроком и взаимодействовать с приложением.

В этом блоке реализуются следующие функции:

* кнопки или для управления движением игрока влево и вправо;
* кнопка для активации и деактивации отталкивающего механизма игрока;
* отображение изображения с модели игрока от первого лица и распознанного положения мяча на игровом поле;
* отображение состояния подключения к модели игрока;
* первичная настройка подключения к модели игрока.

Блок пользовательского интерфейса является важной частью программы, так как он предоставляет пользователю удобный и интуитивно понятный способ управления моделью игрока и взаимодействия с приложением.

### 2.2.2 Блок подключения к модели игрока

Блок подключения к модели игрока отвечает за установление связи между приложением и моделью игрока симулятора компьютерной игры Пинг-Понг. В этом блоке реализуются следующие функции:

* поиск и обнаружение доступных моделей игроков симулятора Пинг-Понг;
* установление соединения с выбранной моделью игрока;
* обработка ошибок и уведомлений. Если происходят ошибки в процессе подключения или обмена данными, блок подключения должен быть способен обработать эти ситуации, отображать соответствующие уведомления и предпринимать соответствующие действия для восстановления соединения или устранения проблемы.

### 2.2.3 Блок обработки пользовательского ввода

Блок обработки пользовательского ввода отвечает за получение и обработку действий и команд, вводимых пользователем через пользовательский интерфейс приложения. В этом блоке реализуются следующие функции:

* получение пользовательского ввода;
* интерпретация пользовательского ввода;
* валидация пользовательского ввода;
* генерация команд и данных для отправки в блок отправки данных;
* обратная связь с пользователем.

### 2.2.4 Блок отправки данных

Блок отправки данных отвечает за передачу команд и данных, сгенерированных блоком обработки пользовательского ввода, на модель игрока. В этом блоке реализуются следующие функции:

* установление соединения с моделью игрока;
* передача команд и данных;
* обработка ошибок и уведомлений;
* подтверждение доставки данных;
* завершение соединения.

### 2.2.5 Блок захвата видеоданных с модели игрока

Блок захвата видеоданных с модели игрока отвечает за получение видеопотока с камеры, установленной на модели игрока, и обработку этого видеопотока для распознавания объектов. В этом блоке реализуются следующие функции:

* захват видеопотока;
* передача видеопотока в блок обработки видеоданных;
* обработка ошибок и уведомлений при захвате видеопотока.

### 2.2.6 Блок обработки видеоданных

Блок обработки видеоданных отвечает за обработку видеопотока, полученного из блока захвата видеоданных. В этом блоке происходит распознавание и отслеживание объектов в видеопотоке, чтобы определить их положение и движение. Далее результаты обработки визуализируются, чтобы пользователи могли наглядно увидеть и проанализировать данные. Этот блок играет ключевую роль в обработке видеоданных и предоставляет информацию, необходимую для принятия решений и эффективного управления моделью игрока.

### 2.2.7 Блок получения данных

Блок получения данных представляет собой компонент, который обеспечивает прием информации о текущем игровом процессе с игровой платформы через беспроводное соединение. Данные упакованы в пакеты, которые содержат информацию о текущем игровом процессе, например, о забитых голах, количестве жизней игроков и других игровых параметрах. Данный блок играет важную роль в передаче данных, полученных от игровой платформы.

### 2.2.8 Блок обработки полученных данных

Блок обработки полученных данных является компонентом, который выполняет обработку информации, полученной от блока получения данных. Блок обработки полученных данных выполняет операции по форматированию, фильтрации, сортировке и адаптации информации для понятного и комфортного восприятия игровых данных игроком. Он преобразует данные в удобный формат, чтобы далее отправить эти данные в блок пользовательского интерфейса для наилучшего отображения.

# 3 ФУНКЦИОНАЛНОЕ ПРОЕКТИРОВАНИЕ

В разделе структурного проектирования была спроектирована структура разрабатываемого комплекса и дано описание основных блоков системы. В этом разделе будут описаны ранее представлены блоки с точки зрения разработки функций, которые реализуются в данном дипломном проекте. Функциональная схема устройства представлена на чертеже ГУИР.400201.025 Э2.

## 3.1 Блоки устройства управления

Для устройства управления была определена следующая структура:

* блок управления;
* блок элементов управления перемещением;
* блок элементов управления отталкиванием мяча;
* блок приёма и передачи радиосигнала;
* блок отображения информации;
* блок заряда элемента питания;
* блок звуковой индикации;
* блок питания.

### 3.1.1 Блок управления

Блок управления является важной составляющей устройства управления, так как его основная функция заключается в обеспечении связи и взаимодействия между различными блоками системы. Он играет роль посредника, передающего команды и получающего обратную связь от других блоков.

Для реализации блока управления может быть выбран микропроцессор или микроконтроллер. Сравнительная характеристика микроконтроллера и микропроцессора представлена в таблице 3.1.

Таблица 3.1 – Сравнительная характеристика микропроцессора и микроконтроллера

|  |  |  |
| --- | --- | --- |
| Параметр | Микроконтроллер | Микропроцессор |
| 1 | 2 | 3 |
| Встроенная память | Имеет встроенную оперативную и постоянную память | Не имеет встроенной памяти |
| Периферийные устройства | Имеет встроенные периферийные устройства (интерфейсы ввода-вывода, таймеры и т.д.) | Требует внешних устройств для подключения и функционирования (интерфейсы, таймеры т.д.) |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Энергопотребление | Низкое | Высокое |
| Стоимость | Более доступная | Обычно более высокая |
| Размер | Имеет маленькие размеры | Имеет более крупные размеры |

В целом, микроконтроллер и микропроцессор являются двумя различными типами интегральных схем, используемых для обработки данных и управления различными системами. Микроконтроллеры обладают встроенными компонентами и периферийными устройствами, что делает их более удобными для реализации систем управления. Они обеспечивают низкое энергопотребление, более доступную стоимость и специализированы на управлении и обработке данных во встраиваемых системах. Микропроцессоры, с другой стороны, предназначены для обработки общего назначения и требуют внешних устройств для функционирования.

Также для реализации блока управления необходимы интерфейсы поскольку они обеспечивают связь и взаимодействие с другими устройствами или системами. Они обеспечивают передачу данных в обоих направлениях, контроль и управление подключенными устройствами.

Оперативная память также необходима для реализации блока управления, поскольку обеспечивает быстрый доступ к данным и временное хранение информации, необходимой для работы системы. Она используется для временного хранения промежуточных результатов вычислений и других временных данных.

Для хранения всех исполняемых инструкций разработанного программного кода и неизменяемых данных необходима постоянная память. Постоянная память, такая как флэш-память или энергонезависимая память, используется для хранения программного кода и данных, которые должны сохраняться даже при отключении питания. Она обеспечивает сохранение и доступ к важным данным и программам даже после перезагрузки системы.

Исходя из вышеперечисленных требований, был выбран вариант использования микроконтроллера. Микроконтроллер объединяет в себе все необходимые компоненты для обеспечения связи и обработки данных в системе управления. Он обладает встроенным центральным процессором, оперативной и постоянной памятью, а также различными интерфейсами, что делает его идеальным выбором для реализации блока управления.

### 3.1.2 Блок элементов управления перемещением

Блок элементов управления перемещением модели игрока состоит из компонентов, которые предназначены для управления движением моделью игрока. Игроку будет представлено два варианта набора элементов управления моделью игрока:

* набор, состоящий из тактовых кнопок;
* набор, состоящий из одного джойстика.

Управление перемещением модели игрока осуществляется только вправо и влево, поэтому набор из тактовых кнопок состоит только из двух тактовых кнопок, а набор из джойстика состоит из одного одноосевого джойстика.

Отличительной особенностью джойстика от набора из тактовых кнопок заключается в более точном управлении. Управление с помощью джойстика позволяет более точное и плавное управление моделью игрока. Джойстик обладает аналоговыми стиками, которые могут реагировать на различные уровни нажатия и движения, что позволяет более точно контролировать движение модели. В то время как управление с помощью двух кнопок обычно имеет только два состояния – нажатие или отпускание, что может ограничить точность и плавность управления.

Кнопки подключены к дискретным входам микроконтроллера и используют интерфейс GPIO. Джойстик подключён к одному аналоговому входу микроконтроллера и использует интерфейс GPIO.

### 3.1.3 Блок элементов управления отталкиванием мяча

Блок элементов управления отталкиванием мяча представляет собой тактовую кнопку, которая отвечает за активацию механизма отталкивания мяча, расположенного на модели игрока. Тактовая кнопка подключается к дискретным входам микроконтроллера и использует интерфейс GPIO.

### 3.1.4 Блок приёма и передачи радиосигнала

Блок приёма и передачи радиосигнала выполняет функцию беспроводной связи между устройством управления игрока и игровой платформой. Он позволяет передавать сигналы о перемещении модели игрока и активации механизма отталкивания без использования проводов.

Выбор беспроводного способа передачи сигнала имеет несколько причин. Во-первых, он обеспечивает более гибкую и удобную связь между устройством управления игрока и игровой платформой. Благодаря отсутствию проводов не возникает ограничений в перемещении игрока и его устройства управления.

Во-вторых, беспроводной способ передачи сигнала позволяет избежать запутывания проводов и упрощает настройку и установку игровой системы. Необходимость прокладывать провода между устройством управления и платформой может создавать проблемы в организации игрового пространства и приводить к неправильной установке и пользованию системой.

Кроме того, беспроводной способ позволяет игроку свободно перемещаться во время игры без ограничений, что повышает комфорт и реалистичность игрового процесса.

Таким образом, выбор беспроводного способа передачи сигнала обусловлен удобством использования, отсутствием ограничений в перемещении и установке, а также повышением комфорта игрока во время игры.

Беспроводной способ передачи сигнала может быть реализован по Bluetooth или по радиоканалу. Оба способа передачи сигнала имеют свои преимущества и недостатки.

В таблице 3.2 представлена сравнительная характеристика передачи по Bluetooth и передачи по радиоканалу.

Таблица 3.2 – Сравнительная характеристика передачи по Bluetooth и передачи по радиоканалу

|  |  |  |
| --- | --- | --- |
| Параметры сравнения | Передача по Bluetooth | Передача по радиоканалу |
| Дальность передачи сигнала | Ограничена (~10 м) | Большая (~100 м и более) |
| Пропускная способность | Ограничена (обычно до 3 Мбит/с) | Высокая (до 54 Мбит/с и выше) |
| Устойчивость к помехам | Средняя | Высокая |
| Задержка передачи сигнала | Низкая (обычно до 100 мс) | Низкая (обычно до 50 мс) |
| Потребление энергии | Низкое | Среднее-высокое |
| Совместимость с другими устройствами | Широкая | Широкая |
| Сложность настройки и установки | Низкая | Низкая |
| Стоимость оборудования | Низкая | Низкая-средняя |
| Гибкость и мобильность | Высокая | Высокая |

Радиоканал выбран в качестве способа передачи сигнала по нескольким причинам. Во-первых, передача по радиоканалу обладает более большой дальностью, что позволяет игроку свободно перемещаться на большом расстоянии от игровой платформы. В то же время, передача по Bluetooth ограничена дальностью до 10 метров.

Во-вторых, радиоканал обеспечивает высокую пропускную способность, что позволяет передавать больший объем данных за более короткое время. Bluetooth имеет более ограниченную пропускную способность, обычно до 3 Мбит/с, в то время как радиоканал может достигать скоростей до 54 Мбит/с и выше.

Третий фактор – устойчивость к помехам. Передача по радиоканалу обладает более высокой устойчивостью к помехам, так как частотный диапазон радиосигнала менее загружен и менее подвержен внешним вмешательствам, по сравнению с Bluetooth.

Задержка передачи сигнала также является важным параметром. Оба способа обладают низкой задержкой, но передача по радиоканалу обычно имеет еще немного меньшую задержку, до 50 мс, по сравнению с Bluetooth, до 100 мс.

Оба способа передачи сигнала обладают широкой совместимостью с другими устройствами и имеют низкую сложность настройки и установки.

Стоимость оборудования для обоих способов передачи сигнала обычно низкая, однако стоимость оборудования для передачи по радиоканалу может быть немного выше из-за более высокой пропускной способности.

И, наконец, гибкость и мобильность – оба способа передачи обеспечивают высокую гибкость и мобильность, позволяя игроку свободно перемещаться и управлять игровой платформой.

В целом, передача по радиоканалу была выбрана в данном случае, так как обладает большей дальностью, высокой пропускной способностью, устойчивостью к помехам и низкой задержкой передачи сигнала. Это позволяет обеспечить комфортную и надежную беспроводную связь между устройством управления игрока и игровой платформой.

Для подключения блока приёма и передачи радиосигнала к микроконтроллеру используется интерфейс SPI. Входы блока приёма и передачи радиосигнала подключаются к выходам микроконтроллера, которые поддерживают SPI интерфейс, такие как MOSI, MISO, SCK и CSN.

Интерфейс SPI выбран для подключения блока приёма и передачи радиосигнала к микроконтроллеру по следующим причинам:

1. Полный дуплекс: SPI поддерживает передачу данных в режиме полного дуплекса, что позволяет одновременно передавать и принимать данные1.
2. Высокая скорость передачи данных: SPI обеспечивает высокоскоростное сопряжение микроконтроллеров и периферии, что является важным фактором для обеспечения быстрой и эффективной связи.
3. Синхронизация: В отличие от стандартного последовательного порта, SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством.

### 3.1.5 Блок отображения информации

Блок отображения информации в устройстве управления отвечает за отображение различных данных, включая информацию о зарядке и разрядке, а также о состоянии подключения устройства управления к игровой платформе. Он представляет собой дисплей, который интегрируется в устройство управления.

Для выполнения функционала отображения информации о зарядке и разрядке, а также о состоянии подключения устройства управления к игровой платформе, дисплей должен быть компактным, но достаточно информативным. Он должен быть способен отображать текстовую и графическую информацию на достаточно маленькой площади.

Сравнительная характеристика типов дисплеев представлена в таблице 3.3.

Таблица 3.3 – Сравнительная характеристика типов дисплеев

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры | OLED | LCD | LED |
| Яркость | Высокая | Средняя | Средняя |
| Контрастность | Высокая | Низкая | Средняя |
| Угол обзора | Широкий | Узкий | Средний |
| Цветопередача | Высокая | Средняя | Средняя |
| Энергоэффективность | Высокая | Низкая | Высокая |

Исходя из таблицы 3.3 OLED-дисплеи обладают высокой яркостью, высоким контрастом и широким углом обзора, также OLED-дисплеи позволяют отображать текст и графику на черном фоне, что обеспечивает хорошую читаемость и эстетический вид, поэтому в разрабатываемом устройстве будет использоваться OLED-дисплей.

Для подключения блока отображения информации к микроконтроллеру используется интерфейс I2C. Причины использования интерфейса I2C:

1. Минимальное количество проводов: Интерфейс I2C использует всего два провода – SDA и SCL – для передачи данных и управления. Это делает подключение дисплея к микроконтроллеру более простым и экономически выгодным, поскольку требуется меньше пинов на микроконтроллере.
2. Мультиплексирование: I2C поддерживает мультиплексирование, что позволяет подключать несколько устройств к одним SDA и SCL линиям, используя уникальные адреса для каждого устройства. Это означает, что вы можете подключить несколько дисплеев к одному микроконтроллеру, расширяя функциональность системы, необходимую для отображения информации на различных местах.
3. Простота программирования: Интерфейс I2C имеет широкую поддержку во многих микроконтроллерах и дисплеях, что упрощает программирование и интеграцию. Существуют библиотеки и драйверы, которые облегчают разработку кода для управления дисплеем по протоколу I2C.
4. Гибкость и расширяемость: I2C позволяет подключать различные типы дисплеев, включая OLED, LCD и TFT, к одному микроконтроллеру. Это дает возможность выбирать наиболее подходящий тип дисплея.

Таким образом, выбор дисплея с использованием OLED-технологии и интерфейса подключения I2C обеспечивает компактность, информативность, простоту подключения и масштабируемость для блока отображения информации в устройстве управления.

### 3.1.6 Блок заряда элемента питания

Блок заряда элемента питания представляет собой зарядное устройство, которое используется для зарядки элемента питания в устройстве контроля их разрядки. Он обычно состоит из нескольких компонентов, которые выполняют различные функции.

Функционал блока заряда элемента питания включает в себя:

1. Регулирование тока зарядки. Блок заряда контролирует ток, который поступает в аккумулятор, чтобы обеспечить безопасную и эффективную зарядку. Он регулирует ток в зависимости от состояния аккумулятора, чтобы предотвратить его перегрев или повреждение.
2. Контроль напряжения. Блок заряда также контролирует напряжение зарядки, чтобы обеспечить оптимальные условия для аккумулятора.
3. Защита от короткого замыкания: Блок заряда оснащен механизмами защиты от короткого замыкания. Это позволяет предотвратить повреждение блока заряда и аккумулятора в случае неправильного подключения или короткого замыкания.
4. Защита от переразряда: Блок заряда обеспечивает защиту от переразряда аккумулятора. Это достигается путем контроля напряжения и отключения аккумулятора, когда напряжение аккумулятора достигает определенного минимального уровня. Это защищает аккумулятор от глубокого разряда, что может повредить его и сократить срок службы.
5. Индикация зарядки: Блоки заряда элементов питания оснащен индикаторами, которые показывают текущий статус зарядки. Для этого используется светодиоды, которые в зависимости от статуса зарядки загораются определённым цветом.

Блок заряда элемента подключён к блоку питанию для контроля заряда и разряда и к остальным блокам для подачи питания.

### 3.1.7 Блок звуковой индикации

Блок звуковой индикации выполняет функцию оповещения о состоянии подключения устройства управления к игровой платформе. Блок воспроизводит звуковые сигналы, чтобы информировать игрока о успешном подключении или разъединении устройства. Для реализации блока звуковой индикации необходимо устройство воспроизведения звукового сигнала.

Сравнительная характеристика устройств воспроизведения звукового сигнала представлена в таблице 3.4.

Таблица 3.4 – Сравнительная характеристика устройств воспроизведения звукового сигнала

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Пьезодинамик | Динамик на основе электродинамического преобразователя | Пьезокерамический преобразователь |
| 1 | 2 | 3 | 4 |
| 1 Объем и компактность | Маленький | Большой | Маленький |
| 2 Энергопотребление | Низкое | Высокое | Низкое |

Продолжение таблицы 3.4

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 3 Громкость | Достаточно громкий | Высокая | Средняя |
| 4 Частотная характеристика | Ограниченная | Широкий диапазон | Ограниченная |
| 5 Стоимость | Низкая | Различается в зависимости от модели | Средняя |
| 6 Качество звука | Хорошее | Очень хорошее | Хорошее |
| 7 Размер и вес | Маленький | Большой | Маленький |
| 8 Устойчивость к воздействию | Хорошая | Низкая | Хорошая |
| 9 Мощность | Низкая | Высокая | Средняя |

Из таблицы 3.4 видно, что пьезодинамик обладает следующими преимуществами: маленький объем и компактность, низкое энергопотребление, достаточно громкий звук, низкая стоимость, хорошее качество звука и хорошая устойчивость к воздействию. Исходя из требований для блока звуковой индикации, пьезодинамик является оптимальным выбором, так как он обеспечивает достаточно громкий и четкий звук при небольшом объеме и энергопотреблении, что важно для интеграции в устройство управления игровой платформой.

Также звуковые сигналы должны быть легко распознаваемыми и ассоциироваться с определенным состоянием подключения, чтобы игрок мог мгновенно определить, что происходит с его устройством управления.

### 3.1.8 Блок питания

Блок питания представляет собой устройство, предназначенное для обеспечения электропитания других устройств. Варианты подачи питания могут быть представлены в двух основных видах:

1. Внешний источник питания: Внешний источник питания представляет собой отдельное устройство, которое подключается к устройству с помощью кабеля или разъема. Это может быть зарядное устройство, адаптер или другое устройство, подключаемое к розетке или другому источнику электропитания. Внешний источник питания передает электричество устройству через кабель или разъем.
2. Внутренний источник питания: Внутренний источник питания интегрирован непосредственно в устройство и обеспечивает электропитание его компонентов. Это может быть встроенный аккумулятор, батарея или другой источник питания, размещенный внутри корпуса устройства.

Было принято решение использовать внутренний источник питания в виде аккумулятора, так как внутренний источник питания позволяет устройству быть полностью более портативным и мобильным, а также позволяет быть независимым от внешних источников питания, таких как розетка или батареи.

Внутренние источники питания включают в себя аккумуляторы и батарейки.

Сравнительная характеристика внутренних источников питания представлена в таблице 3.5.

Таблица 3.5 – Сравнительная характеристика внутренних источников питания

|  |  |  |
| --- | --- | --- |
| Параметры сравнения | Аккумуляторы | Батарейки |
| Ёмкость | Высокая | Низкая |
| Перезаряжаемость | Да | Нет |
| Срок службы | Длительный | Ограниченный |
| Стоимость | Средняя | Низкая |
| Экологическая дружественность | Хорошая | Средняя |
| Напряжение | Стабильное | Постепенно падает |
| Возможность использования в низких температурах | Да | Нет |

Аккумуляторы были выбраны как внутренний источник питания по ряду причин. Во-первых, они обладают высокой емкостью, что позволяет им хранить большое количество энергии и обеспечивать продолжительное время работы устройства без необходимости замены источника питания. Батарейки, в свою очередь, имеют ограниченную емкость и требуют периодической замены.

Во-вторых, аккумуляторы являются перезаряжаемыми, что позволяет повторно использовать их после разрядки. Это экономически выгодно и удобно для пользователя, так как нет необходимости постоянно покупать новые батарейки.

Срок службы аккумуляторов также длительный. При правильном использовании и зарядке аккумуляторы могут прослужить длительное время, в то время как батарейки имеют ограниченный срок службы и требуют замены после разрядки.

Стоимость аккумуляторов обычно средняя, так как они требуют начальных затрат на приобретение зарядного устройства и самих аккумуляторов. Батарейки, с другой стороны, имеют низкую стоимость, но требуют периодической замены, что в долгосрочной перспективе может быть более затратным.

Напряжение аккумуляторов остается стабильным на протяжении их работы, что обеспечивает стабильное питание устройства. В то время как напряжение батареек постепенно падает по мере их разрядки.

Существует несколько видов аккумуляторов, включая никель-кадмиевые (NiCd), никель-металл-гидридные (NiMH), литий-ионные (Li-ion) и литий-полимерные (Li-po) аккумуляторы.

Сравнительная характеристика типов аккумуляторов представлена в таблице 3.6.

Таблица 3.6 – Сравнительная характеристика типов аккумуляторов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметры сравнения | NiCd | NiMH | Li-ion | Li-po |
| Ёмкость | Средняя | Высокая | Высокая | Высокая |
| Саморазрядка | Высокая | Средняя | Низкая | Низкая |
| Жизненный цикл | Средний | Средний | Высокая | Высокая |
| Вес | Тяжёлый | Средний | Лёгкий | Лёгкий |
| Стоимость | Низкая | Средняя | Высокая | Высокая |

Был выбран литий-полимерный аккумулятор для устройства управления по нескольким причинам. Во-первых, литий-полимерные аккумуляторы обладают высокой емкостью, что позволяет устройству функционировать длительное время без перезарядки.

Во-вторых, они имеют низкую саморазрядку, что обеспечивает сохранение энергии в аккумуляторе даже при длительном хранении. Кроме того, литий-полимерные аккумуляторы легкие и компактные, что делает их идеальным выбором для портативных устройств, таких как устройство управления игроками.

Несмотря на то что литий-ионные и литий-полимерные аккумуляторы обладают одинаковыми характеристиками, литий-полимерные аккумуляторы обладают большей гибкостью в оформлении и могут быть произведены в различных формах и размерах. Это позволяет лучше интегрировать аккумулятор в дизайн устройства управления и сделать его более компактным и эргономичным.

## 3.2 Описание функций прошивки устройства управления

Прошивка устройства управления включает в себя следующие функции:

* инициализация подключения к игровой платформе по радиоканалу;
* обработка сигнала управления перемещением модели игрока;
* обработка сигнала отталкивания мяча;
* обработка сигнала восстановление жизней модели игрока;
* передача сигналов по радиоканалу;
* отображение информации о количестве жизней у модели игрока и о проценте заряда аккумулятора.

### 3.2.1 Реализация функции инициализации подключения к игровой платформе по радиоканалу

Реализация функции инициализации подключения к игровой платформе по радиоканалу может включать следующие шаги:

1. Настройка радиочастотного модуля для работы с требуемыми параметрами, такими как активация модуля, режим подтверждения приёма, время между попытками повторного соединения, количество попыток повторного соединения, разрешение на отсылку данных в ответ на входящий сигнал, размер пакета, выбор канала, уровень мощности, скорость обмена.
2. Включение радиочастотного модуля и попытка установки соединения с игровой платформой. При успешном подключении к игровой платформе происходит переход к основному циклу программы.

Важно уточнить то, что параметры радиочастотного модуля должны быть одинаковы у приёмка и у передатчика, иначе данные передаваться не будут.

### 3.2.2 Реализация функции обработки сигнала управления перемещением модели игрока

Реализация функции обработки сигнала управления перемещением модели игрока представляет собой получение сигнала с элементов управления и интерпретация этого сигнала.

Для получения сигналов с элементов управления используется аналоговое чтение, в случае если элементом управления является джойстик, и цифровое чтение, в случае если элементом управления являются тактовые кнопки, сигналов с соответствующих пинов.

При интерпретации по значению, полученного с джойстика, определяется скорость и направление движения модели игрока. Так как кнопка имеет только два состояния, то было принято решение, что при нажатой кнопки значение скорости модели игрока будет максимальной, при отжатой модель игрока не будет двигаться.

### 3.2.3 Реализация функции обработки сигнала отталкивания мяча

Реализация функции обработки сигнала отталкивания мяча представляет собой считывание состояния кнопки, отвечающую за подачу сигнала отталкивания мяча. При нажатой кнопке подаётся соответствующий сигнал отталкивания мяча, при отжатой кнопке ничего не происходит.

### 3.2.4 Реализация функции обработки сигнала восстановления жизней модели игрока

Реализация функции обработки сигнала восстановления жизней модели игрока представляет собой считывание состояния кнопок, отвечающие за управление перемещением модели игрока. При нажатии и удержании в течение пяти секунд двух тактовых кнопок, которые отвечают за управление перемещением модели игрока, подаётся сигнал о восстановлении всех жизней игрока. Активация функции восстановления жизней игрока происходит таким образом, чтобы избежать случайного нажатия в процессе игры.

### 3.2.4 Реализация функции передачи сигналов по радиоканалу

Реализация функции передачи сигналов по радиоканалу представляет собой кодирование полученных значений с элементов управления и отправки их на игровую платформу по радиоканалу.

Кодирование будет осуществляется путём инициализации специальной структуры, которая содержит значения направления и скорости перемещения модели игрока, а также значение активации функции отталкивания мяча. Далее после кодирования готовое сообщение, состоящее из специальной структуры, передаётся радиочастотному модулю для дальнейшей отправки на игровую платформу по радиоканалу.

### 3.2.5 Реализация функции отображения информации о количестве жизней у модели игрока и о проценте заряда аккумулятора

Реализация функции отображения информации о количестве жизней у модели игрока представляет собой отображение на дисплее текстовой информации в виде числа количества жизней игрока, полученного с игровой платформы. Также при утрате игроком всех жизни будет предложено восстановить жизни для дальнейшего продолжения игрового процесса, в противной случае продолжить игровой процесс будет невозможно ввиду отсутствия жизней у игрока.

Реализация функции отображения информации о статусе аккумулятора представляет собой отображение на дисплее текстовой информации в виде числа процентов оставшегося заряда аккумулятора.

## 3.3 Описание функционала программной части

Одна из основных функций программной части является подключение к модели игрока. Программа проверяет включён Wi-Fi. Если включён сканирует Wi-Fi сети на наличие сетей моделей игроков, подключается к одной из них. Следующей одной из главных функций является захват изображения с камеры модели игрока и применение к изображению ряд алгоритмов для обнаружения мяча. Также основной функцией является обработка пользовательского ввода и отправка сигналов управления на модель игрока. Часть функций программной части представлена на диаграмме последовательности, которая представлена на чертеже ГУИР.400201.025 PP.1.

## 3.4 Описание диаграммы классов

Программная часть проекта включает в себя следующие классы:

* класс VideoCamCapture;
* класс DetectBall;
* класс GameInfo;
* класс ReceiverData;
* класс ManagePlayer;
* класс SenderSignals;
* класс Socket;
* класс ConnectionToPlayer;
* класс WiFiApi;
* класс QQuickImageProvider;
* класс QObject;
* класс BroadcastReceiver.

Диаграмма классов программной части представлена в приложении А.

### 3.4.1 Класс VideoCamCapture

Класс VideoCamCapture предназначен для захвата видео с камеры модели игрока.

Поля класса:

* timeUpdateFrame – поле типа QTimer, которое отвечает интервал времени обновления кадра;
* image – поле типа QImage, содержащее данные изображения кадра;
* m\_networkManager – поле типа QNetworkAccessManager, которое отвечает за запрос получения кадра с камеры модели игрока;
* m\_reply – поле типа QNetworkReply\*, которое отвечает за ответ на запрос получения кадра с камеры модели игрока;
* ipCamera – поле типа QUrl, которое отвечает за хранение URL адреса модели игрока;
* processThread – поле типа QThread, которое отвечает за поток, в котором выполняется обработка кадров;
* frames – поле типа QList<QByteArray>, которое отвечает за список байтов, которые представляют кадры, полученные с камеры игрока;
* detector – экземпляр класса DetectBall (описан ниже), который отвечает за обнаружение мяча на кадрах;
* mutex – поле типа QMutex, которое используется для синхронизации доступа к данным кадров между основным потоком и потоком обработки.

Методы класса:

* replyFinished – метод, который вызывается при окончании ответа на запрос кадра с камеры модели игрока;
* captureFrame – метод, который отвечает за оправку запроса кадра с камеры модели игрока;
* startStream – метод, который отвечает за начало захвата кадров с камеры модели игрока;
* requestImage – метод, который отвечает за отображение кадра в пользовательском интерфейсе;
* processThreadFun – метод, который отвечает за извлечение кадра из списка frames, обработку их для обнаружения мяча в кадре и обновление поле image;
* imageChanged – сигнал, который emit-ится при обновлении поля image.

### 3.4.2 Класс DetectBall

Класс DetectBall предназначен для определения и выделения положения мяча в кадре.

Методы класса:

* findBall – метод, который предназначен для определения положение мяча в кадре;
* convertQImageToMat – метод, который предназначен для конвертации переменную типа QImage в переменную типа Mat, которая содержит данные кадра для обработки;
* convertMatToQImage – метод, который предназначен для конвертации переменную типа Mat в переменную типа QImage, которая содержит данные кадра для отображения.

### 3.4.3 Класс GameInfo

Класс GameInfo предназначен для хранения информации об игре. Позволяет извлекать из полученных данных и хранить информацию о количестве жизней игрока.

Поля класса:

* countLives – поле, которое содержит данные о количестве жизней модели игрока;
* reciever – экземпляр класса ReceiverData (описан ниже), предназначен для получения данных об игре с модели игрока;
* frontEnd – поле типа QObject, представляющее фронтенд приложения;
* recieveThread – поле типа QThread, которое отвечает за поток, в котором выполняется обработка данных, полученных от игрового платформы.

Методы класса:

* receiveProcess – метод, запускающий поток приёма и обработки данных;
* initReceiver – метод, инициализирующий соединение с игровой платформой;
* showError – сигнал, который emit-ится при возникновении ошибки и передаёт описание ошибки.

### 3.4.4 Класс ReceiverData

Класс ReceiverData предназначен для получения данных об игре.

Методы:

* init – метод, инициализирующий соединение с игровой платформой по указанному IP-адресу и порту;
* recieveData – метод, получающий данные от сервера и записывающий их в указанный буфер.

### 3.4.5 Класс ManagePlayer

Класс ManagePlayer предназначен для обработки пользовательского ввода. Позволяет получать сигналы от нажатых пользователем кнопок и вносить данные об этих сигналах в кадр для дальнейшей их пересылки модели игрока.

Поля класса:

* frame – поле, которое содержит данные сигналов для отправки на модель игрока;
* frontEnd – поле типа QObject, представляющее фронтенд приложения;
* sender – экземпляр класса SenderSignals (описан ниже), предназначен для отправки данных на игровую платформу о сигналах управления модели игрока.

Методы:

* inputProcessor – метод, который предназначен для обработки нажатых кнопок;
* buildFrame – метод, который предназначен для построения кадра для отправки на модель игрока;
* initSender – метод, инициализирующий соединение с игровой платформой;
* showError – сигнал, emit-ится при возникновении ошибки;
* startStreamSignal – сигнал, emit-ится при необходимости начать потоковую передачу видео с камеры;
* initGameInfoConnection – сигнал, emit-ится при необходимости установить соединение с игровой платформой для получения данных от неё.

### 3.4.6 Класс SenderSignals

Класс SenderSignals предназначен для отправки данных, содержавшие данные о сигналах управления моделью игрока.

Методы класса:

* init – метод, инициализирующий соединение с игровой платформой по указанному IP-адресу и порту;
* sendFrame – метод, который предназначен для отправки данных на модель игрока.

### 3.4.7 Класс Socket

Класс Socket предназначен для инициализации сокета для дальнейшего сетевого взаимодействия с моделью игрока.

Поля класса:

* socketinfo – поле типа struct sockaddr\_in, которое предназначено для обработка адреса сетевого взаимодействия;
* socketDev – поле типа int, которое содержит дескриптор сокета для сетевого взаимодействия с моделью игрока.

Методы класса:

* initSocket – метод для инициализации сокета;
* setSocket – метод, который предназначен для установки значения переменной socketDev;
* getSocket – метод, который предназначен для получения значения переменной socketDev.

### 3.4.8 Класс ConnectionToPlayer

Класс ConnectionToPlayer предназначен для подключение к Wi-Fi сети модели игрока.

Поля класса:

* avaliablePlayers – поле типа QVariantList, которое содержит список MAC-адресов доступных игроков;
* ipAddressPlayer – поле типа QUrl, которое отвечает за хранение URL адреса модели игрока;
* frontEnd – поле типа QObject, представляющее фронтенд приложения;
* javaWiFiApi – поле типа QJniObject, представляющее Java API для работы с Wi-Fi.

Методы класса:

* checkWiFiConnection – метод, проверяющий Wi-Fi соединение;
* connectToPlayer – метод, который предназначен для установки соединения с Wi-Fi сетью модели игрока;
* wifiScanComplete – метод, обрабатывающий результаты сканирования Wi-Fi и заполняющий список доступных игроков;
* setStatusConnection – метод, который получает статус соединения с выбранным игроком;
* wifiErrorMsg – сигнал, emit-ится при возникновении ошибки Wi-Fi соединения;
* initManageConnection – сигнал, emit-ится при возникновении ошибки Wi-Fi соединения.

### 3.4.9 Класс WiFiApi

Класс WiFiApi является Java классом и предназначен для работы c API Wi-Fi Android.

Поля класса:

* wifiManager – поле типа WifiManager, используемое для управления Wi-Fi соединениями.

Методы класса:

* wifiScanComplete – нативный метод, который вызывается после завершения сканирования Wi-Fi сетей;
* statusConnection – нативный метод, который вызывается для передачи статуса подключения;
* checkEnableWifi – метод для проверки, включен ли Wi-Fi;
* connectToPlayer – метод для подключения к Wi-Fi сети с заданным BSSID;
* checkConnection – метод проверки текущего Wi-Fi подключения;
* getScanResultsJson – метод для преобразования списка результатов сканирования Wi-Fi сетей в строку с перечислением BSSID;
* onReceive – переопределенный метод, который вызывается при получении широковещательного сообщения. Проверяет, успешны ли результаты сканирования, и вызывает нативный метод wifiScanComplete с результатами.

### 3.4.10 Классы из библиотек

В программной части дипломного проекта используется два класса из библиотек:

– QObject;

– QQuickImageProvider.

Класс QObject является базовым классом для всех объектов Qt. Класс QObject предоставляет механизмы для управления жизненным циклом объектов, сигналами и слотами для обмена информацией между объектами, а также поддержку иерархической структуры объектов.

Класс QQuickImageProvider является классом в Qt, который позволяет предоставлять изображения для использования в QML-интерфейсе. Он предоставляет способ загрузки изображений из различных источников, таких как файлы, базы данных или сетевые запросы, и динамически обновлять их в QML-интерфейсе.

# 4 ПРИНЦИПИАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В этом разделе подробно рассматривается практическая реализация функциональной схемы аппаратных средств устройства. Принципиальное проектирование является важной стадией в разработке устройства, на которой определяются и выбираются конкретные компоненты для реализации проекта. Сравнение и обоснование выбора конкретных компонентов проводилось на основе анализа их характеристик и соответствия требованиям, учитывая особенности данного дипломного проекта.

Принципиальная схема устройства управления моделью игрока представлена на чертеже ГУИР.400201.025 Э3.

## Компоненты устройства

Для реализации устройства управления необходимы следующие компоненты:

* микроконтроллер;
* органы управления (кнопки, джойстик);
* модуль радиопередачи сигнала;
* аккумулятор;
* устройство защиты аккумулятора;
* устройство отображения информации;
* устройство воспроизведения звукового сигнала.

### 4.1.1 Микроконтроллер

Микроконтроллер – это компактное интегральное устройство, включающее в себя процессор, память и периферийные устройства, предназначенные для выполнения конкретных задач управления. В отличие от более универсальных микропроцессоров, микроконтроллеры часто используются в встроенных системах, где они играют ключевую роль в управлении различными электронными устройствами и системами. Они могут выполнять задачи реального времени, управлять различными типами ввода и вывода, и обладают низким энергопотреблением, что делает их идеальными для автономных и портативных решений.

Для упрощения разработки устройства управления было принято решение использовать готовые платформы на базе микроконтроллеров. Готовые платформы на базе микроконтроллера представляют собой интегрированные решения, состоящие из микроконтроллера и всех необходимых компонентов для работы, таких как стабилизаторы напряжения, интерфейсы ввода/вывода, разъемы питания и часто дополнительные периферийные устройства.

При выборе подходящей платформы на базе микроконтроллера для устройства управления были учтены следующие критерии:

1. Компактность: Плата должна быть достаточно маленькой, чтобы легко интегрироваться в корпус пульта управления, не занимая много места и не утяжеляя устройство.
2. Производительность: Микроконтроллер должен обладать достаточной вычислительной мощностью для обработки сигналов от джойстика и кнопок, а также для управления экраном и обработки данных о состоянии устройства.
3. Питание: Плата должна поддерживать работу от аккумулятора и иметь низкое энергопотребление, чтобы обеспечить длительное время работы без подзарядки.
4. Возможности ввода/вывода: Необходимо достаточное количество портов для подключения всех необходимых компонентов, таких как джойстик, кнопки и экран.

Для сравнения были выбраны платы Arduino Nano v3.0, ESP8266, STM32F103C8T6. Результаты сравнения приведены в таблице 4.1.

Таблица 4.1 – Сравнение платформ на базе микронтроллера

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры сравнения | Arduino Nano v3.0 | ESP8266 | STM32F103C8T6 |
| Размеры | 45 мм × 18 мм | 48 мм × 25 мм | 53 мм × 22 мм |
| Процессор | ATmega328P | ESP8266 | ARM Cortex-M3 |
| Тактовая частота | 16 МГц | 80 МГц | 72 МГц |
| Память (Flash) | 32 КБ | 4 МБ | 64 КБ |
| Память (SRAM) | 2 КБ | 160 КБ | 20 КБ |
| Память (EEPROM) | 1 КБ | - | 2 КБ |
| Рабочее напряжение | 5 В | 3.3 В | 3.3 В |
| Макс. ток на вывод | 40 мА | 12 мА | 25 мА |
| Цифровые I/O порты | 14 | 11 | 37 |
| Аналоговые входы | 8 | 1 | 10 |
| PWM выходы | 6 | 4 | 12 |
| Интерфейсы | I2C, SPI, UART | I2C, SPI, UART | I2C, SPI, UART |

Исходя из сравнительной таблицы 4.1 в качестве микроконтроллера используется Arduino Nano v3.0. Основные причины выбора данной платформы включают следующие аспекты:

1. Компактность: Arduino Nano v3 является самой маленькой из рассмотренных плат, что делает её идеальной для интеграции в ограниченное пространство корпуса пульта управления. Размеры платы позволяют легко разместить её внутри устройства без необходимости увеличения габаритов пульта.
2. Производительность: Несмотря на относительно низкую тактовую частоту (16 МГц), ATmega328P, используемый в Arduino Nano v3, обладает достаточной мощностью для решения задач управления джойстиком, кнопками и экраном. Для большинства встроенных приложений, таких как управление игровыми моделями, этой производительности вполне достаточно.
3. Энергопотребление: Работа при напряжении 5 В и низкое энергопотребление делают эту плату подходящей для работы от аккумулятора, что особенно важно для мобильного устройства.
4. Достаточное количество портов ввода/вывода: Arduino Nano v3 предоставляет 14 цифровых I/O портов и 8 аналоговых входов, что достаточно для подключения всех необходимых компонентов пульта управления.

Описание прошивки микроконтроллера представлено в разделе 3.2. Исходный код программы представлен в приложении Б. Схема программы представлена на чертеже ГУИР.400201.025 ПД.

### 4.1.2 Модуль радиопередачи сигнала

В каждом устройства управления имеется радиомодуль, который служит для беспроводного обмена данными с игровой платформой. Игроки могут использовать устройства управления на расстоянии до десяти метров от игровой платформы. При этом беспроводная связь должна быть достаточно производительной и надежной, чтобы игроки могли своевременно реагировать на игровые события. Для связи с микроконтроллером используется интерфейс SPI.

Для сравнения были выбраны модели модулей радиопередачи NRF24L01, MX-05V, XD-RF-5V. Результаты сравнения приведены в таблице 4.2.

Таблица 4.2 – Сравнение модулей радиопередачи

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры сравнения | NRF24L01 | MX-05V | XD-RF-5V |
| Частота радиосигнала | 2.4 ГГц | 433 МГц | 433 МГц |
| Напряжение питания | 1.9 – 3.6 В | 3.5 – 12 В | 5 В |
| Потребляемый ток | до 14 мА | до 28 мА | до 6 мА |
| Рабочая температура | от -40 до +85 ºС | от -20 до +85 ºС | |
| Скорость  приёма/передачи  данных | до 2 Мб/с | до 10 Кб/с | |
| Расстояние  приёма/передачи | до 100 м | до 200 м | |

После анализа модулей радиопередачи, представленных в таблице 4.2 видно, что модуль NRF24L01 имеет лучшие характеристики, по сравнению с передатчиком MX-05V и приемником XD-RF-5V, но рассчитан на меньшее расстояние передачи радиосигнала. Для разрабатываемого устройства параметр радиопередачи не является критически важным, но при необходимости дальность передачи модулей NRF24L01 можно увеличить в связи с возможностью подключения к данным модулям антенны, увеличивающей дальность передачи сигнала.

В данном устройстве будут использованы модули радиопередачи NRF24L01, так как модули NRF24L01 работают в полудуплексном режиме. Каждый из двух узлов работает в режиме и приемника и передатчика: передатчик, отправив сообщение ждет на подтверждение приема сообщения со стороны приемника.

В соответствии с документацией к модулю NRF24L01 необходимо подключить фильтрующий конденсатор между контактами питания (VCC) и землей (GND). Номинальная емкость конденсатора составляет 100 мкФ, а максимальное напряжение 6.3 В. Конденсатор CL31A107MQHNNNE соответствует данным характеристикам.

### 4.1.3 Органы управления (кнопки, джойстик)

Органы управления пульта играют ключевую роль в взаимодействии пользователя с устройством. В данном проекте устройство управления будет оснащен джойстиком и кнопками, которые обеспечат интуитивное и удобное управление.

Джойстик – это устройство управления, использующееся для управления перемещением объектов. Он состоит из двух осей, по которым можно перемещать некоторый объект в двухмерном пространстве.

Для управления движением и направлением моделей игроков был выбран джойстик KY-023. Этот джойстик обладает следующими характеристиками:

1. Компактность: KY-023 имеет компактные размеры, что позволяет легко интегрировать его в корпус пульта управления без значительного увеличения его габаритов.
2. Двухосевой контроль: Джойстик поддерживает управление по двум осям (X и Y), что позволяет точно контролировать движение моделей игроков в двухмерном пространстве.
3. Центральная кнопка: В дополнение к двум осям, джойстик KY-023 оснащен встроенной кнопкой, которая активируется при нажатии на сам джойстик. Это добавляет дополнительный элемент управления, который можно использовать для выполнения различных действий, таких как старт или пауза игры.
4. Аналоговый выход: Джойстик KY-023 выдает аналоговые сигналы, которые легко считываются микроконтроллером через аналоговые входы. Это обеспечивает плавное и точное управление движением.

Кнопки, в свою очередь, представляют собой устройства управления, которые можно нажимать, чтобы выполнить какое-то действие. Для выполнения различных команд и взаимодействия с игровым интерфейсом будут использованы кнопки модели KAN0611-0901B. Эти кнопки обладают следующими характеристиками:

1. Надежность: KAN0611-0901B известны своей надежностью и долговечностью, что важно для часто используемого устройства, такого как игровой пульт.
2. Компактные размеры: Эти кнопки имеют небольшие размеры, что позволяет легко разместить их на корпусе пульта, обеспечивая удобный доступ для пользователя.
3. Тактильная обратная связь: Кнопки обеспечивают четкую тактильную обратную связь, что помогает пользователю понять, что нажатие было зарегистрировано.
4. Простота подключения: Кнопки KAN0611-0901B легко подключаются к микроконтроллеру через цифровые входы, что упрощает процесс их интеграции в систему управления.

### 4.1.4 OLED-дисплеи

OLED-экраны – экраны на органических светодиодах, не использующие заднюю подсветку, так как самостоятельно излучают свет, чем обеспечивают большую энергоэффективность устройства. Кроме того, они способны отображать реальный чёрный цвет, так как полностью тухнут в этот момент. Основными достоинствами OLED-дисплеев является их способность отображать более яркие и насыщенные цвета с высокой контрастностью, а также практически неограниченные углы обзора.

Для сравнения были выбраны следующие модели OLED-дисплеев: Waveshare Electronics 0,96inch, Troyka-OLED 0,96inch, GME12864-49 0,96inch. Результаты сравнения представлены в таблице 4.3.

Таблица 4.3 – Сравнение OLED-дисплеев.

|  |  |  |  |
| --- | --- | --- | --- |
| Основные характеристики | GME12864-49 0,96inch | Waveshare Electronics 0,96inch | Troyka-OLED 0,96inch |
| 1 | 2 | 3 | 4 |
| Питание | 3,3В; 5 В | 3,3В; 5 В | 3,3В; 5 В |
| Максимальное энергопотребление | 80 мВт | 80 мВт | 90 мВт |
| Разрешение | 128 × 64 точек | 128 × 64 точек | 128 × 64 точек |
| Цвет пикселей (светодиодов) | бирюзовый | желтый, синий | белый |
| Угол обзора | >160° | >160° | >160° |

Продолжение таблицы 4.3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Яркость | >120 кд/м2 | >120 кд/м2 | >120 кд/м2 |
| Время отклика | < 10 мкс | < 10 мкс | < 10 мкс |
| Размер | 24,7×27×1,2 мм | 50,8×25,4 ×1,2 мм | 50,8×25,4 ×1,2 мм |
| Рабочая температура | -40..+85 ⁰C | -20…+70℃ | -40..+85 ⁰C |
| Аппаратный интерфейс | I2C | I2C | I2C |

Основные требования для OLED-дисплея: высокая энергоэффективность, достаточная яркость, необходимый для размещения информации размер и разрешение. OLED-дисплей GME12864-49 0,96inch.

Данный дисплей имеет разрешение 128х64 и размер 24,7×27×1,2 мм. Напряжение питания 3,3В. Рабочий диапазон температур от -40 до +85 ºС. Имеет аппаратную реализацию интерфейса I2C.

### 4.1.5 Устройство воспроизведения звукового сигнала

В качестве устройства воспроизведения звукового сигнала был выбран пьезодинамик.

Пьезодинамик – это устройство, которое преобразует электрические сигналы в акустические волны (звук) с использованием пьезоэлектрического эффекта. Пьезоэлектрический материал внутри динамика деформируется при приложении электрического напряжения, создавая звуковые волны.

При выборе пьезодинамика необходимо учитывать следующие параметры:

1. Громкость (звуковое давление). Уровень звукового давления, измеряемый в децибелах (дБ), указывающий на громкость устройства.
2. Частотный диапазон. Диапазон частот, на которых пьезодинамик может эффективно работать.
3. Энергопотребление. Количество электроэнергии, потребляемое устройством.
4. Размеры. Физические размеры пьезодинамика, которые должны соответствовать конструктивным особенностям пульта.

Были отобраны следующие модели пьезодинамиков для сравнительного анализа: HC0905T/208, PS1240P02BT, ABT-402-RC. Результаты сравнения приведены в таблице 4.4.

Таблица 4.4 – Сравнение модулей пьезодинамиков

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры сравнения | HC0905T/208 | PS1240P02BT | ABT-402-RC |
| 1 | 2 | 3 | 4 |
| Рабочее напряжение | 3-20 В | 1-30 В | 3-24 В |

Продолжение таблицы 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Громкость (звуковое давление) | 85 дБ | 90 дБ | 80 дБ |
| Частотный диапазон | 2-5 кГц | 1-5 кГц | 2-4 кГц |
| Энергопотребление | 10 мА | 15 мА | 20 мА |
| Размеры | 9x5 мм | 12x4 мм | 10x6 мм |

Исходя из сравнительной таблицы 4.4 модель HC0905T/208 была выбрана по следующим причинам:

1. Соответствие рабочему напряжению: Устройство работает в диапазоне 3-20 В, что соответствует требованиям нашего проекта.
2. Приемлемая громкость: Уровень звукового давления 85 дБ обеспечивает достаточно громкое уведомление для пользователя.
3. Энергоэффективность: Низкое энергопотребление в 10 мА способствует увеличению срока службы аккумулятора.
4. Компактность: Размеры 9x5 мм подходят для интеграции в конструкцию пульта управления.

### 4.1.5 Аккумулятор

Для устройства управления был выбран литий-полимерный (Li-Po) аккумулятор. Литий-полимерные аккумуляторы обладают высокой плотностью энергии, малым весом и гибкостью в форм-факторах, что делает их идеальными для использования в устройстве управления.

При выборе модели аккумулятора учитывались следующие критерии выбора:

1. Емкость. Количество энергии, которое может хранить аккумулятор, измеряется в миллиампер-часах (мАч).
2. Напряжение. Номинальное напряжение аккумулятора, которое должно соответствовать требованиям устройства.
3. Размеры. Физические размеры аккумулятора, чтобы он подходил для интеграции в конструкцию пульта.
4. Вес. Важный параметр для портативных устройств, чтобы не увеличивать общий вес пульта.

Согласно этим критериям были выбраны следующие модели аккумуляторов: LP402535, LP503040, LP401730. Подробные данные о сравнении представлены в таблице 4.5.

Таблица 4.5 – Сравнение аккумуляторов

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры сравнения | LP402535 | LP503040 | LP401730 |
| 1 | 2 | 3 | 4 |
| Емкость | 320 мАч | 550 мАч | 150 мАч |

Продолжение таблицы 4.5

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Напряжение | 3.7 В | 3.7 В | 3.7 В |
| Размеры | 35x25x4 мм | 40x30x5 мм | 30x17x4 мм |
| Вес | 10 г | 12 г | 8 г |
| Цикл заряд-разряд | 500 циклов | 500 циклов | 500 циклов |

Исходя из сравнительной таблицы 4.5 модель LP402535 была выбрана по следующим причинам:

1. Оптимальная емкость: 320 мАч предоставляет достаточно энергии для продолжительной работы пульта управления, обеспечивая баланс между емкостью и размером.
2. Соответствие напряжению: Номинальное напряжение 3.7 В идеально подходит для нашего устройства.
3. Компактные размеры: Размеры 35x25x4 мм позволяют легко интегрировать аккумулятор в конструкцию пульта управления.
4. Легкий вес: Вес 10 г минимально влияет на общий вес пульта, обеспечивая удобство использования.
5. Долговечность: 500 циклов заряд-разряд гарантируют длительный срок службы аккумулятора.

### 4.1.6 Устройство защиты аккумулятора

Для обеспечения безопасности и долговечности литий-полимерного (Li-Po) аккумулятора в нашем проекте необходимо использовать устройство защиты аккумулятора (Battery Protection Circuit, BPC). Такое устройство предотвращает критические состояния, которые могут повредить аккумулятор или создать опасность для пользователя.

При выборе устройства защиты необходимо учитывать следующие критерии:

1. Совместимость с типом аккумулятора: Устройство должно быть специально разработано для работы с Li-Po аккумуляторами.
2. Рабочее напряжение: Диапазон напряжений, в котором устройство может эффективно работать.
3. Максимальный ток защиты: Максимально допустимый ток, который устройство может выдержать без срабатывания защиты.
4. Физические размеры: Размеры устройства должны быть совместимы с конструкцией пульта управления.
5. Защита от короткого замыкания: Наличие данной функции и время срабатывания.
6. Простота интеграции: Устройство должно быть легко интегрировано в схему нашего устройства.

Для сравнительного анализа были отобраны следующие модели устройств защиты аккумуляторов: PCM-LiPo-1S1P-3A, HW-373 v1.1, DW01A + FS8205A. Подробные данные о сравнении представлены в таблице 4.6.

Таблица 4.6 – Сравнение устройств защиты аккумуляторов

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры сравнения | PCM-LiPo-1S1P-3A | HW-373 v1.1 | DW01A + FS8205A |
| Совместимость | Li-Po | Li-Po | Li-Po |
| Рабочее напряжение | 2.5 В - 4.2 В | 2.4 В - 4.3 В | 2.4 В - 4.3 В |
| Максимальный ток защиты | 3 А | 3 А | 2 А |
| Размеры | 20x15x2 мм | 20x20x2 мм | 12x8x1 мм |
| Защита от короткого замыкания | Да | Да | Да |
| Время срабатывания защиты от КЗ | <1 мс | <1 мс | <1 мс |

Исходя из сравнительной таблицы 4.6 модель HW-373 v1.1 была выбрана по следующим причинам:

Совместимость: Устройство специально разработано для работы с Li-Po аккумуляторами.

1. Рабочее напряжение: Диапазон от 2.4 В до 4.3 В идеально подходит для наших нужд, обеспечивая защиту как от переразряда, так и от перенапряжения.
2. Максимальный ток защиты: 3 А обеспечивает достаточную защиту при пиковых нагрузках, что идеально подходит для нашего пульта управления.
3. Компактные размеры: Размеры 20x20x2 мм позволяют легко интегрировать устройство в конструкцию пульта управления без значительных изменений.
4. Защита от короткого замыкания: Устройство обеспечивает защиту от короткого замыкания с быстрым временем срабатывания менее 1 мс, что критически важно для безопасности.
5. Дополнительные функции: Наличие индикатора состояния позволяет визуально контролировать работу устройства и состояние аккумулятора, что повышает удобство эксплуатации.

## 4.3 Разработка схемы питания

В разрабатываемом устройстве управления питание осуществляется от аккумулятора. Также в данном устройстве используется две линии питания соответственно: 5В, 3,3В. Необходимо выбрать стабилизатор напряжения, которые будут стабилизировать напряжение до 3,3В, и преобразователь напряжения, который будет повышать напряжение с 3,7В до 5В.

Для выбора требуется обращать внимание на максимальное значение тока, которое может протекать через стабилизатор и преобразователь. Потребляемой ток компонентов устройства управления представлен в таблице 4.7.

Таблица 4.7 – Потребление тока компонентов

|  |  |  |  |
| --- | --- | --- | --- |
| Компонент | Кол-во | Uпит, В | Imax, мА |
| Плата Arduino Nano v3.0 | 1 | 5 | 22 |
| Модуль радиопередачи NRF24L01 | 1 | 3,3 | 16 |
| Джойстик KY-023 | 1 | 5 | 8 |
| OLED-дисплей GME12864-49 0,96inch | 1 | 5 | 20 |
| Пьезодинамик HC0905T/208 | 1 | 5 | 80 |
| Тактовая кнопка, 3 шт. | 3 | 5 | 0,03 |
| Суммарный ток, потребляемый устройствами 5 В, мА | | | 130,03 |
| Суммарный ток, потребляемый устройствами 3,3 В, мА | | | 16 |

Исходя из таблицы 4.7 суммарный максимальный потребляемый ток устройств по линии 5 В составляет 130,03 мА, соответственно критериями выбора повышающего преобразователя являются входное напряжение 3,7 В и максимальным протекающим током не менее 1 А.

Согласно критериям выбора был выбран повышающий DC-DC преобразователь MINGWU DCDC5.0V. Характеристики повышающего DC-DC преобразователя MINGWU DCDC5.0V представлены в таблице 4.8.

Таблица 4.8 – Характеристика повышающего DC-DC преобразователя MINGWU DCDC5.0V

|  |  |
| --- | --- |
| Параметр | MINGWU DCDC5.0V |
| Входное напряжение | 0,9-5 В |
| Выходное напряжение | 5 В |
| Выходной ток | 1 А |

Исходя из таблицы 4.7 суммарный максимальный потребляемый ток устройств по линии 3,3 В составляет 16 мА, соответственно критериями выбора повышающего преобразователя являются входное напряжение 3,7 В и максимальным протекающим током не менее 200 мА.

Согласно критериям выбора был выбран стабилизатор напряжения 662K. Характеристики стабилизатора напряжения 662K представлены в таблице 4.9.

Таблица 4.9 – Характеристика стабилизатора напряжения 662K

|  |  |
| --- | --- |
| Параметр | 662K |
| Входное напряжение | 1.8 … 6 В |
| Выходное напряжение | 3,3 В |
| Выходной ток | 200 мА |

# 5 РаЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## Разработка пользовательского интерфейса

Разработка пользовательского интерфейса является важным шагом в создании визуального прототипа интерфейса пользователя для приложения управления для симулятора компьютерной игры «Пинг-Понг». Разработка пользовательского интерфейса помогает не только визуализировать основные элементы интерфейса, но и облегчает процесс обсуждения и внесения изменений до начала программирования. В данном подразделе будут рассмотрены основные элементы интерфейса, их расположение и функциональность.

### 5.1.1 Основной экран

Основной экран приложения и диалоговые окна составляют ключевые части пользовательского интерфейса. В данном подразделе будет подробно описан основной экран, а также перечислены его основные элементы.

Основной экран является центральной частью приложения, предоставляя пользователю возможность управлять моделью игрока и наблюдать за игрой через изображение с камеры. Основной экран включает следующие элементы:

* изображение с камеры;
* кнопка перемещения влево;
* кнопка перемещения вправо;
* кнопка активации отталкивающего механизма;
* кнопка восстановления жизней;
* отображение количества жизней.

Изображение с камеры расположено в центре экрана, занимая основное пространство, чтобы пользователь мог удобно наблюдать за игрой и действиями модели. Размер изображения должен быть достаточно большим, чтобы обеспечить хорошую видимость деталей.

Кнопки перемещения влево и вправо находятся слева от изображения на одном уровне с ним. Кнопка перемещения влево и вправо расположены близко друг другу для того, чтобы пользователь мог быстро среагировать и сразу изменить направление движения модели игрока. Эти кнопки предназначены для управления движением модели игрока влево и вправо и должны быть достаточно крупными и удобно расположенными для быстрого доступа.

Кнопка активации отталкивающего механизма расположена справа от изображения с камеры на одном уровне с ним. Она должна быть легко доступной и выделяться визуально, чтобы пользователь мог быстро активировать механизм при необходимости.

Кнопка восстановления жизней находится в правом верхнем углу экрана. Это позволяет пользователю легко найти и нажать её, чтобы восстановить жизни модели игрока.

Количество оставшихся жизней отображается в левом верхнем углу экрана. Этот элемент должен быть всегда видимым, чтобы пользователь мог контролировать состояние модели игрока.

Разработанный макет основного экрана представлена на рисунке 5.1.



Рисунок 5.1 – Разработанный макет основного экрана

### 5.1.2 Диалоговые окна

Приложение будет включать три диалоговых окна:

* окно выбора игрока для подключения;
* окно сообщения об ошибке;
* окно сообщения о нулевом остатке жизней.

Эти окна помогают пользователю взаимодействовать с приложением и получать важную информацию.

#### 5.1.2.1 Окно выбора игрока для подключения

Окно выбора игрока для подключения является важной частью интерфейса приложения, так как оно позволяет пользователю выбрать модель игрока, с которой будет установлено соединение по Wi-Fi. Это окно должно быть интуитивно понятным и легко доступным для пользователя.

Окно выбора игрока для подключения появляется в центре экрана, поверх основного экрана. Оно должно быть достаточно большим, чтобы уместить все необходимые элементы, но при этом не перекрывать весь основной экран. Также окно выбора игрока должно быть стилистически согласовано с остальным интерфейсом приложения. Оно должно иметь четкие и понятные элементы управления, с хорошей читаемостью текста и достаточно крупными кнопками для удобного взаимодействия. Фон окна может быть полупрозрачным, чтобы пользователь видел, что он находится поверх основного экрана, но при этом основной экран не отвлекал его внимание.

Окно выбора игрока включается в себя следующие элементы:

* заголовок: "Выберите игрока";
* кнопки для выбора игрока: Список кнопок, каждая из которых соответствует доступной для подключения модели игрока;
* кнопки: "Обновить" и "Выйти".

Разработанный макет окна выбора игрока для подключения представлен на рисунке 5.2.



Рисунок 5.2 – Разработанный макет окна выбора игрока для подключения

#### 5.1.2.2 Окно сообщения об ошибке

Окно сообщения об ошибке является важным элементом пользовательского интерфейса, так как оно информирует пользователя о возникновении проблем или неполадок в работе приложения. Это окно должно быть легко заметным и понятным, чтобы пользователь мог быстро разобраться в причине ошибки и принять необходимые меры.

Окно сообщения об ошибке появляется в центре экрана, поверх основного экрана. Оно должно быть достаточно большим, чтобы уместить текст сообщения и кнопку для закрытия окна, но при этом не занимать слишком много места.

Окно сообщения об ошибке должно быть стилистически согласовано с остальным интерфейсом приложения. Оно должно иметь четкие и понятные элементы управления, с хорошей читаемостью текста и достаточно крупной кнопкой для удобного взаимодействия.

Окно сообщения об ошибке включает в себя следующие элементы:

* заголовок: «Ошибка»;
* текст сообщения: Описание ошибки;
* кнопка: «Закрыть».

Разработанный макет окна сообщения об ошибке представлен на рисунке 5.3.



Рисунок 5.3 – Разработанный макет окна сообщения об ошибке

#### 5.1.2.3 Окно сообщения о нулевом остатке жизней

Окно сообщения о нулевом остатке жизней является важной частью пользовательского интерфейса, так как оно уведомляет пользователя о том, что у модели игрока закончились жизни. Это окно должно быть легко заметным и понятным, чтобы пользователь мог быстро принять решение о дальнейших действиях.

Окно сообщения о нулевом остатке жизней появляется в центре экрана, поверх основного экрана. Оно должно быть достаточно большим, чтобы уместить текст сообщения и кнопку восстановление жизней, но при этом не занимать слишком много места.

Окно сообщения об ошибке включает в себя следующие элементы:

* иконка, иллюстрирующая отсутствие жизней;
* заголовок: «Осталось 0 жизней»;
* текст сообщения: «Необходимо восстановить жизни»;
* кнопка: «Восстановить жизни».

Разработанный макет окна сообщения о нулевом остатке жизней представлен на рисунке 5.4.

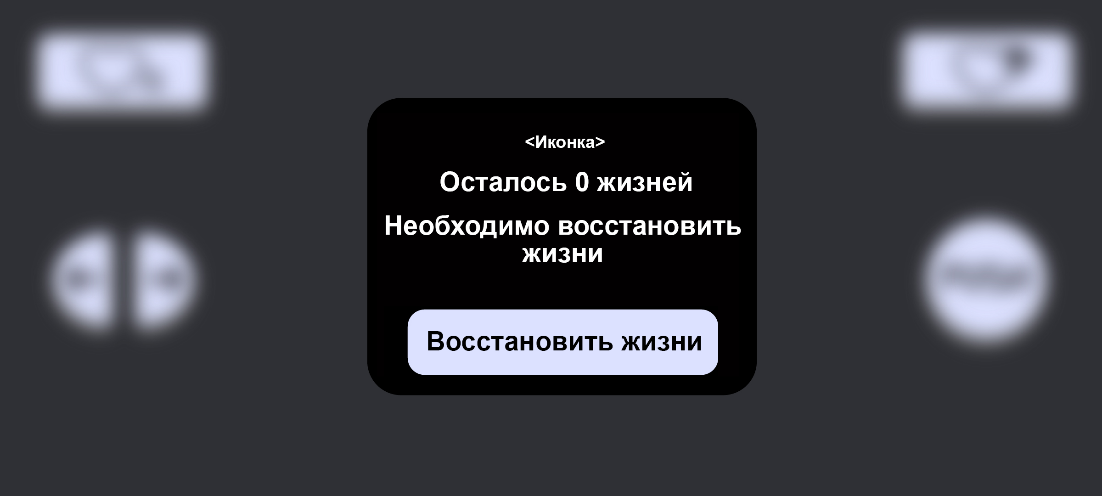


Рисунок 5.4 – Разработанный макет окна сообщения о нулевом остатке жизней

### 5.1.3 Цветовая палитра

Цветовая палитра интерфейса играет ключевую роль в создании визуального стиля и обеспечивает удобство использования приложения. В данном интерфейсе используются следующие цвета:

* #01174b (Темно-синий);
* #dbe1ff (Светло-голубой);
* #2f3036 (Темно-серый);
* #121318 (Черный);
* #ffffff (Белый);

Темно-синий цвет ассоциируется с надежностью и профессионализмом. Он обеспечивает хороший контраст с более светлыми элементами интерфейса, делая текст и иконки более читаемыми. Этот цвет часто используется для создания серьезного и доверительного настроения, что особенно важно для интерфейсов, требующих внимания и концентрации.

Светло-голубой цвет создает ощущение легкости и спокойствия. Он используется для фона и подсветки активных элементов, чтобы выделить их на общем фоне интерфейса. Этот цвет приятен для глаз и не вызывает усталости при длительном использовании.

Темно-серый цвет обеспечивает хороший контраст как с темными, так и со светлыми фоновыми цветами. Он используется для текста и иконок, что делает их легко читаемыми и заметными. Также этот цвет подходит для второстепенных кнопок, которые не должны отвлекать внимание от основных действий пользователя.

Черный цвет добавляет глубину и четкость интерфейсу. Он используется для фона и разделительных линий, создавая визуальные границы между различными элементами интерфейса. Этот цвет также хорошо подходит для теней, добавляя объем и улучшая восприятие интерфейса.

Белый цвет символизирует чистоту и простоту. Он используется для текста на темных фонах, делая его максимально читаемым. Белый цвет также применяется для кнопок и других интерактивных элементов, чтобы они выделялись и были легко доступны для пользователя.

## 5.2 Функция распознавание мяча

Функция findBall предназначена для обнаружения мяча в кадре.

* 1. Преобразуется входное изображение в формат OpenCV
  2. Преобразуется цветное изображение в оттенки серого
  3. Сглаживается изображение: удаляется шум с изображения с помощью медианного фильтра.
  4. Вызывается функция HoughCircles для нахождения окружности на изображении с помощью алгоритма Хафа.
  5. Результат обработки записывается в circles.
  6. Итерируется по найденным окружностям.
  7. Извлекается информация об i-ой окружности.
  8. Извлекаются координаты центра окружности.
  9. Отрисовывается центр окружности зеленым цветом.
  10. Извлекается радиус окружности.
  11. Отрисовывается окружность красным цветом.
  12. Преобразуется изображение в формат Qt QImage и возвращается.

## 5.3 Функция connectToPlayer

Функция connectToPlayer предназначена для подключения устройства к модели игрока по Wi-Fi.

* 1. Получается объект ConnectivityManager для управления сетевыми подключениями.
  2. Создаётся спецификатор сети NetworkSpecifier, который описывает сеть Wi-Fi. Устанавливается MAC-адрес устройства и пароль для WPA2.
  3. Создается объект NetworkRequest, который описывает запрос на подключение к сети. Указывается, что запрос касается сети Wi-Fi. Удаляется требование доступа к интернету. Устанавливается спецификатор сети, созданный ранее.
  4. Создается объект NetworkCallback, который будет получать уведомления о событиях, связанных с сетью.
  5. Отправляется запрос на подключение к сети, используя созданный NetworkRequest и NetworkCallback.
  6. Устанавливается время ожидания ответа в 50 секунд.

# 6 СБОРКА УСТРОЙСТВА

В этом разделе описывается процесс сборки устройства управления. Сборка устройства разделена на три основные этапа:

* разработка печатной платы;
* разработка корпуса устройства;
* окончательная сборка устройства.

Схема печатной платы представлена на схеме ГУИР.400201.025 Э4.

## 6.1 Разработка печатной платы

Разработка печатной платы является одним из ключевых этапов в создании устройства управления. Печатная плата представляет собой основу, на которой размещаются электронные компоненты устройства. Разработка печатной платы разделена на два этапа:

* проектирование печатной платы;
* изготовление печатной платы.

### 6.1.1 Проектирование печатной платы

Проектирование печатной платы является многоэтапным процессом, который начинается с разработки электрической схемы и заканчивается созданием производственных файлов. Каждый этап играет ключевую роль в создании функционального и надежного устройства.

Первым этапом является разработка электрической схемы. На этом этапе создается детализированная электрическая схема устройства, включающая все компоненты и их взаимосвязи. В качестве программного обеспечения для проектирования схем была выбрана программа EasyEDA. После происходит размещение всех компонентов на схеме и их соединение в соответствии с архитектурой устройства. Важно провести проверку схемы, чтобы убедиться в правильности соединений и отсутствии ошибок, таких как незамкнутые цепи или короткие замыкания.

Следующим этапом является размещение компонентов на плате. Сначала определяются размеры и форма платы в соответствии с требованиями устройства. Размер проектируемой платы составляет 65x40 мм. Форма платы – прямоугольная. Крупные и критически важные компоненты, такие как микроконтроллеры, коннекторы и источники питания, размещаются первыми, чтобы обеспечить оптимальное функционирование устройства. Затем распределяются все остальные компоненты, учитывая электрические и механические требования. Нужно также позаботиться о том, чтобы компоненты были легко доступны для пайки и обслуживания.

Разводка печатных проводников, или трассировка, – это следующий этап, на котором осуществляется прокладка проводников, соединяющих компоненты в соответствии с электрической схемой. Сначала определяется количество слоев платы. Разрабатываемая плата будет являться двусторонней. Затем производится трассировка силовых цепей для питания и заземления, чтобы минимизировать сопротивление и индуктивность. После этого прокладываются сигнальные проводники с учетом требований к сигналам, таких как частота и чувствительность к помехам. При трассировке необходимо учитывать правила проектирования, включая ширину проводников, зазоры и минимальные радиусы изгиба. В данной разрабатываемой плате ширина проводников составляет 1 мм, ширина зазора не менее 0,6 мм.

После завершения трассировки проводится проверка и оптимизация платы. Электрическая проверка позволяет убедиться, что проект соответствует заданным правилам проектирования. Анализ целостности сигнала помогает выявить возможные проблемы с высокочастотными сигналами. Термическая проверка оценивает тепловые характеристики платы и помогает предотвратить возможные перегревы. На этом этапе также проводится оптимизация размещения и трассировки для улучшения характеристик платы, таких как уменьшение длины проводников, улучшение теплового управления и минимизация электромагнитных помех.

Завершающим этапом является создание производственных файлов. Генерируются файлы, необходимые для изготовления печатной платы, которые содержат информацию о трассировке и местоположении компонентов. Также создаются файлы для сверлильных машин, содержащие информацию о координатах и диаметрах отверстий. Создается список всех компонентов с указанием их позиций и спецификаций. Подготавливаются сборочные чертежи, включающие информацию о размещении компонентов и пайке.

### 6.1.2 Изготовление печатной платы

Изготовление печатных плат – это процесс, включающий несколько этапов, начиная с подготовки материалов и заканчивая финальной проверкой готовых плат. Каждый этап важен для получения качественного и надежного изделия.

Процесс изготовления печатных плат начинается с выбора и подготовки материалов. Основа платы будет использоваться односторонний стеклотекстолит, для травления – лимонная кислота, перекись водорода и соль, для сверления – сверла диаметром 0,9 и 3 мм.

Одним из важнейших этапов является нанесение рисунка печатной платы. Для этого используется УФ-принтер Mimaki UJF-3042, с помощью которого лаком на поверхность стеклотекстолита наносится рисунок дорожек и контактных площадок. Лак наносится в три слоя для обеспечения четкости и дополнительной защиты. Каждый слой лака отверждается под воздействием ультрафиолетового света, что придаёт ему прочность и устойчивость. Лак не только формирует рисунок дорожек, но и служит защитным покрытием для проводников, предотвращая их окисление и повреждение в процессе производства и эксплуатации.

После нанесения рисунка плата погружается в травильный раствор, где незащищенные лаком участки меди растворяются. Травление платы осуществляется в растворе:

* 100 миллилитров перекиси водорода;
* 50 грамм лимонной кислоты;
* семь грамм поваренной соли.

По завершении травления плата промывается и высушивается.

На рисунке 6.1 представлена плата после травления.

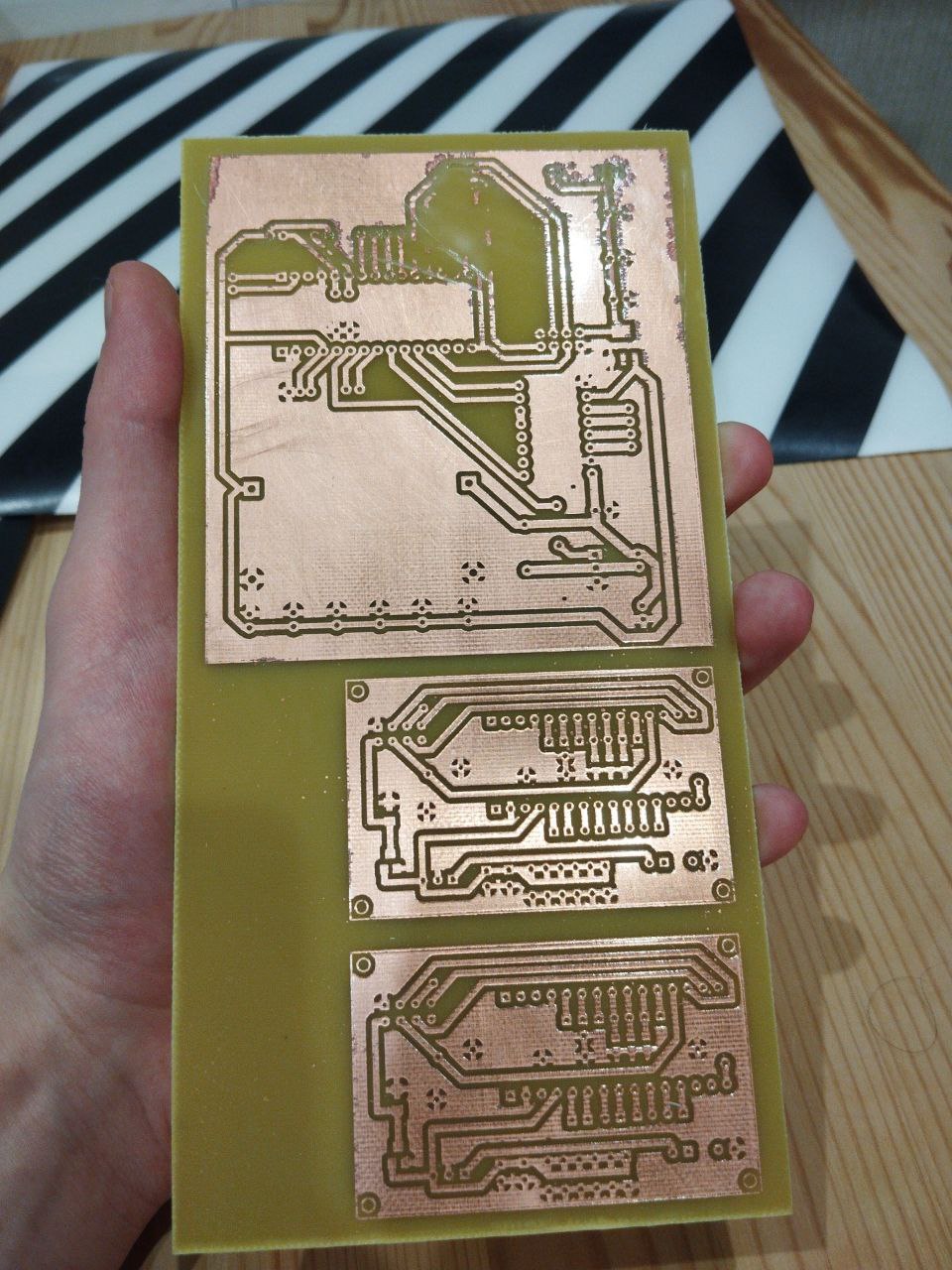


Рисунок 6.1 – Плата после травления

Следующим этапом является сверление отверстий под выводные компоненты и переходные отверстия с использованием сверлильного станка. Необходимо просверлить все необходимые для монтажа отверстия необходимого диаметра, а именно отверстия 0,9 мм для крепления элементов и 3 мм для крепления платы к корпусу.

Затем осуществляется установка и пайка электронных компонентов на плату. После монтажа компонентов плата проходит проверку на соответствие электрической схеме, тестирование на отсутствие коротких замыканий и правильность соединений, а также визуальный контроль качества исполнения и проверку на наличие дефектов.

На рисунке 6.2 представлена плата после пайки и монтажа функциональных узлов.

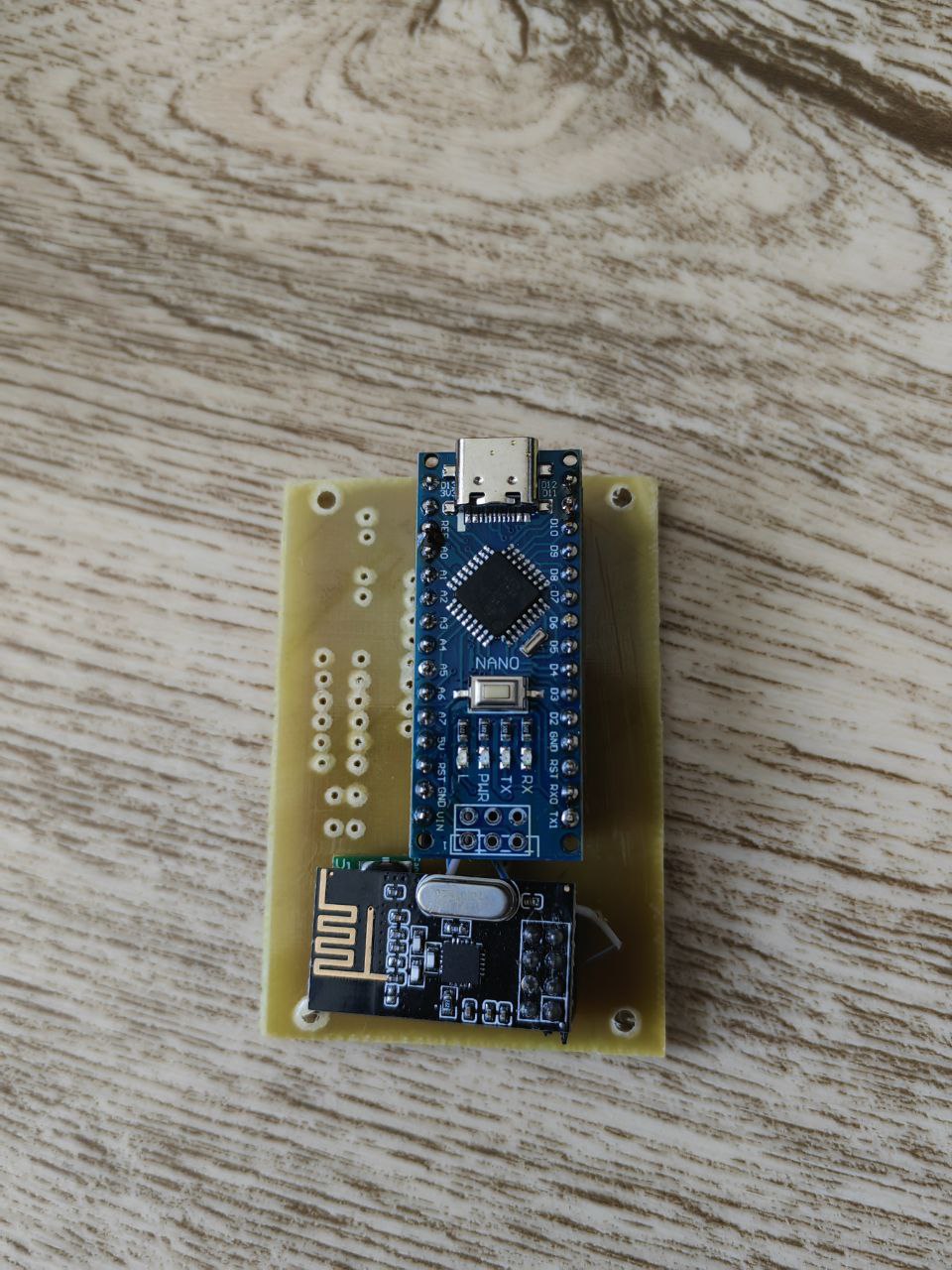


Рисунок 6.2 – Плата после пайки и монтажа функциональных узлов

## 6.2 Разработка корпуса устройства

Разработка корпуса устройств включает в себя несколько ключевых этапов, начиная от концепта дизайна и заканчивая созданием физического прототипа с помощью 3D-принтера.

Первым шагом является определение основных требований к корпусу. Это включает в себя размеры, форму, расположение кнопок, разъемов и других элементов. Для начала создаются эскизы на бумаге или в графическом редакторе, чтобы визуализировать идеи и понять, как будет выглядеть конечный продукт.

После утверждения концепта дизайна создается 3D-модель корпуса с использованием программного обеспечения для 3D-проектирования, такого как AutoCAD. В процессе моделирования учитываются все технические аспекты: размещение печатной платы, крепежные элементы и другие важные детали. 3D-модель позволяет визуализировать конструкцию и вносить необходимые изменения перед печатью.

После создания 3D-модели проводится её проверка на соответствие всем требованиям и нормам. Это включает проверку размеров, совместимости с внутренними компонентами и общей эргономики. На этом этапе вносятся изменения и оптимизации, чтобы улучшить конструкцию и облегчить последующее изготовление.

На рисунках 6.3, 6.4, 6.5 представлены 3D модели нижней, верхней части корпуса и корпуса кнопок.

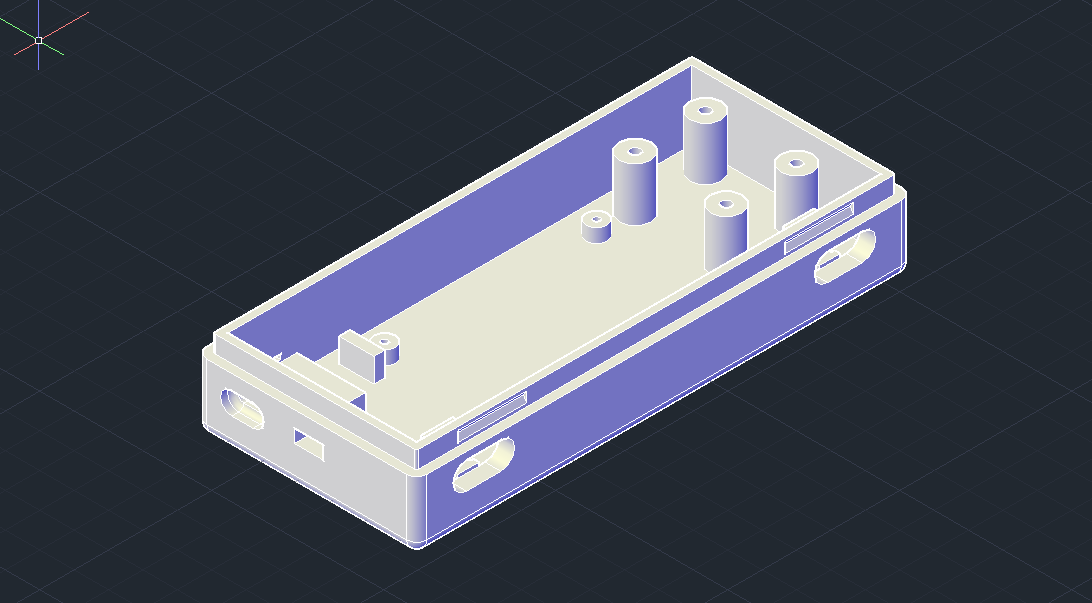


Рисунок 6.3 – 3D модель нижней части корпуса

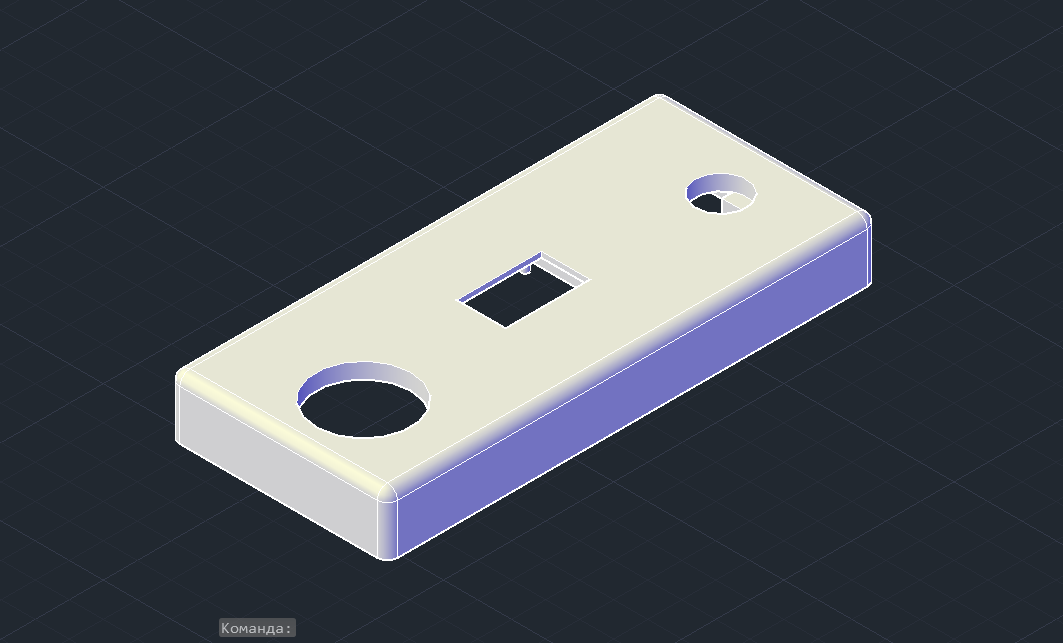


Рисунок 6.4 – 3D модель верхней части корпуса



Рисунок 6.5 – 3D модель корпусов кнопок

Когда 3D-модель завершена и проверена, начинается этап печати финального варианта корпуса. Модель отправляется на 3D-принтер, где изготавливается корпус. Для печати используются ABS пластик. В процессе печати учитываются параметры слоя, заполнение, скорость и другие параметры, влияющие на качество и прочность полученной детали.

На рисунках 6.6, 6.7, 6.8 представлены напечатанные 3D модели нижней, верхней части корпуса и корпуса кнопок.



Рисунок 6.6 – Напечатанная 3D модель нижней части корпуса



Рисунок 6.7 – Напечатанная 3D модель верхней части корпуса



Рисунок 6.8 – Напечатанная 3D модель корпусов кнопок

## 6.3 Сборка устройства

На этапе сборки устройства необходимо установить функциональные узлы схемы, которые крепятся непосредственно к корпусу, поэтому данные элементы не были добавлены на печатную плату. В разрабатываемом устройстве к таким элементам относятся джойстик, экран, три тактовые кнопки, аккумулятор и переключатель для включения устройства.

Экран вынесен отдельно и соединяется с основной платой устройства четырьмя проводами: два аналоговых информационных и два провода питания.

Джойстик подключается к основной плате пятью проводами, включая три информационных и два провода питания.

Три тактовые кнопки вынесены отдельно для удобства монтажа и соединяются с основной платой при помощи шестью проводами: три информационных и три провода питания.

Установка основной платы в корпус устройства осуществляется на стойки, расположенные на высоте 3 мм. Установка джойстика также осуществляется на стойки, расположенные на высоте 11 мм. Монтаж платы с экраном и кнопками производится на верхнюю часть корпуса устройства.

На рисунке 6.9 представлена плата с установленными внешними элементами.



Рисунок 6.9 – Плата с установленными внешними элементами

На рисунке 6.10 представлено собранное устройство управления.



Рисунок 6.10 – Собранное устройство управления

# 7 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

В этом разделе изложена программа и методология тестирования, являющиеся ключевым этапом в проверке качества создаваемого устройства управления для симулятора компьютерной игры «Пинг-Понг». Методы тестирования, используемые здесь, позволяют объективно оценить функциональность и соответствие продукта заявленным требованиям. Подробно описаны методика проведения испытаний, выбор тестовых сценариев, критерии оценки результатов, а также особенности проведения экспериментов и анализа полученных данных. Также в разделе приводится информация об используемом оборудовании, программном обеспечении и ресурсах, необходимых для успешного выполнения тестовых процедур.

Тестирование разделено на 2 этапа:

* тестирование аппаратной части;
* тестирование программной части.

## 7.1 Объекты тестирования

Объектами тестирования являются:

* аппаратная реализация системы управления;
* программная реализация системы управления.

## 7.1 Порядок проведения тестов для аппаратной части

В таблице 7.1 представлен список тестов аппаратной системы управления для симулятора компьютерной игры Пинг-Понг.

Таблица 7.1 – Список тестов аппаратной системы управления для симулятора компьютерной игры Пинг-Понг

|  |  |
| --- | --- |
| Наименование тестирования | Раздел с описанием тестирования |
| 1 Подготовка тестовой среды | Раздел 7.2 |
| 2 Тестирование подачи сигнала передвижения модели игрока | Раздел 7.3 |
| 3 Тестирование подачи сигнала отталкивания мяча | Раздел 7.4 |
| 4 Тестирование подачи сигнала восстановления жизней игрока | Раздел 7.5 |
| 5 Тестирование отображения информации на экране | Раздел 7.6 |
| 6 Тестирование воспроизведение звукового сигнала | Раздел 7.7 |

## 7.2 Подготовка тестовой среды

Подготовка включает создание среды, в которой будут проводиться испытания аппаратного устройства с целью оценки его работоспособности.

### 7.2.1 Необходимые для тестирования компоненты

Для проведения программы тестирования и испытаний аппаратной системы управления потребуются следующие компоненты:

1. Игровая платформа. Модели игроков, управление которых осуществляется с помощью устройства управления, расположены на игровой платформе.
2. Игровой мяч. Мяч для игры поставляется в комплекте. Также подойдет любой мяч, который соответствует следующим критериям: достаточно легкий, чтобы при отталкивании моделью игрока смог достигнут ворот противника, иметь небольшой диаметр, чтобы свободно проходить в ворота игрока.
3. Блок питания. Блок питания необходим для подключения устройства к сети 220 В / 50 Гц. Оригинальный блок питания идет в комплекте с игровой платформой.

### 7.2.2 Включение игровой платформы

Включение игровой платформы осуществляется посредством подключения блока питания к разъему на корпусе и перевода выключателя в положение «ON». В результате должны включится светодиоды блоков отображения на стороне каждого игрока, что сообщает о корректном включении устройства и завершении инициализации компонентов системы.

### 7.2.3 Включение устройства управления

Включение устройства осуществляется посредством перевода выключателя на его корпусе в положение «ON». Индикатором успешного включения устройства является появление информации на встроенном экране.

## 7.3 Методика тестирования подачи сигнала передвижения модели игрока

Передвижение модели игрока осуществляется с помощью джойстика или двух кнопок. Методика тестирования подачи сигнал передвижения модели игрока представлена в таблице 7.2.

Таблица 7.2 – Методика тестирования подачи сигнала передвижения модели игрока

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| 1 | 2 |
| Наклон джойстика влево | Передвижение модели игрока влево, со скоростью прямо пропорциональной углу наклона |

Продолжение таблицы 7.2

|  |  |
| --- | --- |
| 1 | 2 |
| Наклон джойстика вправо | Передвижение модели игрока вправо, со скоростью прямо пропорциональной углу наклона |
| Нажатие и удержание кнопки передвижения влево | Передвижение модели игрока влево, с максимальной скоростью |
| Нажатие и удержание кнопки передвижения вправо | Передвижение модели игрока вправо, с максимальной скоростью |

## 7.4 Методика тестирования подачи сигнала отталкивания мяча

Подача сигнала отталкивания мяча осуществляется посредством кнопки. Методика тестирования подачи сигнала отталкивания мяча представлена в таблице 7.3.

Таблица 7.3 – Методика тестирования механизма отталкивания мяча

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Однократное нажатие кнопки | Активация механизма отталкивания мяча и возврат его в исходное положение |
| Удержание кнопки | Многократная активация механизма отталкивания мяча и возврат его в исходное положение |

## 7.5 Тестирование подачи сигнала восстановления жизней игрока

Подача сигнала восстановление жизней осуществляется посредством нажатия и удерживания кнопок передвижения в течении 5 секунд. Методика тестирования подачи сигнала восстановления жизней игрока представлена в таблице 7.4.

Таблица 7.4 – Методика тестирования подачи сигнала восстановления жизней игрока

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Нажать и удерживать кнопки передвижения в течении 5 секунд | Отображение на дисплее 11 жизней, включение всех 11 светодиодов в блоке отображения игровой платформы |

## 7.6 Тестирование отображения информации на экране

Отображение информации о количестве жизней игрока и проценте заряда аккумулятора осуществляется посредством дисплея. Методика тестирования отображения информации на экране представлена в таблице 7.5.

Таблица 7.5 – Методика тестирования отображения информации на экране

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Перевести выключатель на устройстве управления в положение «ON» | Отображение на дисплее текущего количества жизней игрока и процента заряда аккумулятора |
| Поместить и извлечь мяч в ворота игрока один раз, когда количество жизней больше одной | Уменьшение на 1 количества жизней, отображённых на дисплее |
| Поместить и извлечь мяч в ворота игрока один раз, когда количество жизней равняется единице | Отображение уведомления о необходимости восстановления жизней |
| Напряжение источника питания ниже 3,24 В | Отображение уведомление о низком уровне заряда аккумулятора |

## 7.7 Тестирование воспроизведения звукового сигнала

Методика тестирования воспроизведения звукового сигнала представлена в таблице 7.6.

Таблица 7.6 – Методика тестирования воспроизведения звукового сигнала

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Перевод выключателя на устройстве управления в положение «ON» | Воспроизведение короткого звукового сигнала |
| Поместить и извлечь мяч в ворота | Воспроизведение длинного звукового сигнала |
| Напряжение источника питания ниже 3,24 В | Воспроизведение трёх коротких звуковых сигнала |

## 7.8 Порядок проведения тестов для программной части

В таблице 7.7 представлен список тестов программной системы управления для симулятора компьютерной игры Пинг-Понг.

Таблица 7.7 – Список тестов программной системы управления для симулятора компьютерной игры Пинг-Понг

|  |  |
| --- | --- |
| Наименование тестирования | Раздел с описанием тестирования |
| 1 | 2 |
| 1 Подготовка тестовой среды | Раздел 7.9 |
| 2 Тестирование Wi-Fi соединения | Раздел 7.10 |
| 3 Тестирование подачи сигнала управления моделью игрока | Раздел 7.11 |

Продолжение таблицы 7.7

|  |  |
| --- | --- |
| 1 | 2 |
| 4 Тестирование управления и отображения количества жизней игрока | Раздел 7.12 |
| 5 Тестирование отображения изображения с камеры модели игрока | Раздел 7.13 |
| 6 Тестирование обнаружения мяча | Раздел 7.14 |

## 7.9 Подготовка тестовой среды

Подготовка включает создание среды, в которой будут проводиться испытания программной системы с целью оценки её работоспособности.

### 7.9.1 Необходимые для тестирования компоненты

Для проведения программы тестирования и испытаний программной системы управления потребуются следующие компоненты:

1. Игровая платформа. Модели игроков, управление которых осуществляется с помощью устройства управления, расположены на игровой платформе.
2. Игровой мяч. Мяч для игры поставляется в комплекте. Также подойдет любой мяч, который соответствует следующим критериям: достаточно легкий, чтобы при отталкивании моделью игрока смог достигнут ворот противника, иметь небольшой диаметр, чтобы свободно проходить в ворота игрока.
3. Блок питания. Блок питания необходим для подключения устройства к сети 220 В / 50 Гц. Оригинальный блок питания идет в комплекте с игровой платформой.

### 7.9.2 Включение игровой платформы

Включение игровой платформы осуществляется посредством подключения блока питания к разъему на корпусе и перевода выключателя в положение «ON». В результате должны включится светодиоды блоков отображения на стороне каждого игрока, что сообщает о корректном включении устройства и завершении инициализации компонентов системы.

### 7.9.3 Подготовка приложения

Необходимо скачать и установить \*.apk файл с приложением. Затем необходимо включить Wi-Fi на устройстве и запустить приложение.

## 7.10 Тестирование Wi-Fi соединения

Подключение к модели игрока осуществляется посредством Wi-Fi соединения. Методика тестирования Wi-Fi соединения представлена в таблице 7.8.

Таблица 7.8 – Методика тестирования Wi-Fi соединения

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Запуск приложения | Поиск и отображения списка доступных игроков |
| Выбрать доступного игрока для подключения | Осуществление подключения менее чем за 10 секунд и отображения уведомления об успешном подключении |
| Выбрать недоступного игрока для подключения | Отображение уведомления, что игрок недоступен |
| Отключение игровой платформы | Осуществление разрыва соединения и отображение уведомления о потере соединения с игроком |

## 7.11 Тестирование подачи сигнала управления моделью игрока

Управление моделью игрока осуществляется посредством нажатия кнопок. Методика тестирования подачи сигнала управления моделью игрока представлена в таблице 7.9.

Таблица 7.9 – Методика тестирования подачи сигнала управления моделью игрока

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Нажатие и удержание кнопки передвижения влево | Передвижение модели игрока влево, с максимальной скоростью |
| Нажатие и удержание кнопки передвижения вправо | Передвижение модели игрока вправо, с максимальной скоростью |
| Однократное нажатие кнопки отталкивания мяча | Активация механизма отталкивания мяча и возврат его в исходное положение |
| Удержание кнопки отталкивания мяча | Многократная активация механизма отталкивания мяча и возврат его в исходное положение |

## 7.12 Тестирование управления и отображения количества жизней игрока

Управления количеством жизней игрока осуществляется посредством нажатия кнопки. Отображение количества жизней осуществляется посредством интерфейса приложения Методика управления и отображения количества жизней игрока представлена в таблице 7.10.

Таблица 7.10 – Методика тестирования управления и отображения количества жизней игрока

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Подключиться к модели игрока | Отображение в пользовательском интерфейсе текущего количества жизней игрока |
| Поместить и извлечь мяч в ворота игрока один раз, когда количество жизней больше одной | Уменьшение на 1 количества жизней, отображённых в пользовательском интерфейсе |
| Поместить и извлечь мяч в ворота игрока один раз, когда количество жизней равняется единице | Отображение уведомления о необходимости восстановления жизней |
| Нажать кнопку восстановления жизней игрока | Отображение в пользовательском интерфейсе текущего количества 11 жизней, включение всех светодиодов в блоке отображения игровой платформы |

## 7.13 Тестирование отображения изображения с камеры модели игрока

Методика тестирования отображения изображение с камеры модели игрока представлена в таблице 7.11.

Таблица 7.11 – Методика тестирования отображения изображение с камеры модели игрока

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Запуск приложения | Отображение видеопотока данных с камеры модели игрока |

## 7.14 Тестирование обнаружения мяча

Методика тестирования обнаружения мяча представлена в таблице 7.12.

Таблица 7.12 – Методика тестирования обнаружения мяча

|  |  |
| --- | --- |
| Содержание теста | Ожидаемый результат |
| Поместить мяч так, чтобы отобразился в кадре камеры модели игрока | Отрисовка окантовки вокруг мяча |

# 8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Данный раздел проекта представляет собой руководство пользователя для аппаратно-программной системы управления в симуляторе компьютерной игры Пинг-Понг. Здесь представляется подробное описание элементов управления, указываются требования к системе, а также описывается процесс предварительной настройки устройства и приложения.

## 8.1 Руководство пользователя аппаратной части

### 8.1.1 Включение устройства

Включение устройства осуществляется с помощью переключателя расположенного на правой грани устройства. Для того, чтобы включить устройство необходимо перевести переключатель в положение «ON». Расположение переключателя устройства управления представлено на рисунке 8.1.



Рисунок 8.1 – Переключатель включения устройства управления

После включения устройства управления начинается процесс инициализации, и на дисплее должна отобразиться надпись, сигнализирующая о том, что происходит процесс инициализации. Если же дисплей устройства не загорелся, значит устройство разряжено и его необходимо зарядить. Процесс зарядки описан в подразделе 8.1.2. Отображение надписи на дисплее во время процесса инициализации устройства управления представлено на рисунке 8.2.

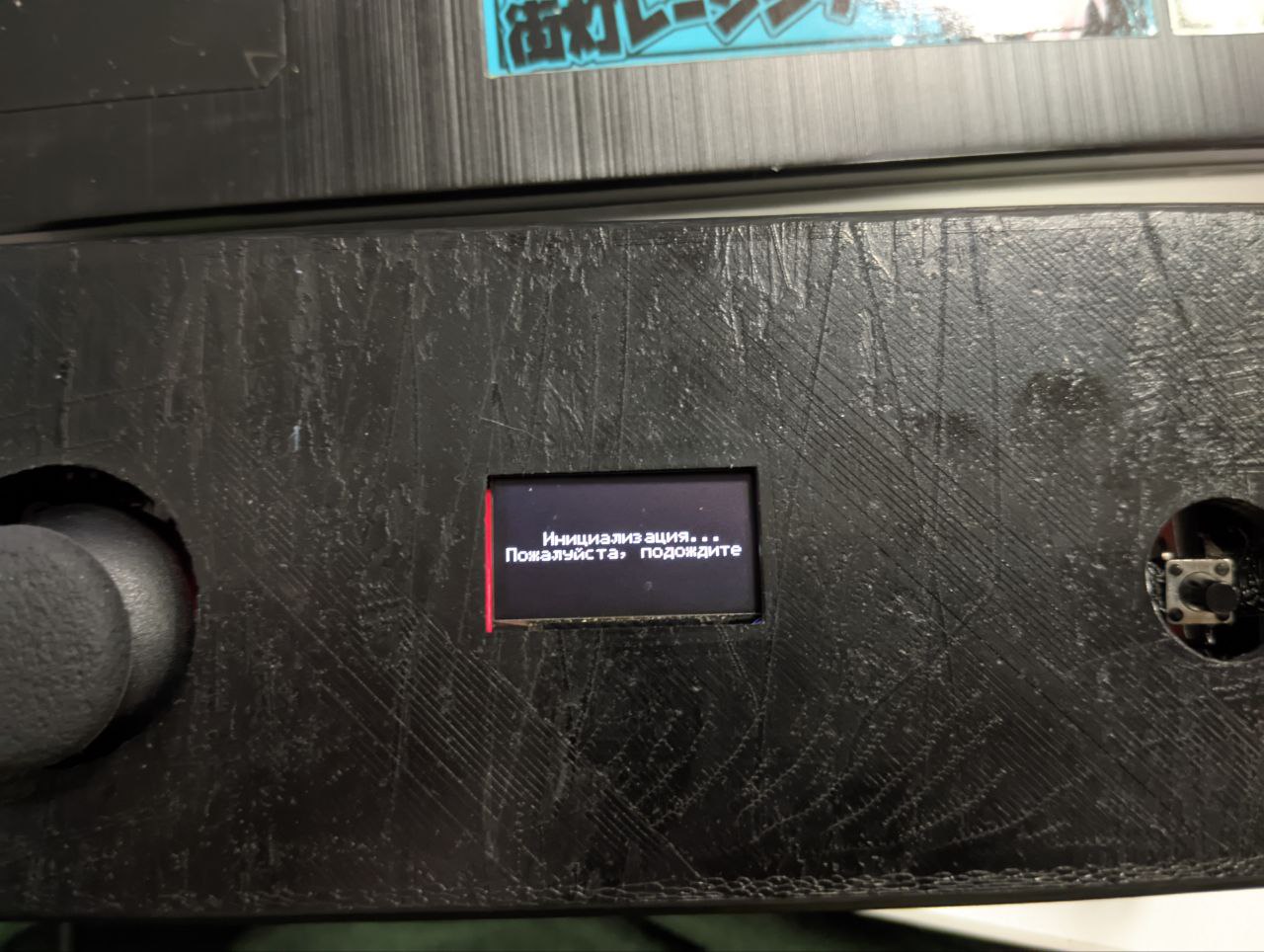


Рисунок 8.2 – Отображение надписи на дисплее во время инициализации

После завершения процесса инициализации на дисплее должна отобразиться надпись, сигнализирующая о том, что инициализация прошла успешна, и должен воспроизвестись короткий звуковой сигнал. Отображение надписи на дисплее во время процесса инициализации устройства управления представлено на рисунке 8.3.



Рисунок 8.3 – Отображение надписи на дисплее после завершения инициализации

После успешной инициализации устройства управления на дисплее отобразится главный экран. Отображение главного экрана представлено на рисунке 8.4.



Рисунок 8.4 – Отображение главного экрана

## 8.1.2 Зарядка устройства

Для зарядки на правой грани устройства рядом с переключатель имеется разъём зарядки типа USB Type-C. Подключите провод питания к устройству и убедитесь, что устройство начало заряжаться. На нижней грани справа имеется световая индикация зарядки, где индикация красным цветом означает, что устройство заряжается, индикация синим цвет сигнализирует, что устройство полностью заряжено. Расположение индикации на устройстве управления представлено на рисунке 8.5.



Рисунок 8.5 – Расположение индикации заряда аккумулятора

При достижении уровня заряда аккумулятора 20 и ниже процентов на дисплее отобразится уведомление, что уровень заряда аккумулятора низкий, и воспроизведутся три коротких звуковых сигнала. Отображение уведомления о низком уровне заряда аккумулятора представлено на рисунке 8.6.



Рисунок 8.6 – Отображение уведомления о низком уровне заряда аккумулятора

Отображение заряда аккумулятора производится в правом верхнем углу главного экрана. Отображение уровня заряда аккумулятора представлено на рисунке 8.7.



Рисунок 8.7 – Отображение уровень заряда аккумулятора

### 8.1.3 Управление моделью игрока

В устройстве управления представлено два возможных варианта управления перемещением модели игрока:

* с помощью джойстика;
* с помощью набора из двух кнопок.

Джойстик расположен на верхней грани пульта. Расположение джойстика на устройстве управления представлено на рисунке 8.8.



Рисунок 8.8 – Джойстик устройства управления

При наклоне джойстика влево подаётся сигнал на игровую платформу и модель игрока перемещается влево, при наклоне джойстика вправо соответственно модель игрока перемещается вправо. В зависимости от угла наклона стика изменяется скорость перемещения модели игрока.

Набор кнопок, состоящий из кнопки вправо и кнопки влево, расположены на по бокам задней грани устройства. На рисунке 8.9 представлено расположение кнопок, отвечающие за перемещение.



Рисунок 8.9 – Расположение кнопок перемещения на устройстве

В случае кнопок при нажатии кнопки влево модель игрока перемещается влево, при нажатии кнопки вправо соответственно вправо, при этом скорость модели игрока не регулируется и устанавливается максимальной.

### 8.1.4 Управление отталкиванием мяча

За подачу сигнала отталкивания мяча отвечает кнопка, расположенная на верхней грани устройства управления. Расположение кнопки отталкивания мяча представлено на рисунке 8.10.



Рисунок 8.10 – Расположение кнопки отталкивания мяча

При нажатии кнопки отталкивания мяча отправляется сигнал на игровую платформу, где по соответствующему сигналу активируется механизм отталкивания мяча на модели игрока.

### 8.1.5 Управление жизнями игрока

По умолчанию у каждого игрока в начале игры 11 жизней. Количество жизней отображаются на главном экране. При попадании мяча в ворота с игровой платформы подаётся сигнал на устройство управления об обновлении количества жизней. Отображение количества жизней на главном экране представлено на рисунке 8.11.



Рисунок 8.11 – Отображение количества жизней на главном экране

При достижении количества жизней равного нулю на дисплее отобразится уведомление о том, что осталось ноль жизней и пользователю необходимо будет восстановить жизни для продолжения игрового процесса. Отображение уведомления о необходимости восстановить жизни представлено на рисунке 8.12.



Рисунок 8.12 – Отображение уведомления о необходимости восстановить жизни

## 8.2 Руководство пользователя программной части

### 8.2.1 Начало работы

Начало работы приложения представляет собой установку соединения с моделью игрока. При запуске приложения необходимо, чтобы Wi-Fi на устройстве был включён. Если при запуске приложения Wi-Fi на устройстве будет отключён, то отобразится уведомление о необходимости включить Wi-Fi на устройстве. Отображение уведомления о необходимости включить Wi-Fi представлено на рисунке 8.13.

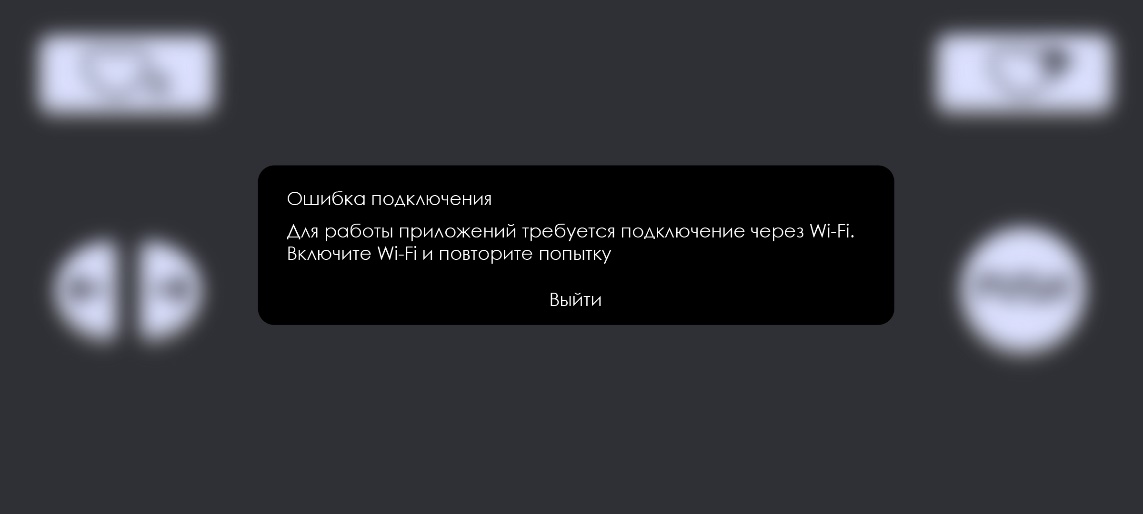


Рисунок 8.13 – Отображение уведомления о необходимости включить Wi-Fi

Если при запуске приложения Wi-Fi включён, сразу начинается поиск по близости доступных для подключения моделей игроков. Если доступных для подключения моделей игроков не обнаружится, то отобразится окно, где будет предложено выйти из приложения или повторить попытку поиска. Отображение окна при необнаружении доступных моделей игроков представлено на рисунке 8.14.

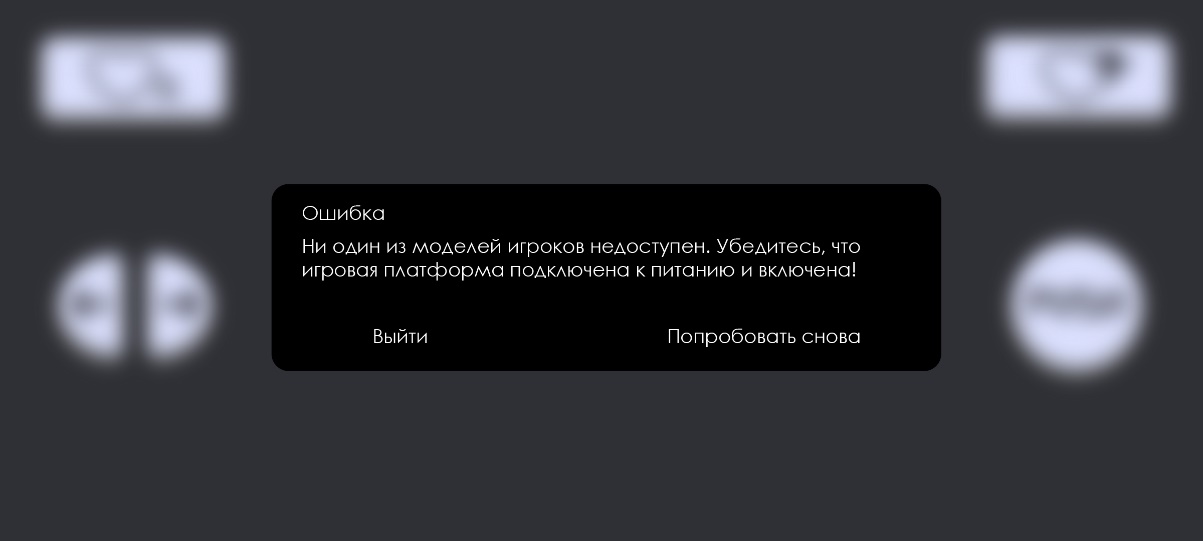


Рисунок 8.14 – Отображение окна при необнаружении доступных моделей игроков

При обнаружении доступных игроков отобразится окно выбора игроков для подключения. Отображение окна выбора игроков для подключения представлено на рисунке 8.15.



Рисунок 8.15 – Отображение окна выбора игроков для подключения

После выбора игрока для подключения необходимо нажать на соответствующую кнопку для подключения. После нажатия необходимо подтвердить подключение. После успешного подключения отобразится главное окно приложения. При неудачном подключении отобразится уведомление, что модель игрока недоступна и будет предложено выйти из приложения, либо повторить попытку. Отображение уведомления о неудачном подключении представлено на рисунке 8.16.

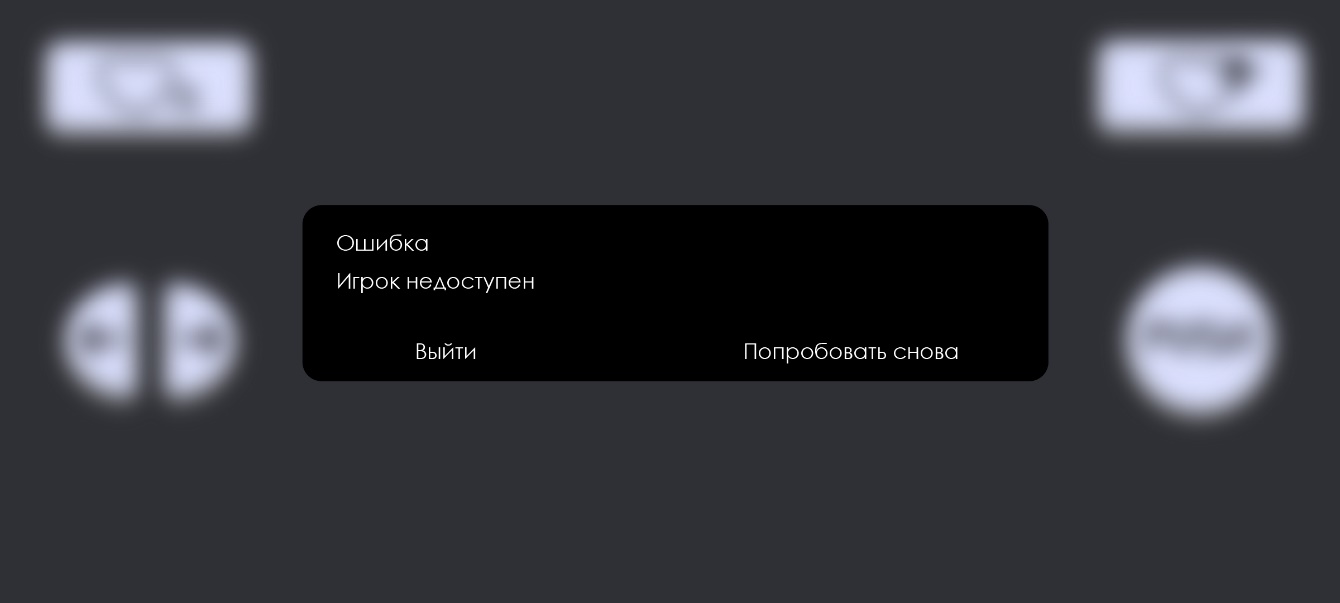


Рисунок 8.16 – Отображение уведомления о неудачном подключении

### 8.2.2 Главный экран приложения

После успешного подключения к модели игрока отобразится главный экран приложения. Отображение главного экрана представлено на рисунке 8.17.

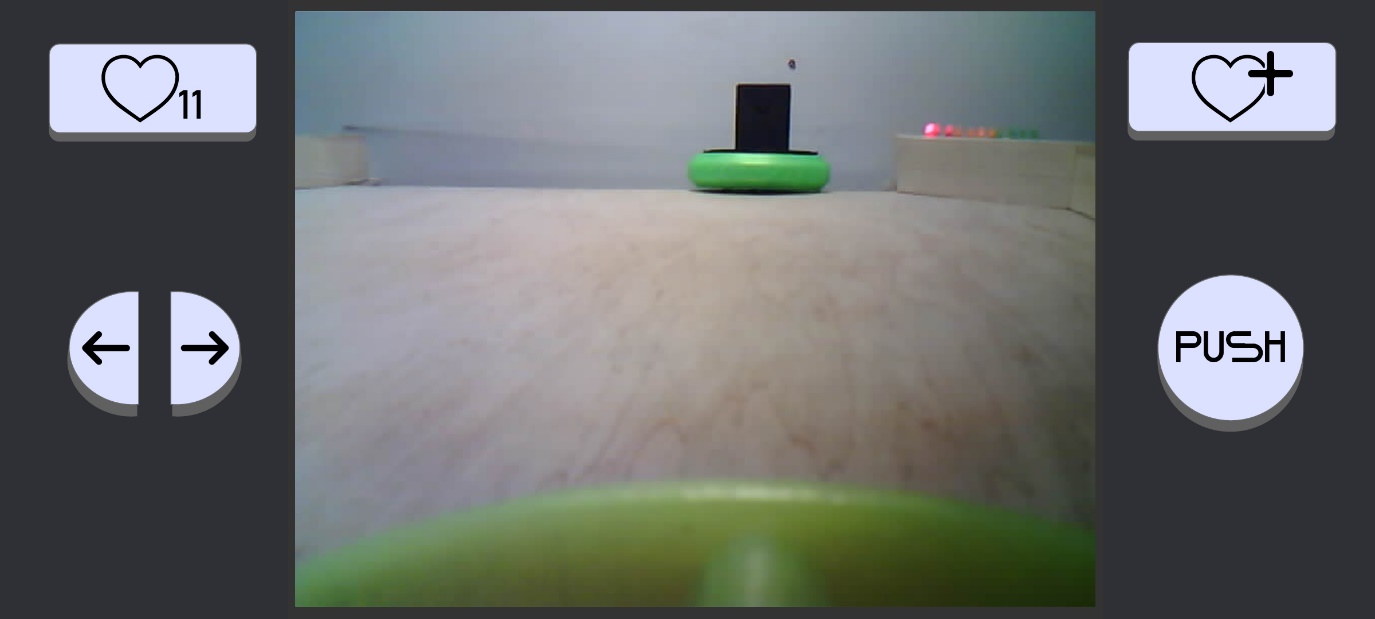


Рисунок 8.17 – Отображение главного экрана

На главном экране отображены вид с камеры модели игрока и элементы управления моделью игрока такие как кнопки перемещения влево и вправо и кнопка отталкивания мяча, также отображена кнопка восстановления жизней и количество жизней.

При попадании в кадр камеры модели игрока мяча мяч обнаруживается и выделяется окантовкой. Обнаружение мяча в кадре представлено на рисунке 8.18.



Рисунок 8.18 – Обнаружение в кадре мяча

При потере соединения с моделью игрок отобразится уведомление, что соединение потеряно. Отображение уведомления о потере соединения представлено на рисунке 8.19.

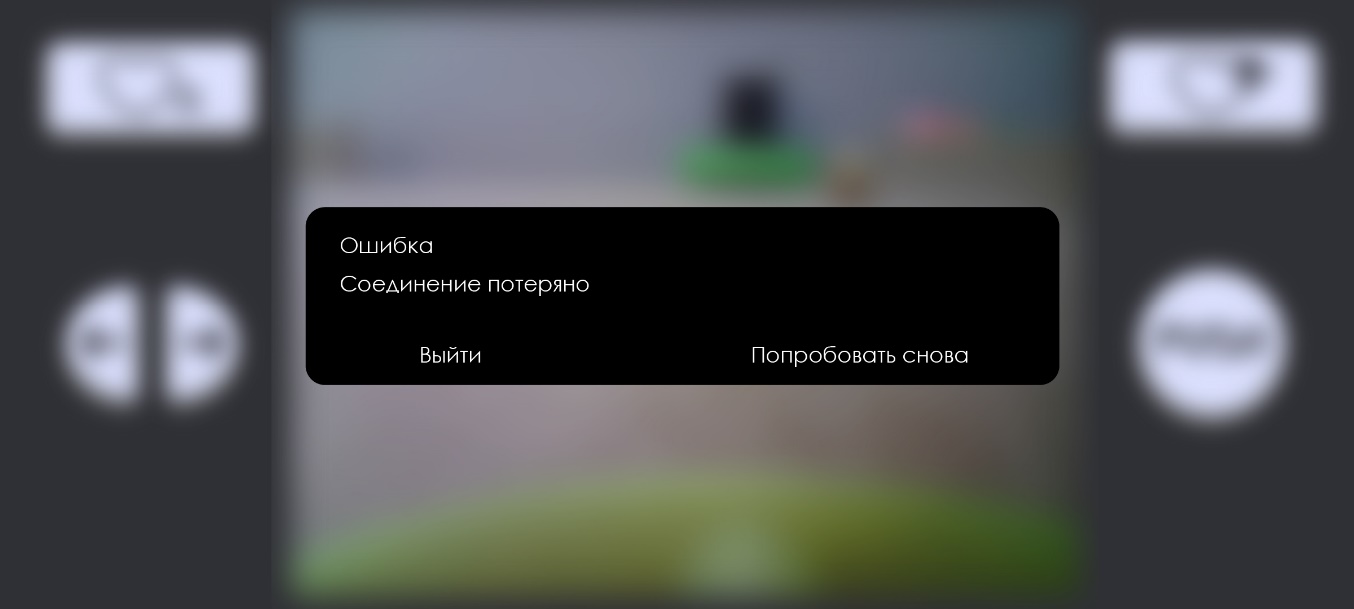


Рисунок 8.19 – Отображение уведомления о потере соединения

При достижении количества жизней равного нулю отобразится уведомления, что осталось ноль жизней и необходимо восстановить жизни. По нажатию кнопки «Восстановить жизни» восстанавливается все жизни и игровой процесс продолжается. Отображение уведомления о необходимости восстановить жизни представлено на рисунке 8.20.

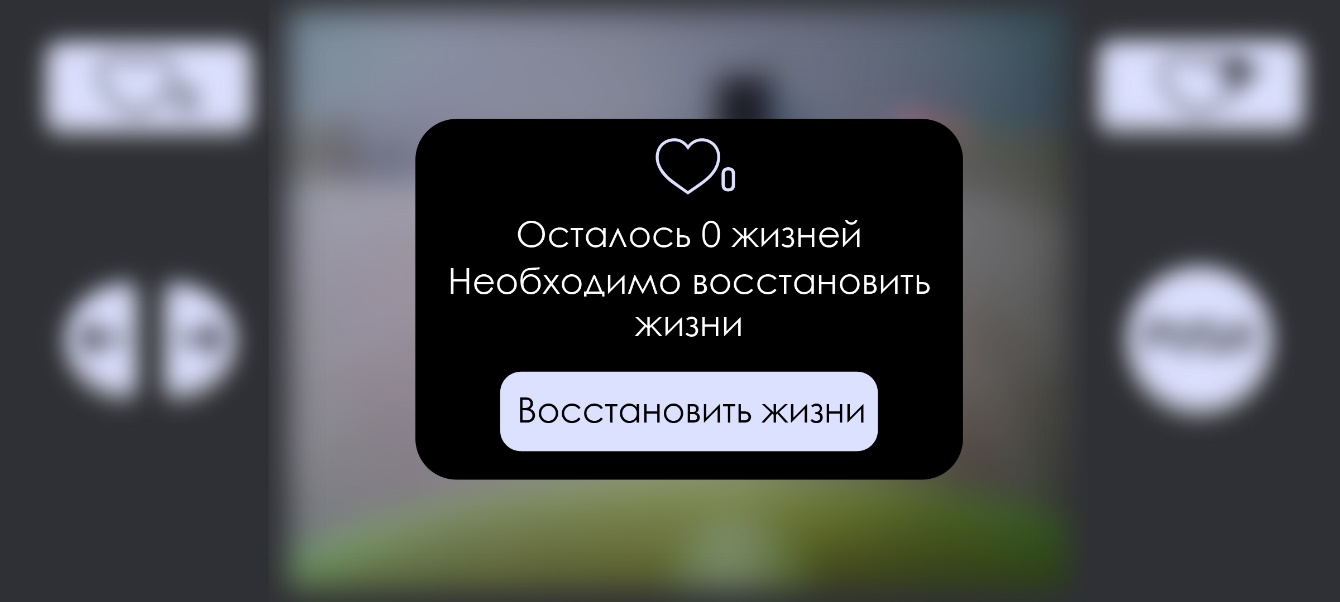


Рисунок 8.20 – Отображение уведомления о необходимости восстановить жизни

# 9 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И РЕАЛИЗАЦИИ АППАРАТНО-ПРОГРАММНой СИСТЕМы УПРАВЛЕНИЯ ДЛЯ СИМУЛЯТОРА КОМПЬЮТЕРНОЙ ИГРЫ «ПИНГ-ПОНГ»

## **9.1 Характеристика аппаратно-программной системы**

Разрабатываемая в данном дипломном проекте аппаратно-программная система управления для симулятора компьютерной игры «Пинг-Понг» состоит из аппаратной части и программного обеспечения, предназначенные для управления моделями игроков с подключением к игровой платформе по радиоканалу и Wi-Fi. Его пользователями могут быть частные лица.

Преимущество разрабатываемого аппаратно-программного комплекса –разнообразие в способе управления моделью игрока, начиная от устройства управления с удобным управлением благодаря беспроводной связи и заканчивая приложением с возможностью отображения видео с модели игрока, что позволяет наблюдать игровое поле в реальном времени, а также распознавать мяч в потоке видеоданных. Цена устройств управления, представленные на рынке, в разы превышает суммарную стоимость всех элементов разрабатываемого устройства.

## **9.2 Расчёт экономического эффекта от производства аппаратно-программной системы**

Для определения результата от вложения инвестиций в проектирование и производство аппаратно-программной системы необходимо определить отпускную цену аппаратно-программной системы на основе расчёта затрат на производство аппаратной части и разработку программной части.

Расчёт прямых затрат на материалы и комплектующие изделия для производства аппаратной части системы осуществляется в соответствии с представленной в конструкторской документации дипломного проекта номенклатурой, нормой расхода материалов, количеством комплектующих на изделие и рыночным ценам. Для производства данного комплекса были использованы материалы, представленные в таблице 9.1, и комплектующие, представленные в таблице 9.2.

Расчёт затрат по статье «Основные и вспомогательные материалы», в которую включается стоимость необходимых для изготовления изделия основных и вспомогательных материалов, осуществляется по формуле (9.1).

где Kтр – коэффициент транспортных расходов;

n – номенклатура применяемых материалов;

Hp*i* – норма расхода материала *i*-го вида на единицу изделия, нат. ед./шт.;

Цотп*i* – цена за единицу материала *i*-го вида, р.

Таблица 9.1 – Расчёт затрат на основные и вспомогательные материалы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование материала | Единица измерения | Норма расхода материала | Цена за единицу материала, р. | Сумма, р. |
| 1 Припой ПОС 61, 150 г | г | 0,28 | 26,21 | 7,60 |
| 2 Флюс ЛТИ-120, 25 мл | мл | 0,40 | 5,79 | 2,32 |
| 3 МГТФ 0.12 кв.мм, Провод монтажный, 1 м | м | 2,80 | 1,55 | 4,34 |
| 4 Текстолит односторонний 1.5 мм, 100х100 мм | мм | 0,60 | 8,00 | 4,80 |
| 5 Пластиковая нить ABS, 1000 г | г | 1,10 | 51,00 | 56,10 |
| 6 Разъем штыревой 2.54 мм, 40 шт | шт | 3,80 | 0,69 | 2,62 |
| Итого |  | | | 77,78 |
| Всего с учетом транспортных расходов (1,1)(Pм) | | | | 85,59 |

Расчёт затрат по статье «Покупные комплектующие изделия, полуфабрикаты», осуществляется по формуле (9.2).

где Kтр – коэффициент транспортных расходов;

m – номенклатура применяемых комплектующих;

*Ni* – количество комплектующих *i*-го вида на единицу изделия, нат. ед./шт.;

Цотп*i* – цена за единицу комплектующего *i*-го вида, р.

Таблица 9.2 – Расчёт затрат на комплектующие изделия

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование комплектующего | Количество на изделие, шт. | Цена за единицу, р. | Сумма, р. |
| 1 | 2 | 3 | 4 |
| 1 Микроконтроллер ATmega328P | 2 | 16,86 | 33,72 |
| 2 Модуль беспроводной связи NRF24L01 | 2 | 10,20 | 20,40 |
| 3 Микросхема HW-373 v1.1 | 2 | 1,62 | 3,24 |

Продолжение таблицы 9.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 4 OLED-дисплей ROHS TZT 0,96inch | 2 | 11,18 | 22,36 |
| 5 Кнопка тактовая KLS7-TS6601-5.0-180 | 8 | 0,10 | 0,80 |
| 6 Регулируемый джойстик KY-023 | 2 | 9,00 | 18,00 |
| 7 Аккумулятор LI-ION 18650 3.7 В | 2 | 26,00 | 56,00 |
| 8 Пьезодинамик | 2 | 0,81 | 1,62 |
| 9 Линейный стабилизатор напряжения 662K 3.3В | 2 | 0,50 | 1,00 |
| 10 DC-DC преобразователь повышающий 5В | 2 | 2,74 | 5,48 |
| Итого |  |  | 162,62 |
| Всего с учётом транспортных расходов (1,1)(Pк) | | | 178,88 |

Расчёт общей суммы прямых затрат на производство аппаратной части представлен в таблице 9.3.

Таблица 9.3 – Расчёт общей суммы прямых затрат на производство аппаратной части

|  |  |
| --- | --- |
| Показатель | Сумма, р. |
| 1 Сырье и материалы | 85,59 |
| 2 Покупные комплектующие изделия | 178,88 |
| Всего прямые затраты на производство аппаратной части | 264,47 |

Расчёт затрат на основную заработную плату разработчиков программной части комплекса представлен в таблице 9.4.

При расчёте заработной платы используется среднемесячная заработная плата в Республике Беларусь для сотрудников ИТ-отрасли. Премия не начисляется.

Основная зарплата определяется по формуле (9.3).

где – коэффициент премий и иных стимулирующих выплат;

*n* – категории исполнителей, занятых разработкой;

– часовой оклад плата исполнителя *i*-й категории, р.;

– трудоёмкость работ, выполняемых исполнителем *i*-й категории, ч.

Часовая заработная плата каждого исполнителя определяется путём деления его месячной заработной платы (оклад плюс надбавки) на количество рабочих часов в месяце (расчётная норма рабочего времени на 2024 г. Для 5-дневной недели составляет 168 ч по данным Министерства труда и социальной защиты населения на момент проведения расчётов).

Таблица 9.4 – Расчёт затрат на основную заработную плату команды разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория разработчика | Месячный оклад, р. | Часовой оклад, р. | Трудоёмкость работ, ч | Итого, р. |
| 1 Инженер-программист | 2250,00 | 13,39 | 82 | 1097,98 |
| 2 Инженер‑системотехник | 1620,00 | 9,64 | 55 | 530,20 |
| 3 Инженер-отладчик | 1310,00 | 7,80 | 15 | 117,00 |
| Итого | | | | 1745,18 |
| Премия и стимулирующие выплаты | | | | 0,00 |
| Всего затраты на основную заработную плату разработчиков | | | | 1745,18 |

Дополнительная заработная плата определяется по формуле (9.4).

где Нд – норматив дополнительной зарплаты, 15%.

Расчёт дополнительной заработной платы:

Отчисления в фонд социальной защиты населения и обязательное страхование БелГосстрах (Зсз) определяется в соответствии с действующим законодательством по формуле (9.5)

где Нсоц – норматив отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством по состоянию на апрель 2024 г. – 35%).

Расчёт отчислений в фонд социальной защиты населения и обязательного страхования БелГосстрах:

Расчёт общей суммы затрат на разработку программной части программно-управляемого комплекса представлен в таблице 9.5.

Таблица 9.5 – Расчёт затрат на разработку программного средства

|  |  |  |
| --- | --- | --- |
| Наименование статье затрат | Формула/таблица для расчёта | Сумма, р. |
| 1 Основная заработная плата разработчиков | Табл. 9.4 | 1745,18 |
| 2 Дополнительная заработная плата разработчиков | Формула (9.4) | 261,77 |
| 3 Отчисления на социальные нужды | Формула (9.5) | 702,43 |
| Затраты на разработку программной части | | 2709,38 |

Формирование отпускной цены аппаратно-программного комплекса осуществляется в соответствии с методикой, представленной в таблице 9.6.

Таблица 9.6 – Методика формирования отпускной цены аппаратно-программного комплекса

|  |  |  |
| --- | --- | --- |
| Показатель | Формула/таблица для расчёта | Сумма, р. |
| 1 Затраты на производство аппаратной части | Табл. 4.3 | 264,47 |
| 2 Затраты на разработку программной части | Табл. 4.5 | 2709,38 |
| 3 Сумма затрат на производство аппаратно-программного комплекса | 264,47 + 2709,38 | 2973,85 |
| 4 Накладные расходы | 2973,85∙ 0,56 | 1665,36 |
| 5 Расходы на реализацию | 2973,85∙ 0,02 | 59,47 |
| 6 Полная себестоимость | 2973,85 + 1665,36 + 59,47 | 4698,68 |
| 7 Плановая прибыль, включаемая в цену | 4698,68∙ 0,10 | 469,86 |
| 8 Отпускная цена | 4698,68 + 469,86 | 5168,54 |

Результатом производства аппаратно-программной системы является прирост чистой прибыли, полученный от их реализации и рассчитываемый по формуле (9.6).

где – прогнозируемый годовой объем производства и реализации, шт.;

– прибыль, включаемая в цену, р.;

– ставка налога на прибыль согласно действующему законодательству,

% (по состоянию на апрель 2024 г. – 20 %).

= 469,86∙ 300 ∙ (1 0,20) = 37588,80 р.

## **9.3 Расчёт инвестиций в проектирование и производство аппаратно-программного комплекса**

Инвестиции в производство аппаратно-программной системы включают в общем случае:

‒ инвестиции на его разработку;

‒ инвестиции в прирост основного капитала (затраты на приобретение необходимого для производства нового изделия оборудования, станков и т.п.);

‒ инвестиции в прирост собственного оборотного капитала (затраты на приобретение необходимых для производства нового изделия материалов, комплектующих, начатой, но незавершённой продукции и т.п.).

### 9.3.1 Расчёт инвестиций на разработку аппаратно-программного комплекса

Инвестиции рассчитываем (Ир) по затратам на разработку нового изделия. Для начала необходимо рассчитать заработную плату технологам и инженерам предприятия-производителя, ввиду того что именно они являются первой статьёй первичных затрат.

Расчёт заработной платы разработчиков нового изделия в таблице 9.7.

Таблица 9.7 – Расчёт заработной платы разработчиков нового изделия

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Формула/таблица для расчёта | Сумма, р. |
| 1 Основная заработная плата разработчиков Зо | Табл. 9.4 | 1745,18 |
| 2 Дополнительная заработная плата разработчиков | Формула (9.4) | 261,77 |
| 3 Отчисления на социальные нужды | Формула (9.5) | 702,43 |
| Инвестиции на разработку нового изделия (Ир) | | 2709,38 |

### 9.3.2 Расчёт инвестиций в прирост основного капитала

Инвестиции в прирост основного капитала не требуются, так как производство нового изделия планируется осуществлять на действующем оборудовании в связи с наличием на предприятии-производителе свободных производственных мощностей.

### 9.3.3 Расчёт инвестиций в прирост собственного оборотного капитала

Расчёт инвестиций в прирост собственного оборотного капитала осуществляется следующим образом:

1. Определяется годовая потребность в материалах по формуле (9.7).

где – прогнозируемый годовой объем производства и реализации, шт.;

– затраты на материалы на единицу изделия, р.

2. Определяется годовая потребность в комплектующих изделиях по формуле (9.8).

где – прогнозируемый годовой объем производства и реализации, шт.;

– затраты на комплектующие изделия на единицу продукции, р.

3. Вычисляются инвестиции в прирост собственного оборотного капитала в процентах от годовой потребности в материалах и комплектующих изделиях – *β* (исходя из среднего уровня по экономике: 20‒30 %) по формуле (9.9).

Рассчитаем инвестиции в прирост собственного оборотного капитала в процентах от годовой потребности в материалах и комплектующих изделиях:

Расчёт инвестиций в прирост собственного оборотного капитала представлен в таблице 9.9.

Таблица 9.9 – Расчёт инвестиций в прирост собственного оборотного капитала

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Формула/таблица для расчёта | Сумма, р. |
| 1 | 2 | 3 |
| 1 Годовая потребность в материалах | Формула (9.7) | 25677,00 |

Продолжение таблицы 9.9

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 2 Годовая потребность в комплектующих изделиях | Формула (9.8) | 53664,00 |
| Инвестиции в прирост собственного капитала | Формула (9.9) | 23802,30 |

## **9.4 Расчёт показателей экономической эффективности инвестиций в проектирование и производство аппаратно-программного комплекса**

Оценка экономической эффективности разработки и производства нового изделия зависит от результата сравнения инвестиций в производство нового изделия (инвестиции в разработку и прирост собственных оборотных средств) и полученного годового прироста чистой прибыли.

Сумма инвестиций меньше суммы годового экономического эффекта, то есть инвестиции окупятся менее чем за год, оценка экономической эффективности инвестиций в производство нового изделия осуществляется на основе расчёта рентабельность инвестиций (затрат) по формуле (9.10).

где  – прирост чистой прибыли от производства и реализации новых изделий, р.;

, – инвестиции в разработку нового изделия и прирост собственного оборотного капитала, р.

Сравнивая инвестиции в разработку изделия и прирост собственных оборотных средств, с приростом годовой чистой прибыли можно сделать вывод, что инвестиции окупаются в течении года.

Таким образом, рентабельность инвестиций в разработку и производство данного комплекса превышает ставки по долгосрочным депозитам в белорусских рублях на момент расчётов, что свидетельствует об экономической эффективности инвестиций. Следовательно, производство аппаратно-программной системы управления для симулятора компьютерной игры «Пинг-Понг» является экономически эффективным, инвестиции в его разработку целесообразны.

# ЗАКЛЮЧЕНИЕ

В ходе работы над дипломным проектом была произведена работа по проектированию и разработке аппаратно-программной системы управления для симулятора компьютерной игры «Пинг-Понг».

В результате проделанной работы были рассмотрены существующие аналоги, что позволило разработать прототип устройства управления на основе микроконтроллера ATmega328P. Также было разработано приложение под ОС Android для управления моделью игрока, для передачи и отображения видео с камеры модели игрока, а также для отслеживания положение мяча на видео.

Были произведены эксперименты с реализацией передачи данных между устройствами, выявлены особенности работы систем передачи данных по радиоканалу и через технологию Wi-Fi. Также были изучены разные подходы к реализации распознавания окружностей на изображении.

В результате были спроектированы и реализованы: устройство управления и приложение для управления моделью игрока и распознавания мяча с камеры модели игрока. С устройства управления можно подключаться к игровой платформе по радиоканалу и управлять перемещением модели игрока как с помощью джойстика, так и с помощью кнопок, также можно по нажатию кнопки активировать отталкивающий механизм на модели игрока. С приложения можно подключится к модели игрока по Wi-Fi и управлять его перемещением и активировать отталкивающий механизм. Также в приложении можно вывести изображение с камеры модели игрока с распознаванием мяча в реальном времени. Для обнаружения мяча использовался алгоритм Хафа.

Для разработки данного комплекса использовались такие инструменты, как интегрированная среда разработки для микроконтроллеров Arduino IDE, среда автоматизации проектирования электроники EasyEDA, система автоматизированного проектирования для дизайна Autodesk AutoCAD.

Проведя расчёт экономической эффективности, можно сделать вывод, что разработка и производство данного комплекса являются целесообразными, принесут выгоду как компании-разработчику, так и пользователю данной системы.

Основные преимущества системы:

* открытое программное обеспечение;
* распознавание мяча в видеопотоке в реальном времени;
* беспроводное управление моделями игроков;
* относительно небольшие размеры устройства управления;
* небольшая итоговая стоимость относительно аналогов.

Возможные улучшения системы:

* расширение списка поддерживаемых ОС приложением;
* разработка более эргономичного корпуса для устройства управления;
* возможность автоматической игры соперника для одиночных сеансов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Игровой манипулятор Nintendo Wii Remote [Электронный ресурс]. – Электронные данные. – Режим доступа: http://gamehouse.by/2933/ – Дата доступа: 26.03.2024.

[2] Контроллер движений PlayStation Move [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.playstation.com/ru-ua/accessories/playstation-move-motion-controller/ – Дата доступа: 26.03.2024.

[3] Joy-Con & controllers [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.nintendo.com/us/store/products/joy-con-set-l-r/ – Дата доступа: 26.03.2024.

[4] Steam Controller [Электронный ресурс]. – Электронные данные. – Режим доступа: https://store.steampowered.com/app/353370/Steam\_Controller/ – Дата доступа: 26.03.2024.

[5] Wi-Fi [Электронный ресурс]. – Электронные данные. – Режим доступа: https://byfly.by/wifi – Дата доступа: 26.03.2024.

[6] Bluetooth Technology Overview [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/ – Дата доступа: 26.03.2024.

[7] ESB [Электронный ресурс]. – Электронные данные. – Режим доступа: https://developer.nordicsemi.com/nRF\_Connect\_SDK/doc/latest/nrf/protocols/esb/index.html – Дата доступа: 26.03.2024.

[8] I2C [Электронный ресурс]. – Электронные данные. – Режим доступа: https://3d-diy.ru/wiki/arduino-moduli/interfeys-peredachi-dannykh-i2c/ – Дата доступа: 26.03.2024.

[9] SPI [Электронный ресурс]. – Электронные данные. – Режим доступа: https://3d-diy.ru/wiki/arduino-moduli/interfeys-peredachi-dannykh-spi/ – Дата доступа: 26.03.2024.

[10] OpenCV [Электронный ресурс]. – Электронные данные. – Режим доступа: https://opencv.org/about/ – Дата доступа: 26.03.2024.

Вычислительные машины, системы и сети: дипломное проектирование [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.bsuir.by/m/12\_100229\_1\_136308.pdf – Дата доступа: 10.04.2024.

Экономика проектных решений: методические указания по экономическому обоснованию дипломных проектов [Электронный ресурс]. –Электронные данные. – Режим доступа: https://www.bsuir.by/m/12\_100229\_1\_161144.pdf – Дата доступа: 10.04.2024.

# ПРИЛОЖЕНИЕ А

(*обязательное*)

**Программная часть.**

**Диаграмма классов**

# ПРИЛОЖЕНИЕ Б

(*обязательное*)

**Листинг кода прошивки аппаратной части**

Файл Code\_Controller.ino

001. #include <SPI.h>

002. #include "RF24.h"

003. #include "I2Cdev.h"

004. #include "MPU6050.h"

005. #include <GyverOLED.h>

006. #include <Wire.h>

007.

008. const uint8\_t arrow\_128x29[] PROGMEM = {

009. 0xFF, 0xFF, 0xFF, 0x3F, 0x7F, 0xFF, 0xFF, 0xEF, 0xCF, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xE0, 0xC0, 0x80, 0x80, 0x08, 0x18, 0x78, 0xF8, 0x18, 0x30, 0x70, 0xE0, 0x80, 0xE0, 0x70, 0x30, 0x18, 0x18, 0x08, 0x08, 0x18, 0x18, 0x30, 0x70, 0xE0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x8F, 0xCF, 0xEF, 0xFF, 0xFF, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF,

010. 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3E, 0x7C, 0xF8, 0xF0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xE1, 0x80, 0x00, 0x3F, 0x3F, 0x60, 0x60, 0x5F, 0x70, 0xE0, 0xC0, 0x81, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xE1, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0x7C, 0x3E, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF,

011. 0x0F, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3E, 0x7C, 0xF8, 0xF0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x08, 0x18, 0x30, 0x60, 0xC0, 0xC0, 0x80, 0x00, 0x80, 0xC0, 0xC0, 0x60, 0x30, 0x18, 0x0C, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0x7C, 0x3E, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x0F, 0x0F,

012. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x07, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x07, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

013. };

014.

015. const uint8\_t check\_45x45[] PROGMEM = {

016. 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0x78, 0x3C, 0x1C, 0x1E, 0x1E, 0x0F, 0x0F, 0x0F, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x0F, 0x0F, 0x0F, 0x1E, 0x1E, 0x3C, 0x3C, 0x78, 0xF8, 0xF0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

017. 0x80, 0xE0, 0xF8, 0xFE, 0x7F, 0x1F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x7F, 0xFE, 0xF8, 0xE0, 0x80, 0x00,

018. 0xFF, 0xFF, 0xFF, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xE0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0x7C, 0x3E, 0x1F, 0x0F, 0x07, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xFF, 0xFF, 0xFF, 0x00,

019. 0x1F, 0x7F, 0xFF, 0xFC, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3E, 0x1E, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xE0, 0xFC, 0xFF, 0x7F, 0x1F, 0x00,

020. 0x00, 0x00, 0x01, 0x07, 0x0F, 0x1F, 0x3E, 0x7C, 0xF8, 0xF0, 0xE0, 0xC0, 0xC0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0x7C, 0x3E, 0x1F, 0x0F, 0x07, 0x01, 0x00, 0x00, 0x00,

021. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x03, 0x07, 0x07, 0x0F, 0x0F, 0x0F, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0F, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

022. };

023.

024.

025. const uint8\_t image\_heart\_59x52[] PROGMEM = {

026. 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0xF8, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFE, 0xFE, 0xFC, 0xFC, 0xF8, 0xF8, 0xF0, 0xE0, 0xC0, 0x80, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFE, 0xFE, 0xFC, 0xFC, 0xF8, 0xF0, 0xE0, 0xE0, 0x80, 0x00, 0x00, 0x00,

027. 0xF8, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xF0,

028. 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x1F,

029. 0x00, 0x03, 0x0F, 0x1F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x1F, 0x07, 0x00, 0x00,

030. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

031. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x3F, 0x1F, 0x0F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

032. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x07, 0x0F, 0x07, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

033. };

034.

035. const uint8\_t image\_battery\_48x48[] PROGMEM = {

036. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x60, 0x60, 0x60, 0x60, 0x60, 0x60, 0x60, 0xE0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

037. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x06, 0x06, 0x06, 0x07, 0x06, 0x06, 0x06, 0x06, 0x06, 0x06, 0x06, 0x07, 0x07, 0x06, 0x06, 0xFE, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

038. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

039. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0x40, 0x60, 0x7F, 0x7F, 0x60, 0x40, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

040. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x9C, 0xFF, 0xC1, 0x80, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x80, 0xE3, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

041. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

042. };

043.

044.

045.

046. // Commands

047. // Define command codes

048. #define NULL\_COM\_PAR 0x00

049. #define MOVE\_LEFT 0x01

050. #define MOVE\_RIGHT 0x02

051. #define HIT\_BALL 0x03

052. #define RESURRECTION 0x04

053. #define UPDATE\_LIFES 0x05

054.

055. #define RESURRECTION\_DELAY 10000

056. #define UPDATE\_CHARGE\_DELAY 50000

057. unsigned int resurrection\_timer = 0;

058. unsigned int update\_charge\_timer = 0;

059.

060. bool show\_low\_level\_battery = false;

061.

062. // Structure representing a command and its parameters

063. struct Command {

064. byte code; // Command code (1 byte)

065. byte parameter; // Command parameter (1 byte)

066. };

067.

068. static Command command;

069.

070. #define left\_button 6

071. #define right\_button 7

072. #define solenoid\_button\_pin 5

073. #define y\_pin A1

074. #define buzzer\_pin 3

075. #define power\_control\_pin A3

076.

077. // NRF24L01

078. #define NRF\_CE 10

079. #define NRF\_CSN 9

080. #define PIPE 0x6060606060LL

081.

082. RF24 nrf(NRF\_CE, NRF\_CSN);

083.

084. // Display

085. GyverOLED<SSD1306\_128x64, OLED\_BUFFER> display; // Объявление класса, представляющего интерфейс для работы с дисплеем

086.

087. int sensorValue;

088. const float minVoltage = 3.0; // Минимальное напряжение (разряженный аккумулятор)

089. const float maxVoltage = 4.2; // Максимальное напряжение (полностью заряженный аккумулятор)

090.

091. // Global Variables

092. byte score=11;

093. byte new\_score=11;

094.

095. void update\_display\_lifes() {

096. display.setScale(3);

097. display.setCursor(80,3);

098. display.print(score);

099. display.drawBitmap(15, 6, image\_heart\_59x52, 59, 52);

100. display.update();

101. }

102.

103. void update\_percent\_charge() {

104. sensorValue = analogRead(power\_control\_pin );

105. float voltage = sensorValue \* (5.0 / 1023.0);

106. float percent = ((voltage - minVoltage) / (maxVoltage - minVoltage)) \* 100;

107. if(voltage < 3.24 && !show\_low\_level\_battery) {

108. display.clear();

109. display.setScale(1);

110. display.setCursor(1, 6);

111. display.print("Низкий уровень заряда");

112. display.setCursor(25, 7);

113. display.print("аккумулятора!!!");

114. display.drawBitmap(40, 0, image\_battery\_48x48, 48, 48);

115. display.update();

116. for(int i=0; i<3; i++) {

117. tone(buzzer\_pin, 500);

118. delay(200);

119. noTone(buzzer\_pin);

120. delay(200);

121. }

122. show\_low\_level\_battery = true;

123. }

124. if(voltage > 3.5)

125. show\_low\_level\_battery = false;

126. display.clear();

127. display.setScale(1);

128. display.setCursor(105, 0);

129. display.print((int)percent);

130. display.print("%");

131. display.rect(89, 0, 101, 7, OLED\_STROKE);

132. display.rect(87, 2, 88, 5, OLED\_FILL);

133.

134. if(percent <= 33) {

135. display.rect(97, 2, 98, 5, OLED\_FILL);

136. } else {

137. if(percent <= 66){

138. display.rect(97, 2, 98, 5, OLED\_FILL);

139. display.rect(94, 2, 95, 5, OLED\_FILL);

140. } else {

141. display.rect(97, 2, 98, 5, OLED\_FILL);

142. display.rect(94, 2, 95, 5, OLED\_FILL);

143. display.rect(91, 2, 92, 5, OLED\_FILL);

144. }

145. }

146.

147. update\_display\_lifes();

148. }

149.

150. void setup() {

151. Serial.begin(115200);

152.

153. pinMode(left\_button, INPUT);

154. pinMode(right\_button, INPUT);

155. pinMode(solenoid\_button\_pin, INPUT);

156. pinMode(y\_pin, INPUT);

157. pinMode(power\_control\_pin, INPUT);

158. pinMode(buzzer\_pin, OUTPUT);

159.

160. // I2C/TWI

161. Wire.begin(); // Инициализация шины I2C/TWI

162. display.init();

163.

164. display.clear();

165. display.setScale(1);

166. display.setCursor(20, 3);

167. display.print("Инициализация...");

168. display.setCursor(0, 4);

169. display.print("Пожалуйста, подождите!");

170. display.update();

171.

172. // NRF24L01

173. nrf.begin(); // активировать модуль

174. nrf.setAutoAck(1); // режим подтверждения приёма, 1 вкл 0 выкл

175. nrf.setRetries(0, 15); // (время между попыткой достучаться, число попыток)

176. nrf.enableAckPayload(); // разрешить отсылку данных в ответ на входящий сигнал

177. nrf.setPayloadSize(sizeof(Command)); // размер пакета, в байтах

178.

179. // Установка адреса приемника и передатчика

180. nrf.openWritingPipe(PIPE); // устанавливаем адрес передатчика

181. nrf.openReadingPipe(1, PIPE); // устанавливаем адрес приемника, используя трубу 1

182.

183. nrf.setChannel(0x20); // выбираем канал (в котором нет шумов!)

184. nrf.setPALevel(RF24\_PA\_MAX); // уровень мощности передатчика. На выбор RF24\_PA\_MIN, RF24\_PA\_LOW, RF24\_PA\_HIGH, RF24\_PA\_MAX

185. nrf.setDataRate(RF24\_1MBPS); // скорость обмена. На выбор RF24\_2MBPS, RF24\_1MBPS, RF24\_250KBPS

186. // должна быть одинакова на приёмнике и передатчике!

187. nrf.powerUp(); // начать работу

188. nrf.startListening(); // переводим модуль в режим приема данных

189.

190. display.clear();

191. display.drawBitmap(40, 0, check\_45x45, 45, 45);

192. display.setScale(1);

193. display.setCursor(3, 6);

194. display.print("Инициализация прошла");

195. display.setCursor(43, 7);

196. display.print("успешно!");

197. display.update();

198.

199. tone(buzzer\_pin, 900);

200. delay(100);

201. noTone(buzzer\_pin);

202. delay(900);

203.

204. update\_percent\_charge();

205. }

206.

207. void nrf\_write(){

208. nrf.stopListening(); // Переключаемся в режим передачи

209. nrf.write(&command, sizeof(Command)); // Отправляем команду

210. delay(100); // Небольшая пауза для передачи данных

211. nrf.startListening();

212. }

213.

214. void loop() {

215. command.code = NULL\_COM\_PAR;

216. command.parameter = NULL\_COM\_PAR;

217.

218. update\_charge\_timer++;

219. if(update\_charge\_timer == UPDATE\_CHARGE\_DELAY) {

220. update\_percent\_charge();

221. update\_charge\_timer = 0;

222. }

223.

224. // Read NRF

225. if (nrf.available()) {

226. nrf.read(&command, sizeof(Command));

227. if(command.code == UPDATE\_LIFES) {

228. score = command.parameter;

229. if(score == 0) {

230. display.clear();

231. display.drawBitmap(0, 0, arrow\_128x29, 128, 29);

232. display.setScale(1);

233. display.setCursor(12, 4);

234. display.print("Осталось 0 жизней(");

235. display.setCursor(23, 5);

236. display.print("Зажмите кнопки");

237. display.setCursor(20, 6);

238. display.print("перемещения для");

239. display.setCursor(0, 7);

240. display.print("восстановления жизней");

241. display.update();

242. } else {

243. update\_display\_lifes();

244. }

245. tone(buzzer\_pin, 900);

246. delay(1500);

247. noTone(buzzer\_pin);

248. }

249. }

250.

251. // Joystic

252. unsigned int xValue = map(analogRead(y\_pin), 0, 1023, 0, 255);

253. if ((xValue < 100) || (xValue > 130)) {

254. command.code = xValue < 100 ? MOVE\_LEFT : MOVE\_RIGHT;

255. command.parameter = xValue;

256. nrf\_write();

257. command.code = NULL\_COM\_PAR;

258. command.parameter = NULL\_COM\_PAR;

259. }

260.

261. // Solenoid button

262. if (digitalRead(solenoid\_button\_pin) == HIGH){

263. command.code = HIT\_BALL;

264. command.parameter = NULL\_COM\_PAR;

265. nrf\_write();

266. }

267.

268. //Left and Right buttons

269. if(digitalRead(left\_button) == HIGH || digitalRead(right\_button) == HIGH){

270. //delay(100);

271. if(digitalRead(left\_button) == HIGH && digitalRead(right\_button) == HIGH){

272. resurrection\_timer++;

273. if(resurrection\_timer == RESURRECTION\_DELAY){

274. command.code = RESURRECTION;

275. command.parameter = NULL\_COM\_PAR;

276. resurrection\_timer = 0;

277. nrf\_write();

278. command.code = NULL\_COM\_PAR;

279. command.parameter = NULL\_COM\_PAR;

280. }

281. return;

282. } else {

283. resurrection\_timer = 0;

284. if(digitalRead(left\_button) == HIGH){

285. command.code = MOVE\_LEFT;

286. command.parameter = 0;

287. }

288. if(digitalRead(right\_button) == HIGH){

289. command.code = MOVE\_RIGHT;

290. command.parameter = 255;

291. }

292. if(command.code != NULL\_COM\_PAR) {

293. nrf\_write();

294. command.code = NULL\_COM\_PAR;

295. command.parameter = NULL\_COM\_PAR;

296. }

297. }

298. }

299. resurrection\_timer = 0;

300. }

# ПРИЛОЖЕНИЕ В

(*обязательное*)

**Листинг кода прошивки программной части**

Файл connectiontoplayer.h

01. #ifndef CONNECTIONTOPLAYER\_H

02. #define CONNECTIONTOPLAYER\_H

03.

04. #include <QObject>

05. #include <QGuiApplication>

06. #include <QNetworkInterface>

07. #include <QUrl>

08. #include <sendersignals.h>

09. #include <QProcess>

10. #include <QPermissions>

11. #include <QJniObject>

12.

13. class ConnectionToPlayer : public QObject

14. {

15. Q\_OBJECT

16. public:

17. explicit ConnectionToPlayer(QObject \*engine, QObject \*parent = nullptr);

18.

19. Q\_INVOKABLE void checkWiFiConnection();

20. Q\_INVOKABLE void connectToPlayer(QString playerBSSID);

21.

22. static ConnectionToPlayer \*m\_instance;

23.

24. static ConnectionToPlayer \*instance()

25. {

26. return m\_instance;

27. }

28.

29. void wifiScanComplete(const QString &result);

30. void setStatusConnection(const QString &result);

31.

32. signals:

33. void wifiErrorMsg();

34. void initManageConnection();

35.

36. private:

37. QVariantList avaliablePlayers;

38. QUrl ipAddressPlayer;

39. QObject \*frontEnd;

40. QJniObject javaWiFiApi;

41. };

42.

43. #endif // CONNECTIONTOPLAYER\_H

Файл connectiontoplayer.cpp

01. #include "connectiontoplayer.h"

02. #include <QPermissions>

03. #include <QJniObject>

04. #include <QJsonDocument>

05.

06. ConnectionToPlayer \*ConnectionToPlayer::m\_instance = nullptr;

07.

08. ConnectionToPlayer::ConnectionToPlayer(QObject \*engine, QObject \*parent)

09. : QObject{parent}

10. {

11. this->frontEnd = engine;

12. m\_instance = this;

13. javaWiFiApi = QJniObject("org/qtproject/example/pingpongmobile/WiFiApi");

14.

15. QJniObject activity = QNativeInterface::QAndroidApplication::context();

16. if (activity.isValid()) {

17. QJniObject window =

18. activity.callObjectMethod("getWindow", "()Landroid/view/Window;");

19.

20. if (window.isValid()) {

21. const int FLAG\_KEEP\_SCREEN\_ON = 128;

22. window.callMethod<void>("addFlags", "(I)V", FLAG\_KEEP\_SCREEN\_ON);

23. }

24. }

25. QJniEnvironment env;

26. if (env->ExceptionCheck()) {

27. env->ExceptionClear();

28. }

29. }

30.

31. void ConnectionToPlayer::checkWiFiConnection()

32. {

33. jint wifiState = QJniObject::callStaticMethod<jint>(

34. "org/qtproject/example/pingpongmobile/WiFiApi",

35. "checkEnableWifi",

36. "(Landroid/content/Context;)I",

37. QNativeInterface::QAndroidApplication::context());

38. if(!wifiState) {

39. QMetaObject::invokeMethod(this->frontEnd, "showErrorWiFi");

40. } else {

41. if(javaWiFiApi.isValid()) {

42. javaWiFiApi.callMethod<int>("checkConnection",

43. "(Landroid/content/Context;)I",

44. QNativeInterface::QAndroidApplication::context());

45. } else {

46. //PRINT CRITICAL ERROR

47. }

48. }

49. }

50.

51. void ConnectionToPlayer::connectToPlayer(QString playerBSSID)

52. {

53. QJniObject BSSID = QJniObject::fromString(playerBSSID);

54. jint status = javaWiFiApi.callMethod<jint>("connectToPlayer",

55. "(Landroid/content/Context;Ljava/lang/String;)I",

56. QNativeInterface::QAndroidApplication::context(), BSSID.object<jstring>());

57. }

58.

59. void ConnectionToPlayer::wifiScanComplete(const QString &result)

60. {

61. QStringList listBSSIDs = result.split(' ');

62.

63. for(QString BSSID : listBSSIDs) {

64. if(BSSID == "e0:5a:1b:d1:6d:41" && !avaliablePlayers.contains("e0:5a:1b:d1:6d:41"))

65. avaliablePlayers.append(BSSID);

66. if(BSSID == "08:3a:f2:aa:6d:09" && !avaliablePlayers.contains("08:3a:f2:aa:6d:09"))

67. avaliablePlayers.append(BSSID);

68. }

69.

70. if(avaliablePlayers.isEmpty())

71. QMetaObject::invokeMethod(this->frontEnd, "showError", Q\_ARG(QVariant, "Ни один из моделей игроков недоступен. Убедитесь, что игровая платформа подключена к питанию и включена!"));

72. else

73. QMetaObject::invokeMethod(this->frontEnd, "openListPlayers", Q\_ARG(QVariant, QVariant::fromValue(avaliablePlayers)));

74. }

75.

76. void ConnectionToPlayer::setStatusConnection(const QString &result)

77. {

78. if(result == "Success") {

79. emit initManageConnection();

80. } else {

81. QMetaObject::invokeMethod(this->frontEnd, "showError", Q\_ARG(QVariant, result));

82. }

83. }

Файл detectball.h

01. #ifndef DETECTBALL\_H

02. #define DETECTBALL\_H

03.

04. #include <QObject>

05. #include <opencv2/core/core.hpp>

06. #include <opencv2/highgui/highgui.hpp>

07. #include <opencv2/imgproc.hpp>

08. #include <QImage>

09. #include <vector>

10.

11. class DetectBall : public QObject

12. {

13. Q\_OBJECT

14. public:

15. explicit DetectBall(QObject \*parent = nullptr);

16.

17. QImage findBall(QByteArray inputImage);

18.

19. cv::Mat convertToMat(QByteArray inputImage);

20.

21. QImage convertToQImage(cv::Mat inputImage);

22.

23. signals:

24. };

25.

26. #endif // DETECTBALL\_H

27. #ifndef DETECTBALL\_H

28. #define DETECTBALL\_H

29.

30. #include <QObject>

31. #include <opencv2/core/core.hpp>

32. #include <opencv2/highgui/highgui.hpp>

33. #include <opencv2/imgproc.hpp>

34. #include <QImage>

35. #include <vector>

36.

37. class DetectBall : public QObject

38. {

39. Q\_OBJECT

40. public:

41. explicit DetectBall(QObject \*parent = nullptr);

42.

43. QImage findBall(QByteArray inputImage);

44.

45. cv::Mat convertToMat(QByteArray inputImage);

46.

47. QImage convertToQImage(cv::Mat inputImage);

48.

49. signals:

50. };

51.

52. #endif // DETECTBALL\_H

Файл detectball.cpp

01. #include "detectball.h"

02. #include <QDebug>

03.

04. DetectBall::DetectBall(QObject \*parent)

05. : QObject{parent}

06. {}

07.

08. QImage DetectBall::findBall(QByteArray input)

09. {

10. cv::Mat image = convertToMat(input);

11.

12. cv::Mat gray;

13. cv::cvtColor(image, gray, cv::COLOR\_RGB2GRAY);

14. medianBlur(gray, gray, 5);

15.

16. std::vector<cv::Vec3f> circles;

17.

18. cv::HoughCircles(gray, circles, cv::HOUGH\_GRADIENT, 1, 1000, 120, 30, 20, 110);

19.

20. for (size\_t i = 0; i < circles.size(); i++)

21. {

22. cv::Vec3i c = circles[i];

23. cv::Point center = cv::Point(c[0], c[1]);

24.

25. circle(image, center, 1, cv::Scalar(0, 100, 100), 3, cv::LINE\_AA);

26.

27. int radius = c[2];

28. circle(image, center, radius, cv::Scalar(255, 0, 255), 3, cv::LINE\_AA);

29. }

30.

31. return convertToQImage(image);

32. }

33.

34. cv::Mat DetectBall::convertToMat(QByteArray inputImage)

35. {

36. // cv::Mat outputImage = cv::Mat(inputImage.height(), inputImage.width(), CV\_8UC4,

37. // const\_cast<uchar\*>(inputImage.bits()),

38. // inputImage.bytesPerLine()).clone();

39. std::vector<uchar> data(inputImage.begin(), inputImage.end());

40. cv::Mat outputImage = cv::imdecode(data, cv::IMREAD\_COLOR);

41. return outputImage;

42. }

43.

44. QImage DetectBall::convertToQImage(cv::Mat inputImage)

45. {

46. //cv::Mat image;

47. //cv::cvtColor(inputImage, image, cv::COLOR\_BGR2RGB);

48. QImage outputImage = QImage(inputImage.data, inputImage.cols, inputImage.rows, inputImage.step, QImage::Format\_RGB888).rgbSwapped();

49.

50. return outputImage;

51. }

Файл gameinfo.h

01. #ifndef GAMEINFO\_H

02. #define GAMEINFO\_H

03.

04. #include <QObject>

05. #include <receiverdata.h>

06. #include <QThread>

07.

08. class GameInfo : public QObject

09. {

10. Q\_OBJECT

11. public:

12. explicit GameInfo(QObject \*engine, QObject \*parent = nullptr);

13.

14. int receiveProcess();

15.

16. public slots:

17. void initReceiver(int socket);

18.

19. signals:

20. void showError(QString errorMessage);

21.

22. private:

23. int countLifes;

24. ReceiverData receiver;

25. QObject \*frontEnd;

26. QThread \*recieveThread;

27. };

28.

29. #endif // GAMEINFO\_H

Файл gameinfo.cpp

01. #include "gameinfo.h"

02. #include <QGuiApplication>

03.

04. GameInfo::GameInfo(QObject \*engine, QObject \*parent)

05. : QObject{parent}

06. {

07. this->frontEnd = engine;

08. }

09.

10. int GameInfo::receiveProcess()

11. {

12. struct frameCommand{

13. unsigned char code, param;

14. } data;

15.

16. while(1) {

17. if(receiver.recieveData((unsigned char\*)&data) == SUCCESS) {

18.

19. QMetaObject::invokeMethod(this->frontEnd, "setCountLifes", Q\_ARG(QVariant, QVariant::fromValue(QString::number(static\_cast<int>(data.param)))));

20. }

21.

22. }

23. }

24.

25. void GameInfo::initReceiver(int socket)

26. {

27. this->receiver.setSocket(socket);

28. this->recieveThread = QThread::create(&GameInfo::receiveProcess, this);

29. this->recieveThread->start();

30. }

Файл manageplayer.h

01. #ifndef MANAGEPLAYER\_H

02. #define MANAGEPLAYER\_H

03.

04. #include <QObject>

05. #include <QDebug>

06. #include <sendersignals.h>

07.

08. class ManagePlayer : public QObject

09. {

10. Q\_OBJECT

11. public:

12. explicit ManagePlayer(QObject \*engine, QObject \*parent = nullptr);

13.

14. Q\_INVOKABLE void inputProcessor(unsigned char code);

15. unsigned char\* buildFrame(unsigned char code);

16.

17. public slots:

18. void initSender();

19.

20. signals:

21. void showError(QString errorMessage);

22. void startStreamSignal(QString URL);

23. void initGameInfoConnection(int socket);

24.

25. private:

26. unsigned char\* frame;

27. SenderSignals sender;

28. QObject \*frontEnd;

29. };

30.

31. #endif // MANAGEPLAYER\_H

Файл managerplayer.cpp

01. #include "manageplayer.h"

02. #include <QPermissions>

03. #include <QJniObject>

04. #include <QGuiApplication>

05. #include <QNetworkInterface>

06.

07. ManagePlayer::ManagePlayer(QObject \*engine, QObject \*parent)

08. : QObject{parent}

09. {

10. this->frontEnd = engine;

11. }

12.

13. void ManagePlayer::inputProcessor(unsigned char code)

14. {

15. this->frame = buildFrame(code);

16. if(sender.sendFrame(this->frame, 2) != SUCCESS) {

17. QMetaObject::invokeMethod(this->frontEnd, "showError", Q\_ARG(QVariant, "Ошибка отправки команды"));

18. }

19. }

20.

21. unsigned char \*ManagePlayer::buildFrame(unsigned char code)

22. {

23. unsigned char\* data;

24. data = new unsigned char[2];

25.

26. switch(code) {

27. case 0x01:

28. data[0] = 0x01;

29. data[1] = 255;

30. break;

31. case 0x02:

32. data[0] = 0x02;

33. data[1] = 255;

34. break;

35. case 0x03:

36. data[0] = 0x03;

37. data[1] = 0;

38. break;

39. case 0x04:

40. data[0] = 0x04;

41. data[1] = 0;

42. break;

43. }

44.

45. return data;

46. }

47.

48. void ManagePlayer::initSender()

49. {

50. if(int status = sender.init("192.168.4.1", 10000) != SUCCESS) {

51. QMetaObject::invokeMethod(this->frontEnd, "showError", Q\_ARG(QVariant, "Ошибка соединения с игроком"));

52. } else {

53. QMetaObject::invokeMethod(this->frontEnd, "endConnection");

54. }

55. emit initGameInfoConnection(this->sender.getSocket());

56. emit startStreamSignal("http://192.168.4.1/capture");

57. }

Файл main.cpp

001. #include <QGuiApplication>

002. #include <QQmlApplicationEngine>

003. #include <QQmlContext>

004. #include <opencv2/core/core.hpp>

005. #include <opencv2/core/mat.hpp>

006. #include <opencv2/videoio.hpp>

007. #include "videocamcapture.h"

008. #include "connectiontoplayer.h"

009. #include "manageplayer.h"

010. #include "sendersignals.h"

011. #include "gameinfo.h"

012.

013.

014. #ifdef Q\_OS\_ANDROID

015. //Register special Java callback functions immediately once the app loads because they must be executed if the app is started via external Android intents

016.

017. static void wifiScanComplete(JNIEnv \*env, jobject thiz, jstring text)

018. {

019. ConnectionToPlayer::instance()->wifiScanComplete(env->GetStringUTFChars(text, nullptr));

020. }

021.

022. static void statusConnection(JNIEnv \*env, jobject thiz, jstring text)

023. {

024. ConnectionToPlayer::instance()->setStatusConnection(env->GetStringUTFChars(text, nullptr));

025. }

026.

027.

028. static JNINativeMethod methods[] = {

029. { "wifiScanComplete", // const char\* function name;

030. "(Ljava/lang/String;)V", // const char\* function signature

031. (void \*)wifiScanComplete // function pointer

032. },

033. { "statusConnection", // const char\* function name;

034. "(Ljava/lang/String;)V", // const char\* function signature

035. (void \*)statusConnection // function pointer

036. }

037. };

038.

039.

040. JNIEXPORT jint JNI\_OnLoad(JavaVM\* vm, void\* /\*reserved\*/)

041. {

042. JNIEnv\* env;

043. // get the JNIEnv pointer.

044. if (vm->GetEnv(reinterpret\_cast<void\*\*>(&env), JNI\_VERSION\_1\_6) != JNI\_OK)

045. return JNI\_ERR;

046.

047. // step 3

048. // search for Java class which declares the native methods

049. jclass javaClass = env->FindClass("org/qtproject/example/pingpongmobile/WiFiApi");

050. if (!javaClass)

051. return JNI\_ERR;

052.

053. // step 4

054. // register our native methods

055. if (env->RegisterNatives(javaClass, methods, sizeof(methods) / sizeof(methods[0])) < 0)

056. return JNI\_ERR;

057.

058. return JNI\_VERSION\_1\_6;

059. }

060. #endif

061.

062.

063. int main(int argc, char \*argv[])

064. {

065. #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)

066. QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);

067. #endif

068. QGuiApplication app(argc, argv);

069.

070. QQmlApplicationEngine engine;

071.

072.

073. const QUrl url(QStringLiteral("qrc:/main.qml"));

074. QObject::connect(

075. &engine,

076. &QQmlApplicationEngine::objectCreated,

077. &app,

078. [url](QObject \*obj, const QUrl &objUrl) {

079. if (!obj && url == objUrl)

080. QCoreApplication::exit(-1);

081. },

082. Qt::QueuedConnection);

083. engine.load(url);

084.

085. VideoCamCapture \*streamProvider(new VideoCamCapture);

086. ManagePlayer \*inputPlayer(new ManagePlayer(engine.rootObjects().at(0)));

087. ConnectionToPlayer \*connectionManager(new ConnectionToPlayer(engine.rootObjects().at(0)));

088. GameInfo \*gameInfo(new GameInfo(engine.rootObjects().at(0)));

089.

090. QObject::connect(connectionManager, SIGNAL(initManageConnection()), inputPlayer, SLOT(initSender()));

091. QObject::connect(inputPlayer, SIGNAL(initGameInfoConnection(int)), gameInfo, SLOT(initReceiver(int)));

092. QObject::connect(inputPlayer, SIGNAL(startStreamSignal(QString)), streamProvider, SLOT(startStream(QString)));

093.

094. engine.rootContext()->setContextProperty("streamProvider",streamProvider);

095. engine.rootContext()->setContextProperty("inputPlayer",inputPlayer);

096. engine.rootContext()->setContextProperty("connectionManager", connectionManager);

097. engine.rootContext()->setContextProperty("gameInfo", gameInfo);

098. engine.addImageProvider("live", streamProvider);

099.

100. connectionManager->checkWiFiConnection();

101.

102. return app.exec();

103. }

Файл videocamcapture.h

01. #ifndef VIDEOCAMCAPTURE\_H

02. #define VIDEOCAMCAPTURE\_H

03.

04. #include <opencv2/videoio.hpp>

05. #include <QTimer>

06. #include <QObject>

07. #include <QImage>

08. #include <QQuickImageProvider>

09. #include <QNetworkRequest>

10. #include <QNetworkAccessManager>

11. #include <QNetworkReply>

12. #include <QNetworkRequest>

13. #include <QtMultimedia/QVideoFrame>

14. #include <QUrl>

15. #include <detectball.h>

16. #include <QThread>

17. #include <QMutex>

18.

19. class VideoCamCapture: public QQuickImageProvider

20. {

21. Q\_OBJECT

22.

23. public:

24. VideoCamCapture(QObject \*parent = nullptr);

25.

26. QImage requestImage(const QString &id, QSize \*size, const QSize &requestedSize) override;

27.

28. void captureFrame();

29. void processThreadFun();

30.

31. signals:

32. void imageChanged();

33.

34. public slots:

35. void startStream(QString URL);

36.

37. private slots:

38. void replyFinished(QNetworkReply \*reply);

39.

40. private:

41. QTimer timeUpdateFrame;

42. QImage image;

43. QNetworkAccessManager m\_networkManager;

44. QNetworkReply \*m\_reply;

45. QUrl ipCamera;

46. QThread \*processThread;

47. QList<QByteArray> frames;

48. DetectBall detector;

49. QMutex mutex;

50. };

51.

52. #endif // VIDEOCAMCAPTURE\_H

Файл videcamcapture.cpp

01. #include "videocamcapture.h"

02. #include <QtMultimedia/QMediaPlayer>

03.

04. VideoCamCapture::VideoCamCapture(QObject \*parent) : QQuickImageProvider(QQuickImageProvider::Image) {

05. image = QImage(200,200,QImage::Format\_RGB888);

06. image.fill(QColor("black"));

07.

08. connect(&timeUpdateFrame,&QTimer::timeout,this,&VideoCamCapture::captureFrame);

09. connect(&m\_networkManager, SIGNAL(finished(QNetworkReply\*)), this, SLOT(replyFinished(QNetworkReply\*)));

10. }

11.

12. QImage VideoCamCapture::requestImage(const QString &id, QSize \*size, const QSize &requestedSize)

13. {

14. Q\_UNUSED(id);

15.

16. if(size){

17. \*size = image.size();

18. }

19.

20. if(requestedSize.width() > 0 && requestedSize.height() > 0) {

21. image = image.scaled(requestedSize.width(), requestedSize.height(), Qt::KeepAspectRatio);

22. }

23.

24. return image;

25. }

26.

27. void VideoCamCapture::startStream(QString URL) {

28. this->processThread = QThread::create(&VideoCamCapture::processThreadFun, this);

29. this->processThread->start();

30.

31. ipCamera.setUrl(URL);

32. captureFrame();

33. this->timeUpdateFrame.start(1000/60);

34. }

35.

36. void VideoCamCapture::captureFrame()

37. {

38. QNetworkRequest request(ipCamera);

39. m\_networkManager.get(request);

40. }

41.

42. void VideoCamCapture::processThreadFun()

43. {

44. while(1) {

45. if(!frames.isEmpty()) {

46. QByteArray data;

47. if(frames.size() > 60) {

48. mutex.lock();

49. data = frames.last();

50. frames.clear();

51. mutex.unlock();

52. } else {

53. mutex.lock();

54. data = frames.first();

55. frames.removeFirst();

56. mutex.unlock();

57. }

58. QImage processedImage = detector.findBall(data);

59. this->image = processedImage;

60.

61. emit imageChanged();

62. }

63. }

64. }

65.

66. void VideoCamCapture::replyFinished(QNetworkReply \*reply)

67. {

68. if (reply->error() == QNetworkReply::NoError) {

69. QByteArray data = reply->readAll();

70. mutex.lock();

71. frames.append(data);

72. mutex.unlock();

73. }

74. }

# ПРИЛОЖЕНИЕ Г

(*обязательное*)

**Спецификация**

# ПРИЛОЖЕНИЕ Д

(*обязательное*)

**Перечень элементов**

# ПРИЛОЖЕНИЕ Е

(*обязательное*)

**Ведомость документов**