МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Консультант
_____ В.А. Руднев
«___» _____ 2023 г

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___» _____ 2023 г.

Разработка программного модуля для формирования и отправки управляющих команд на микроконтроллер ESP32

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–090301.2023.145 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____ Ю.Г. Плаксина
«___» _____2023 г.

Автор работы,
студент группы КЭ-405
_____М.А. Талыков
«___» _____2023 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___» _____2023 г.

Челябинск-2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой,
_____ Д.В. Топольский
«___» _____ 2023 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-405
Талыкову Максиму Андреевичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Разработка программного модуля для формирования и отправки управляющих команд на микроконтроллер ESP32» утверждена приказом по университету от 25.04.2023 г. №753-13/12

2. Срок сдачи студентом законченной работы: 01 июня 2023 г.

3. Исходные данные к работе:

- используемые языки: Java;
- платформа разработки Android Studio 2022.1.1;
- простой и интуитивно понятный интерфейс;
- версия Android не ниже 7.0.

4. Перечень подлежащих разработке вопросов:

1. Анализ предметной области;
2. Проектирование интерфейса программного модуля;
3. Разработка алгоритма для передачи управляющих команд;
4. Проектирование программного модуля;
5. Разработка программного модуля.

5. Дата выдачи задания: «____» декабря 2022 г.

Руководитель работы_____/ Ю.Г. Плаксина /

Студент _____/ М.А. Талыков /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Анализ предметной области	01.03.2023	
Проектирование интерфейса программного модуля	01.04.2023	
Разработка алгоритма для передачи управляющих команд	24.04.2023	
Проектирование программного модуля	10.05.2023	
Разработка программного модуля	15.05.2023	
Компоновка текста работы и сдача на нормоконтроль	22.05.2023	
Подготовка презентации и доклада	30.05.2023	

Руководитель работы_____/ Ю.Г. Плаксина /

Студент _____/ М.А. Талыков /

АННОТАЦИЯ

Талыков М.А. Разработка программного модуля для формирования и отправки управляющих команд на микроконтроллер ESP32 – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2023, 41 с., 5 ил., библиогр. список – 16 наим., 2 прил.

В рамках выпускной квалификационной работы разрабатывается программный модуль для формирования и отправки управляющих команд на микроконтроллер ESP32. В ходе работы был проведен аналитический обзор аналогов с выявлением наиболее технологичных решений, были выбраны необходимые для разработки средства и определены функциональные и нефункциональные требования. Опираясь на это, был разработан программный модуль для управления микроконтроллером ESP32 с устройства на операционной системе Android путем отправки управляющих команд через USB по RS-485 интерфейсу в формате JSON и получения ответов от устройства.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современных условиях производственная автоматизация имеет все большее значение, обеспечивая более эффективное и экономичное производство. Для реализации этой цели необходимо использовать высокотехнологичное оборудование, способное обеспечить надежную работу и возможность удаленного управления. Одним из наиболее перспективных решений в этой области является микроконтроллер ESP32, который представляет собой мощное устройство с богатым функционалом [1].

Актуальность данной темы обусловлена заданием заказчика, в котором необходимо разработать программный модуль для формирования и отправки управляющих команд с устройства Android на микроконтроллер ESP32. Данный проект позволит облегчить и ускорить процесс контроля и управления оборудованием, а также повысить точность и качество производства.

Данная операционная система (ОС) была выбрана, потому что она является самой популярной мобильной операционной системой в мире и это открытая операционная система, что означает большую гибкость и свободу для разработчиков [2].

ESP32 является устройством нового поколения и превосходит своих предшественников, таких как STM32 и Arduino, по ряду параметров, например, по вычислительной мощности, по интегрированным сетевым интерфейсам, памяти и другим характеристикам, которые позволяют использовать его в различных областях, включая производственную автоматизацию.

Основным преимуществом микроконтроллеров серии ESP32 является их доступность. Они производятся в Китае, китайской компанией Espressif Systems [3]. Данные микроконтроллеры дешевле чем их аналоги на рынке.

Главной исследовательской задачей данной работы является проверка взаимодействия и стабильности работы программного модуля с микроконтроллером ESP32 для выполнения промышленных задач.

Для достижения поставленной цели, необходимо решить следующие задачи:

– провести анализ предметной области;
– спроектировать интерфейс программного модуля;
– разработать алгоритм для передачи управляющих команд;
– необходимо спроектировать программный модуль;
– разработать программный модуль.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Варианты подключения к микроконтроллеру

Существует несколько вариантов подключения устройства на базе ОС Android к микроконтроллеру ESP32. Вот некоторые из них:

– микроконтроллер можно настроить как точку доступа или подключить его к существующей сети Wi-Fi, а затем можно подключить устройство Android, которое будет использовать протокол TCP/IP для связи и отправлять команды или получать данные;

– ESP32 поддерживает Bluetooth (BR/EDR and Bluetooth Low Energy);

– можно подключить микроконтроллер к устройству через USB кабель, а затем использовать соответствующий протокол для связи между двумя устройствами.

Если же Wi-Fi и Bluetooth подключение устройств требует беспроводного интерфейса, то USB подключению в рамках производственной автоматизации следует подобрать физический интерфейс для соединения Android устройства с ESP32 [4]. Это требуется для того, чтобы пользователю не нужно было постоянно бегать от одного микроконтроллера к другому и переключать между ними провода. Одними из самых популярных интерфейсов являются Ethernet и RS-485 [5]. Рассмотрим их поподробнее.

Ethernet является широко используемым физический интерфейс, предназначенным для передачи данных в производственной автоматизации с большим количеством устройств [6]. Он обеспечивает высокую скорость передачи и обрабатывает большие объемы данных, поддерживает различные протоколы, такие как Modbus TCP, которые могут быть использованы для связи с различными устройствами.

Преимущества Ethernet:

– достигает скоростей передачи данных до 10 Гбит/с, что позволяет обрабатывать большие объемы данных в реальном времени.

– обладает высокой надежностью и стабильностью работы благодаря использованию современных технологий и алгоритмов контроля ошибок.

– может использоваться для подключения различных устройств и обеспечения их взаимодействия с системами управления.

Недостатки Ethernet:

– оборудование Ethernet может быть дорогостоящим, особенно для устройств с высокой скоростью передачи данных;

– имеет ограничения на длину кабеля и расстояние между устройствами, что может ограничивать его использование в крупных производственных объектах;

– требует использования качественной кабельной инфраструктуры, такой как кабель категории 5 или выше.

RS-485 – это физический интерфейс, который широко используется в производственной автоматизации с большим количеством устройств (до 255 в одной сети) и обеспечивает передачу данных на большие расстояния [7].

Преимущества RS-485:

– может передавать данные на расстояние до 1200 метров без потерь качества сигнала, это позволяет использовать RS-485 в крупных производственных объектах и на больших территориях;

– оборудование RS-485 может быть гораздо более доступным, чем Ethernet-оборудование, особенно для устройств с низкой скоростью передачи данных;

– имеет простую архитектуру и простой протокол передачи данных, что облегчает установку и настройку оборудования;

– поддерживает мультиплексирование, что позволяет использовать один кабель для подключения нескольких устройств.

Недостатки RS-485:

– обладает низкой скоростью передачи данных по сравнению с Ethernet, из-за чего могут возникнуть проблему у устройств, которые требуют высокой скорости передачи данных.

– не поддерживает множество протоколов, что может быть проблемой для подключения различных устройств и систем управления.

1.2 Разбор аналогов

На данный момент в открытом доступе есть множество подобных проектов, предназначенных для передачи команд с устройства Android на различные микроконтроллеры через Wi-Fi, Bluetooth и USB. В качестве примеров рассмотрим программные модули для управления микроконтроллером при помощи прямого подключения через OTG кабель, так как данный способ обеспечивает наилучшее соединение между устройствами. Одними из самых популярных в магазине приложений для устройств на ОС Android Google Play являются: Serial USB Terminal, USB Serial Console, USB Serial Terminal Pro, Remote XY. Рассмотрим, что они из себя представляют.

Serial USB Terminal – это бесплатный программный модуль для устройств Android, который поддерживает соединение с микроконтроллерами через интерфейс USB [8]. Он позволяет отправлять и принимать данные, а также управлять настройками порта.

USB Serial Console – это еще один бесплатный программный модуль для Android, который позволяет отправлять и принимать данные через интерфейс USB [9]. Он имеет возможность настройки скорости передачи данных, битности и контроля четности.

USB Serial Terminal Pro – платный программный модуль, который имеет более расширенный функционал, чем Serial USB Terminal [10]. Он также позволяет отправлять и принимать данные через интерфейс USB и позволяет управлять настройками порта, но в отличии от своей бесплатной версии имеет возможность отправки нескольких команд одновременно.

Remote XY – это приложение для смартфонов и планшетов, предназначенное для управления и мониторинга микроконтроллеров и других электронных устройств через беспроводное или проводное соединение [11]. Оно предоставляет различные инструменты для программирования микроконтроллеров, включая библиотеки и примеры кода для популярных платформ, таких как ESP32, Arduino

и других. Также данное приложение позволяет создавать настраиваемые панели управления с кнопками, переключателями, ползунками и другими элементами управления.

Общие плюсы данных приложений:

– эти приложения имеют простой и интуитивно понятный интерфейс, что позволяет использовать их даже новичкам;

– данные проекты предоставляют возможность мониторинга и отладки данных, передаваемых через последовательный порт, у них есть возможность отслеживать входящие и исходящие данные, а также анализировать их формат и структуру, что особенно полезно при разработке и тестировании электронных устройств и программного обеспечения;

– некоторые из этих приложений позволяют пользователям создавать и запускать скрипты для автоматизации операций с последовательным портом, что может значительно упростить задачи, связанные с тестированием, настройкой и мониторингом устройств.

1.3 Основные технологические решения

После проведенного аналитического обзора следует выбрать наилучшие способы для реализации данного проекта и выявить функциональные и нефункциональные требования.

1.3.1 Выбор средств разработки

Использование USB подключения для управления с устройства Android предоставляет ряд преимуществ для производственной автоматизации [12]. Во-первых, оно обеспечивает высокую скорость передачи данных, что является критически важным для быстрой отправки управляющих команд между устройствами. Во-вторых, это позволяет использовать стандартные кабели и разъемы, что упрощает процесс подключения и облегчает замену поврежденных кабелей. В-третьих, USB подключение обеспечивает надежность и защищенность передачи данных, так как оно осуществляется точно и без шумов, что уменьшает количество ошибок и повышает качество передачи данных.

Для реализации проекта был выбран физический интерфейс RS-485, так как он имеет ряд преимуществ перед другими протоколами для производственной автоматизации. Выбранный вариант обладает высокой скоростью передачи данных и широким диапазоном рабочих температур, что является важным фактором в условиях производства. Также он может передавать данные на большие расстояния и поддерживает подключение до 255 устройств к одной линии. Для связи устройства Android и микроконтроллера с помощью RS-485 потребуется преобразователь интерфейсов USB в RS-485. Еще одним плюсом является высокая устойчивость к помехам и надежность передачи данных, что критически важно для производственной автоматизации. Все это делает RS-485 наиболее подходящим для использования в системах управления производственным оборудованием с помощью микроконтроллера и устройства Android. Действительно, можно использовать подключение через Wi-Fi, однако при большом количестве беспроводных сетей будут возникать помехи и устройства будут работать не так эффективно, что в условиях промышленных процессов неприемлемо. При выборе сетевого интерфейса не стоит забывать о том, что проводная сеть заведомо надежнее и будет более приемлемой в производственной автоматике.

Рассмотрев аналогичные программные модули, можно сделать вывод, что интуитивно понятный интерфейс, который позволяет пользователям легко управлять микроконтроллерами со своих устройствах на ОС Android, а также простой доступ к командам и данным, передаваемым через последовательный порт, делают взаимодействие с устройствами, подключенными посредством USB, более удобным даже для новичков.

В качестве языка программирования для разработки выбран Java [13]. Он является старым и хорошо установленным языком программирования, который используется на протяжении длительного времени для разработки Android приложений. Большое количество существующего кода и библиотек написаны на Java, что делает его более доступным и интегрируемым в существующие проекты.

Исходя из выбранных способов реализации проекта следует рассмотреть библиотеки для работы с портом USB. Данные будут передаваться в формате JSON.

Рассмотрим самые популярные библиотеки для работы с последовательным портом устройств Android jSerialComm, RxTx и mik3y/usb-serial-for-android.

jSerialComm предлагает простой API для чтения и записи данных через последовательные порты [14]. Она может работать на различных платформах и обеспечивает базовый функционал для обмена данными. Однако данная библиотека не предоставляет специфической поддержки для работы с форматом JSON.

RxTx предлагает расширенный функционал для работы с последовательными портами и может использоваться для передачи данных в формате JSON [15]. Она имеет хорошую поддержку сообщества и способна управлять потоком данных. Может быть сложной в использовании, особенно для новичков в работе с последовательными портами. Не обновляется и не поддерживается официальным разработчиком с 2012 года.

Библиотека mik3y/usb-serial-for-android является специализированной для работы с последовательными портами через USB на Android устройствах [16]. Она предоставляет оптимизированный API для работы и может легко использоваться для передачи данных в формате JSON. Библиотека предназначена для использования на Android устройствах и может не поддерживать другие платформы.

Рассмотрев данные варианты, можно сделать вывод, что лучше реализовать проект с помощью mik3y/usb-serial-for-android, так как она специально разработана для работы с подключаемыми устройствами через USB на платформе Android, может использоваться для передачи данных в формате JSON и поддерживается разработчиком.

1.3.2 Определение требований

Функциональные требования:

– пользователь должен иметь возможность ввести команду и отправить ее на устройство через последовательный порт;

– после отправки команды, устройство Android должно получать ответ от микроконтроллера и отображать его пользователю;

– пользователь должен иметь возможность настраивать параметры связи, такие как скорость передачи, количество битов данных, стоп-биты и проверку четности;

– программный модуль должен автоматически обнаруживать доступные последовательные порты и предоставлять список для выбора;

– каждая отправленная команда и полученный ответ должны сопровождаться временной меткой для легкого отслеживания и контроля.

Нефункциональные требования:

– вид интерфейса должен быть простым, интуитивно понятным и легким в использовании для пользователей различного уровня опыта;

– программный модуль должен быть стабильным и надежным, обеспечивая корректное соединение и передачу данных между устройствами;

– должна быть обеспечена эффективная работа с минимальной задержкой и быстрой отзывчивостью пользовательского интерфейса.

2 ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПРОГРАММНОГО МОДУЛЯ

Программный модуль должен иметь простой и интуитивно понятный интерфейс для пользователя, поэтому было принято решение разделить его на два основных окна.

Первое содержит основные элементы для общения с микроконтроллером, такие как:

– строка ввода команды: пользователь может ввести управляющую команду в эту строку;

– кнопка отправки команды: пользователь может нажать на эту кнопку, чтобы отправить команду на микроконтроллер;

– область вывода отправленных команд, где будет отображаться список отправленных команд с указанием времени отправки;

– кнопка для перехода во второе окно.

Пример первого окна приложения представлен на рисунках 2.1 и 2.2.

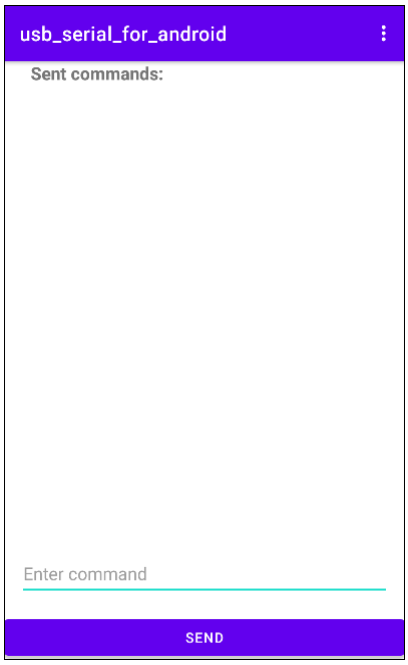

Рисунок 2.1 – Главное окно интерфейса.

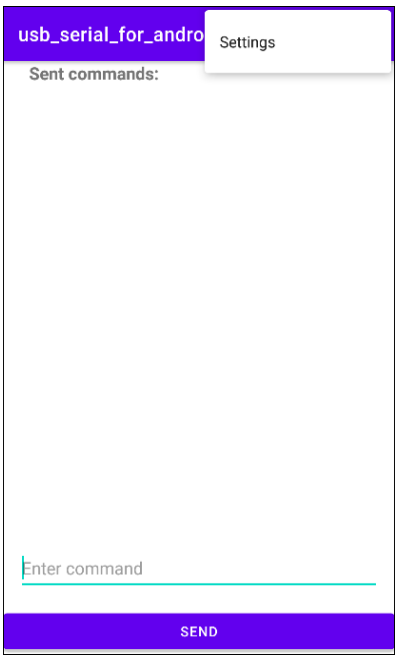

Рисунок 2.2 – Главное окно интерфейса с отображением кнопки для перехода к настройкам.

Второе окно представляет собой настройку последовательного порта и выбор микроконтроллера для последующего общения с ним и содержит следующие элементы:

– выбор порта устройства из списка;

– строка для установки скорости передачи данных (BaudRate). Это значение определяет с какой скоростью устройство будет отправлять данные;

– строка выбора количества битов данных (DataBits), используемых для передачи каждого символа;

– строка для установки стоп-битов (StopBits), которые указывают на конец каждого переданного символа;

– строка выбора режима проверки четности (Parity) для обеспечения целостности передаваемых данных;

– кнопка для подключения к выбранному порту с установленными параметрами.

Пример окна настроек представлен на рисунке 2.3.

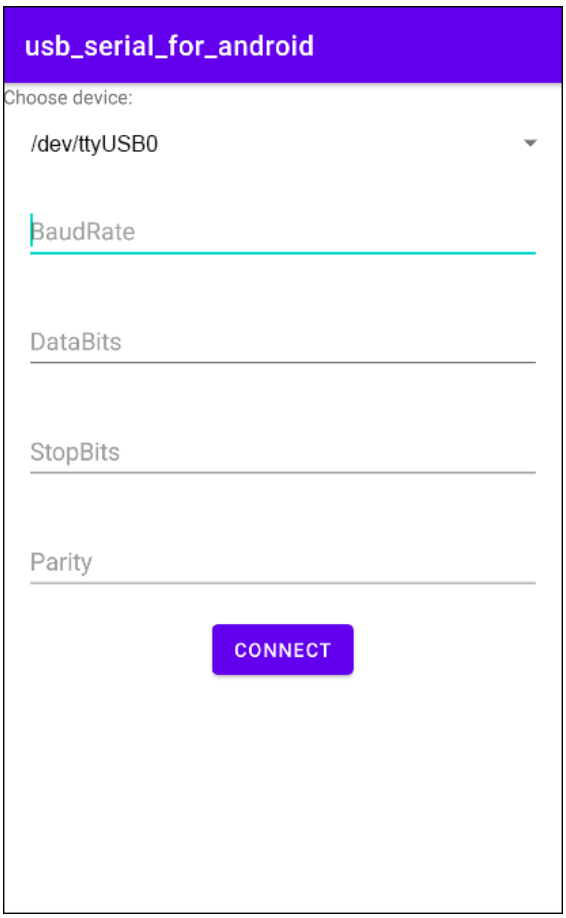

Рисунок 2.3 – Окно настроек с кнопкой подключения к устройству по заданным параметрам.

Плюсы такого интерфейса для пользователя:

– пользователь может легко вводить команды и видеть уже отправленные, что обеспечивает прозрачное взаимодействие с устройством;

– простой и понятный интерфейс, что облегчает использование программного модуля даже для пользователей без опыта;

– возможность установки настроек параметров связи в соответствии с требованиями устройства, что позволяет достичь оптимальной производительности связи;

– автоматическая проверка доступных портов и предоставление их списка для выбора, что упрощает процесс подключения к правильному порту;

– каждый отправленный запрос и ответ сопровождаются временной меткой, что позволяет легко отслеживать и контролировать процессы обмена данными.

3 РАЗРАБОТКА АЛГОРИТМА ДЛЯ ПЕРЕДАЧИ УПРАВЛЯЮЩИХ КОМАНД

Данный алгоритм должен включать в себя такие возможности:

– при передаче управляющих команд важно идентифицировать целевое устройство, с которым будет установлено соединение;

– необходимо установить соединение с микроконтроллером по выбранному порту;

– формирование управляющей команды, которую необходимо отправить устройству. Это может включать создание объекта или структуры данных, которые содержат информацию о команде, ее параметрах и других необходимых деталях;

– преобразование команды в последовательность битов для передачи;

– команда должна быть отправлена устройству по установленному соединению.

Исходя из заявленных возможностей алгоритма для передачи управляющих команд можно сделать вывод, что его конечный вид выглядит таким образом:

1. Пользователь взаимодействует с интерфейсом приложения, устанавливает соединение с выбранным устройством, вводит управляющую команду и нажимает кнопку отправки.

2. Введенная команда передается модулю, ответственному за отправку команд.

3. Модуль преобразует команду в удобный формат для передачи, обеспечивая правильную структуру данных.

4. Сформированная команда преобразуется в последовательность битов или байтов, готовых для передачи через USB.

Схема для данного алгоритма представлена на рисунке 3.4.

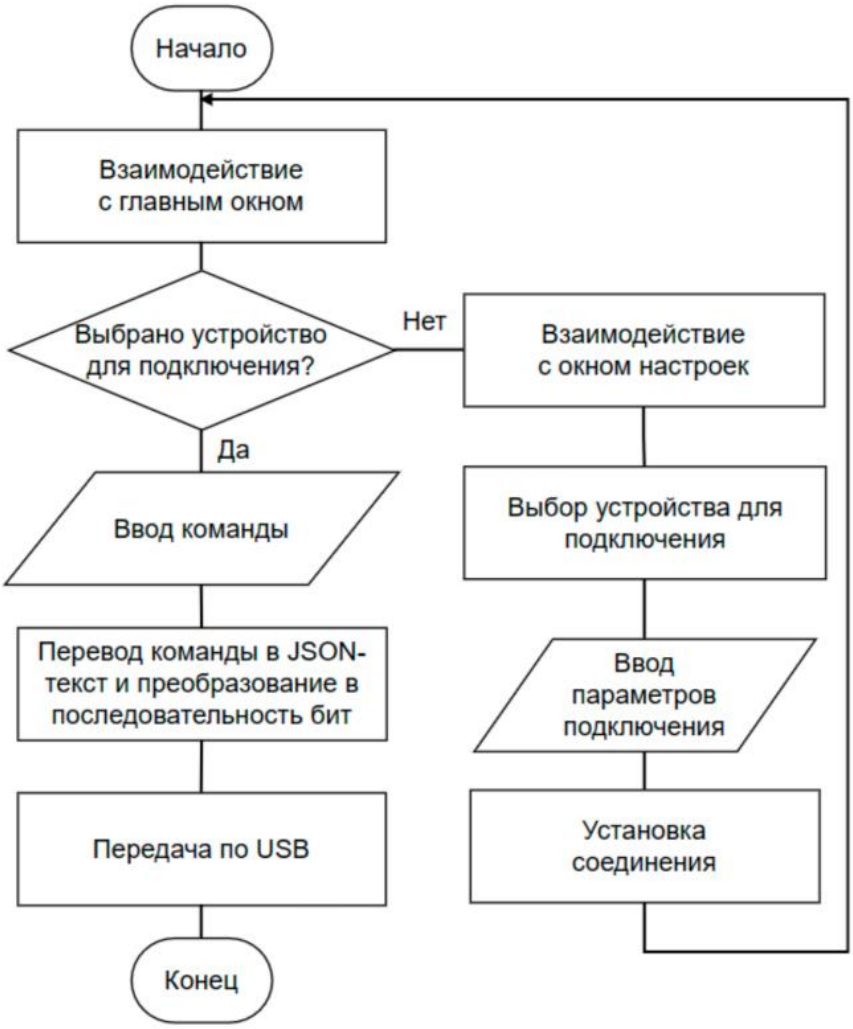

Рисунок 3.4 – Схема алгоритма для передачи управляющих команд

4 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ

В ходе проектирования была реализована UML диаграмма классов программного модуля. Она представлена на рисунке 4.5.

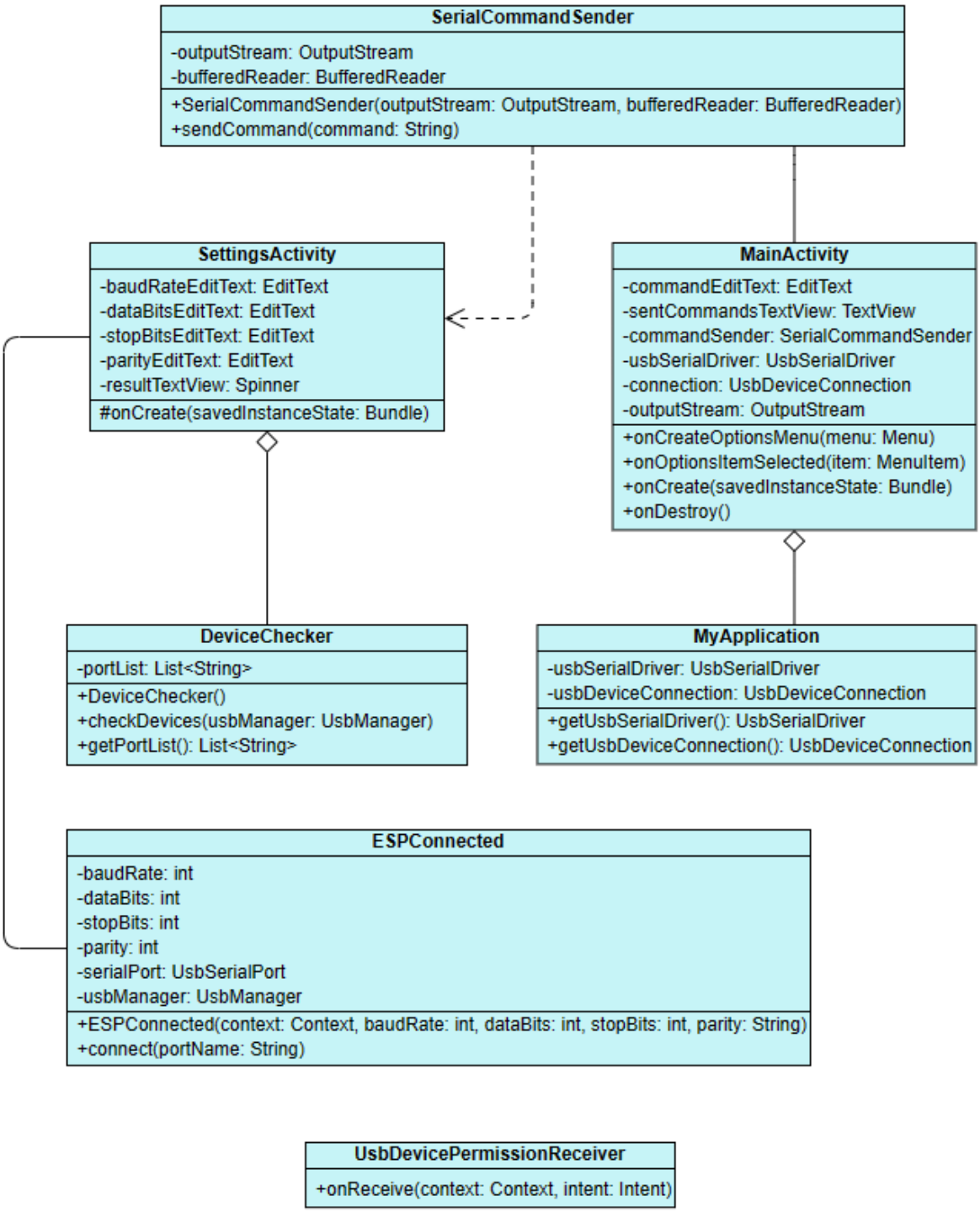

Рисунок 4.5 – UML диаграмма классов

Для реализации проекта потребуются такие классы: DeviceChecker, ESPConnected, SerialCommandSender, SettingsActivity, MyApplication и

MainActivity, UsbDevicePermissionReceiver. Все они нужны для управления устройством, соединенным с микроконтроллером ESP32 через USB по RS-485 интерфейсу. Программный модуль работает на устройствах Android не ниже версии 7.0 и позволяет формировать и отправлять управляющие команды на микроконтроллер.

Каждый из классов проекта выполняет свои определенные задачи и функции. Рассмотрим для чего они нужны:

DeviceChecker – отвечает за проверку подключенных устройств. Он использует UsbManager для получения списка подключенных устройств и проверяет их идентификаторы, чтобы определить, являются ли они устройствами в интерфейсе RS-485. Добавляет найденные порты устройств в список для дальнейшего взаимодействия с ними.

ESPConnected – отвечает за установку соединения с ESP32 и отправку команд. Он получает информацию о порте, к которому нужно подключиться, и использует UsbSerialProber для поиска драйверов и портов. Затем он открывает порт и устанавливает параметры связи, такие как скорость передачи данных, количество бит данных, количество стоп-бит и четность.

SerialCommandSender – этот класс используется для отправки команд с помощью объекта OutputStream и получения ответов с помощью BufferedReader. Он преобразует команду в формат JSON, отправляет ее через OutputStream и ждет ответа от устройства через BufferedReader.

SettingsActivity – активность, которая отображает пользовательский интерфейс для настройки параметров соединения и выбора порта подключения. В этой активности пользователь может ввести значения для скорости передачи данных, количества бит данных, количества стоп-бит и четности. При нажатии на кнопку "Connect" происходит создание экземпляра ESPConnected и установка соединения с выбранным портом ESP32.

MyApplication – класс, расширяющий Application, и предназначен для хранения и передачи состояния приложения между активностями.

MainActivity – главная активность приложения, которая отображает пользовательский интерфейс для отправки команд, отображения их и полученных ответов. В этом модуле пользователь может ввести команду в соответствующую строку и отправить ее с помощью кнопки "Send". Когда команда отправляется, она передается в SerialCommandSender для отправки, а результат отображается в поле отправленных команд.

UsbDevicePermissionReceiver – приемник широковещательных сообщений, который отслеживает подключение по USB и запрашивает разрешение на его использование. Когда устройство подключается, происходит запрос разрешения через UsbManager. При получении разрешения пользователь сможет управлять микроконтроллером с помощью своего устройства.

Общая логика работы проекта сводится к следующим шагам:

1. При запуске приложения DeviceChecker проверяет устройства подключенные к RS-485 интерфейсу и обновляет список доступных портов.
2. В SettingsActivity пользователь может выбрать параметры соединения, и при нажатии на кнопку "Connect" происходит установка соединения с выбранным портом микроконтроллера с помощью ESPConnected.
3. В MainActivity пользователь вводит команду в соответствующую строку и отправляет ее на ESP32 с помощью SerialCommandSender. Ответ отображается в поле для вывода отправленных команд.

Все классы взаимодействуют друг с другом, чтобы обеспечить управление устройством по RS-485 интерфейсу через USB на микроконтроллере ESP32.

5 РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ

На основании построенной UML диаграммы классов были реализованы классы программного модуля для формирования и отправки команд на микроконтроллер ESP32. Рассмотрим, какие методы они содержат.

5.1 DeviceChecker

checkDevices(UsbManager usbManager) предназначен для проверки подключенных устройств к RS-485 физическому интерфейсу. Он принимает объект UsbManager, который используется для получения списка подключенных USB устройств. Внутри метода происходит следующее:

– получается список всех подключенных устройств с помощью вызова usbManager.getDeviceList(), и результат сохраняется в HashMap<String, UsbDevice> deviceList;

– затем выполняется итерация по всем устройствам в deviceList;

– для каждого устройства получается его имя с помощью device.getDeviceName() и сохраняется в String deviceName;

– затем происходит проверка, является ли устройство RS-485, сравнивая его идентификатор производителя и идентификатор продукта со значениями 6790 и 29987 соответственно;

– если устройство является RS-485, то его порт добавляется в список portList, если его имя deviceName еще не содержится в списке.

Метод getPortList() возвращает список портов устройств, полученный после выполнения проверки в методе checkDevices().

5.2 ESPConnected

ESPConnected(Context context, int baudRate, int dataBits, int stopBits, String parity) – конструктор класса, который инициализирует параметры соединения и получает объект UsbManager через контекст Context.

Метод connect(String portName) устанавливает соединение с микроконтроллером по указанному порту. Он принимает имя порта portName в качестве параметра. Внутри происходит следующее:

– получается список доступных драйверов последовательных портов с помощью вызова UsbSerialProber.getDefaultProber().findAllDrivers(usbManager);

– затем выполняется двойная итерация: по каждому драйверу и по каждому последовательному порту, полученному из драйвера;

– если серийный номер устройства, связанного с текущим портом, совпадает с заданным portName, то этот порт сохраняется в переменную port и циклы прерываются;

– если порт не найден, выбрасывается исключение IOException с сообщением об ошибке.

После успешного поиска и выбора порта в методе connect:

– сохраняется выбранный порт в переменную serialPort;

– вызывается метод open(null) для открытия порта;

– вызов метода setParameters(baudRate, dataBits, stopBits, parity) для установки параметров соединения на порту.

Метод disconnect() закрывает соединение с портом. Если serialPort не равен null, то вызывается метод close.

5.3 SettingsActivity

Метод onCreate(Bundle savedInstanceState). Вызывается при создании активности. Действия внутри метода происходят следующим образом:

– вызывается setContentView(R.layout.settings_activity) для установки пользовательского интерфейса из файла layout settings_activity.xml;

– получаются ссылки на все визуальные элементы с помощью findViewById;

– создается объект UsbManager для работы с USB-устройствами;

– формируется экземпляр класса DeviceChecker и вызывается метод checkDevices(usbManager) для обновления списка доступных портов;

– создается адаптер ArrayAdapter и устанавливается для resultTextView для отображения списка доступных портов;

– устанавливается обработчик нажатия на кнопку connectButton;

– получаются значения, введенные пользователем для параметров соединения (baudRate, dataBits, stopBits, parity);

– получается выбранный порт из resultTextView;

– создается экземпляр класса ESPConnected с передачей параметров соединения и выбранного порта;

– вызывается метод connect(selectedPort) для установки соединения с ESP32;

– создается Intent для перехода к MainActivity;

– вызывается startActivity(intent) для запуска MainActivity.

5.4 SerialCommandSender

SerialCommandSender(OutputStream outputStream, BufferedReader bufferedReader) – конструктор класса, который принимает OutputStream и BufferedReader в качестве параметров и инициализирует соответствующие переменные.

Метод sendCommand(String command) отправляет команду на устройство ESP32 и ожидает ответа. Он принимает строку command в качестве параметра и возвращает ответ в виде строки. Внутри метода происходит следующее:

– команда command преобразуется в объект JSONObject с помощью конструктора JSONObject(command);

– команда преобразуется обратно в строку с добавлением символа новой строки "\n" и сохраняется в переменную stringCommand;

– строка команды stringCommand преобразуется в байтовый массив и отправляется через outputStream.write(stringCommand.getBytes());

– outputStream.flush() вызывается для принудительного сброса данных из буфера вывода;

– происходит чтение строки ответа из bufferedReader.readLine() и сохранение в переменную response.

После чтения ответа выполняется проверка:

– если response равен null, выбрасывается исключение IOException с сообщением об ошибке "Received null response from ESP32";

– если response не равен null, он возвращается в качестве результата метода.

5.5 MyApplication

Приватные поля usbSerialDriver и usbDeviceConnection представляют объекты UsbSerialDriver и UsbDeviceConnection соответственно.

Публичные методы getUsbSerialDriver() и getUsbDeviceConnection() возвращают значения полей usbSerialDriver и usbDeviceConnection соответственно.

Методы setUsbSerialDriver(UsbSerialDriver driver) и setUsbDeviceConnection (UsbDeviceConnection connection) используются для установки значений полей usbSerialDriver и usbDeviceConnection соответственно.

Этот класс позволяет хранить экземпляры UsbSerialDriver и UsbDeviceConnection на уровне приложения и предоставляет методы для получения и установки этих значений.

5.6 MainActivity

Содержание класса:

Конструктор MainActivity() инициализирует commandSender и outputStream с нулевыми значениями.

Метод onCreateOptionsMenu(Menu menu), в котором создается меню для активности, используя XML-ресурс.

Метод onOptionsItemSelected(MenuItem item), в котором обрабатывается выбор пункта меню. Если выбран пункт "Settings", то создается и запускается активность SettingsActivity.

Метод onCreate(Bundle savedInstanceState). Вызывается при создании активности. В нем происходит инициализация элементов пользовательского

интерфейса, инициализация commandSender и outputStream, а также установка слушателя для кнопки отправки команд.

Метод onDestroy(), который вызывается при уничтожении активности. В нем закрывается outputStream.

5.7 UsbDevicePermissionReceiver

Данный класс содержит переопределенный метод onReceive(Context context, Intent intent), который вызывается при получении широковещательного сообщения. Внутри происходит следующее:

– получается действие из intent.getAction();

– если действие соответствует UsbManager.ACTION_USB_DEVICE_ATTACHED, то происходит обработка подключенного USB-устройства;

– получается объект UsbDevice из intent.getParcelableExtra (UsbManager.EXTRA_DEVICE);

– если usbDevice не равен null, то выполняется запрос разрешения на использование USB-устройства;

– получается объект UsbManager из context.getSystemService (Context.USB_SERVICE);

– создается PendingIntent для запроса разрешения с помощью PendingIntent.getBroadcast();

– вызывается usbManager.requestPermission(usbDevice, permissionIntent) для запроса разрешения на использование USB-устройства;

– если действие соответствует "com.android.example.USB_PERMISSION", то происходит обработка полученного разрешения на использование USB-устройства;

– получается объект UsbDevice из intent.getParcelableExtra (UsbManager.EXTRA_DEVICE);

– если разрешение было предоставлено (intent.getBooleanExtra (UsbManager.EXTRA_PERMISSION_GRANTED, false) равно true) и usbDevice не

равен null, то выполняются действия при получении разрешения на использование USB-устройства;

– в противном случае, выводится сообщение в журнал о том, что разрешение на использование USB-устройства было отклонено.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы были выполнены следующие задачи:

– был проведен анализ предметной области, в ходе которого были выбраны средства разработки и методы реализации проекта;

– спроектирован интерфейс программного модуля;

– разработан алгоритм для передачи управляющих команд;

– проведено проектирование программного модуля;

– разработан программный модуль.

В процессе анализа предметной области был произведен обзор аналогов, в ходе которого рассматривались различные программные модули для взаимодействия с микроконтроллерами. Все они имели достоинства и недостатки. Самым главным недостатком являлось то, что у многих из них нет возможности взаимодействовать с ESP32, а также не каждое приложение может подойти для использования в промышленных задачах. На фоне этого были выявлены основные решения по реализации проекта, такие как использование USB и RS-485, для установки надежного и бесперебойного соединения между устройством Android и микроконтроллером для передачи управляющих команд. За счет выбора данных интерфейсов при использовании разработанного программного модуля пользователь получает список устройств подключенных к RS-485, что позволяет выбирать микроконтроллер для последующего взаимодействия, не подходя к нему. Таким образом данный проект позволит облегчить и ускорить процесс контроля и управления оборудованием, а также повысить точность и качество производства.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Android. Программирование для профессионалов [Текст] : [16+] / Б. Филлипс, К. Стюарт, К. Марсикано ; [перевел с английского Е. Матвеев]. - 3-е изд. - Санкт-Петербург [и др.] : Питер, 2019. - 687 с.

2 Mobile app development [Электронный ресурс] // Wikipedia the free encyclopedia [сайт]. - URL: https://en.wikipedia.org/wiki/Mobile_app_development (дата обращения: 15.02.23)

3 Espressif Systems [Электронный ресурс] // Cyclowiki [сайт]. - URL: https://cyclowiki.org/wiki/Espressif_Systems (дата обращения: 16.02.23)

4 Беспроводные интерфейсы IrDA, Bluetooth, WiUSB, WiFi, WiMax. [Электронный ресурс] // IT-IATU [сайт]. - URL: https://it-iatu.ru/is/arhitektura-evm/besprovodnye_interfeysy_irda_bluetooth_wiusb_wifi_wimax (дата обращения: 16.02.23)

5 Популярные интерфейсы и протоколы, используемые в приборах и контроллерах [Электронный ресурс] // StudFiles Файловый архив студентов [сайт]. - URL: https://studfile.net/preview/4616102/page:13/ (дата обращения: 22.02.2023)

6 Ethernet — 40 лет: от наброска на салфетке до гигабитных линий связи [Электронный ресурс] // Habr [сайт]. - URL: https://habr.com/ru/articles/526670/ (дата обращения: 22.02.23)

7 RS485 — стандарт промышленных сетей. [Электронный ресурс] // Habr [сайт]. - URL: https://habr.com/ru/companies/milandr/articles/540084/ (дата обращения: 22.02.23)

8 Serial USB Terminal // Google Play 2022.2 [сайт]. - URL: https://play.google.com/store/apps/details?id=de.kai_morich.serial_usb_terminal&hl=fr (дата обращения: 23.02.23)

9 USB Serial Console // Google Play 2022.2 [сайт]. - URL: https://play.google.com/store/apps/details?id=jp.sugnakys.usbserialconsole (дата обращения: 23.02.23)

10 USB Serial Terminal Pro // Google Play 2022.2 [сайт]. - URL: https://play.google.com/store/apps/details?id=com.oneman.tools (дата обращения: 23.02.23)

11 RemoteXY: управление Arduino // Google Play 2022.2 [сайт]. - URL: https://play.google.com/store/apps/details?id=com.shevauto.remotexy.free&hl=ru&gl=US (дата обращения: 23.02.23)

12 Работа с устройствами USB в Android [Электронный ресурс] // Habr [сайт]. - URL: https://habr.com/ru/articles/277093/ (дата обращения: 26.02.23)

13 Comparison of Java and Android API [Электронный ресурс] // Unity Wikipedia the free encyclopedia [сайт]. - URL: https://en.wikipedia.org/wiki/Comparison_of_Java_and_Android_API (дата обращения: 26.02.23)

14 What is jSerialComm? [Электронный ресурс] // jSerialComm [сайт]. - URL: https://fazecast.github.io/jSerialComm/ (дата обращения: 26.02.23).

15 rxtx/rxtx [Электронный ресурс] // GitHub [сайт]. - URL: https://github.com/rxtx/rxtx (дата обращения: 26.02.23)

16 mik3y/usb-serial-for-android [Электронный ресурс] // GitHub [сайт]. - URL: https://github.com/mik3y/usb-serial-for-android (дата обращения: 26.02.23)

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Классы программного модуля

А.1 Фрагмент класса «MainActivity.java»

```
public class MainActivity extends AppCompatActivity {

    private EditText commandEditText;
    private TextView sentCommandsTextView;
    private final SerialCommandSender commandSender;
    private final OutputStream outputStream;
    private SimpleDateFormat dateFormat;

    public MainActivity() {
        this.commandSender = null;
        this.outputStream = null;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_settings) {
            Intent intent = new Intent(this, SettingsActivity.class);
            startActivity(intent);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        commandEditText = findViewById(R.id.commandEditText);
        sentCommandsTextView = findViewById(R.id.sent_commands_text_view);
        dateFormat = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());

        UsbSerialDriver usbSerialDriver = ((MyApplication)
getApplication()).getUsbSerialDriver();
        if (usbSerialDriver != null) {
            UsbDeviceConnection connection = ((MyApplication)
getApplication()).getUsbDeviceConnection();
            if (connection != null) {
                UsbSerialPort serialPort = usbSerialDriver.getPorts().get(0);
                try {
                    serialPort.open(connection);
                    serialPort.setParameters(115200, 8, UsbSerialPort.STOPBITS_1,
UsbSerialPort.PARITY_NONE);
                    byte[] dataToSend = "Hello, World!".getBytes();
                    serialPort.write(dataToSend, 1000); // Отправляем данные
                    byte[] dataReceived = new byte[4096];
```

```
                    int numBytesRead = serialPort.read(dataReceived, 1000);
                    String receivedString = new String(dataReceived, 0, numBytesRead);
                            Log.d(TAG, "Received data: " + receivedString);
                } catch (IOException e) {
                    Toast.makeText(MainActivity.this, e.getMessage(),
Toast.LENGTH_SHORT).show();
                }
            }
        }
        Button sendCommandButton = findViewById(R.id.sendCommandButton);
        sendCommandButton.setOnClickListener(v -> {
            String command = commandEditText.getText().toString();
            try {
                String response = commandSender.sendCommand(command);
                sentCommandsTextView.append(dateFormat.format(new Date()) + " SENT: " +
command + "\nRESPONSE: " + response + "\n\n");
                commandEditText.getText().clear();
            } catch (IOException | JSONException e) {
                Toast.makeText(MainActivity.this, e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException e) {
                Toast.makeText(MainActivity.this, e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

А.2 Фрагмент класса «SettingsActivity.java»

```
public class SettingsActivity extends AppCompatActivity {
    private EditText baudRateEditText;
    private EditText dataBitsEditText;
    private EditText stopBitsEditText;
    private EditText parityEditText;
    private Spinner resultTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings_activity);
        // Get views
        baudRateEditText = findViewById(R.id.boud_rate);
        dataBitsEditText = findViewById(R.id.data_bits);
        stopBitsEditText = findViewById(R.id.stop_bits);
        parityEditText = findViewById(R.id.parity);
        resultTextView = findViewById(R.id.resultTextView);
```

```
        Button connectButton = findViewById(R.id.connect_button);
        UsbManager usbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

        DeviceChecker deviceChecker = new DeviceChecker();
        deviceChecker.checkDevices(usbManager);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, deviceChecker.getPortList());
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        resultTextView.setAdapter(adapter);

        connectButton.setOnClickListener(v -> {
            int baudRate = Integer.parseInt(baudRateEditText.getText().toString());
            int dataBits = Integer.parseInt(dataBitsEditText.getText().toString());
            int stopBits = Integer.parseInt(stopBitsEditText.getText().toString());
            String parity = parityEditText.getText().toString();
            String selectedPort = (String) resultTextView.getSelectedItem();
            ESPConnected espConnected = new ESPConnected(SettingsActivity.this, baudRate,
dataBits, stopBits, parity);
            try {
                espConnected.connect(selectedPort);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            Intent intent = new Intent(SettingsActivity.this, MainActivity.class);
            startActivity(intent);
        });
    }
}
```

А.3 Фрагмент класса «ESPConnected.java»

```
public class ESPConnected {
    private final int baudRate;
    private final int dataBits;
    private final int stopBits;
    private final int parity;
    private UsbSerialPort serialPort;
    private final UsbManager usbManager;
    public ESPConnected(Context context, int baudRate, int dataBits, int stopBits, String
parity) {
        this.baudRate = baudRate;
        this.dataBits = dataBits;
        this.stopBits = stopBits;
        this.parity = Integer.parseInt(String.valueOf(parity));
        this.usbManager = (UsbManager) context.getSystemService(Context.USB_SERVICE);
    }
    public void connect(String portName) throws IOException, NumberFormatException {
        UsbSerialPort port = null;
        List<UsbSerialDriver> availableDrivers =
UsbSerialProber.getDefaultProber().findAllDrivers(usbManager);
        for (UsbSerialDriver driver : availableDrivers) {
            for (UsbSerialPort usbSerialPort : driver.getPorts()) {
                if
(usbSerialPort.getDriver().getDevice().getSerialNumber().equals(portName)) {
                    port = usbSerialPort;
                    break;}
```

```
            }
            if (port != null) {
                break;
            }
        }

        if (port == null) {
            throw new IOException("Could not find port with name: " + portName);
        }

        serialPort = port;
        serialPort.open(null);
        serialPort.setParameters(baudRate, dataBits, stopBits, parity);
    }

    public void disconnect() throws IOException {
        if (serialPort != null) {
            serialPort.close();
        }
    }

    public void sendCommand(JSONObject command) throws IOException {
        if (serialPort != null) {
            serialPort.write(command.toString().getBytes(), 1000);
        }
    }
}
```

А.4 Фрагмент класса «SerialCommandSender.java»

```
public class SerialCommandSender {

    private OutputStream outputStream;
    private BufferedReader bufferedReader;
    public SerialCommandSender(OutputStream outputStream, BufferedReader bufferedReader){
        this.outputStream = outputStream;
        this.bufferedReader = bufferedReader;
    }

    public String sendCommand(String command) throws IOException, JSONException {
        JSONObject jsonCommand = new JSONObject(command);
        String stringCommand = jsonCommand.toString() + "\n";
        outputStream.write(stringCommand.getBytes());
        outputStream.flush();
        String response = bufferedReader.readLine();
        if (response == null) {
            throw new IOException("Received null response from ESP32");
        }
        return response;
    }
}
```

А.5 Фрагмент класса «DeviceChecker.java»

```
public class DeviceChecker {

    private final List<String> portList;
```

```
    public DeviceChecker() {
        this.portList = new ArrayList<>();
    }
    public void checkDevices(UsbManager usbManager) {
        HashMap<String, UsbDevice> deviceList = usbManager.getDeviceList();
        for (UsbDevice device : deviceList.values()) {
            String deviceName = device.getDeviceName();
            if (device.getVendorId() == 6790 && device.getProductId() == 29987) {
                if (!portList.contains(deviceName)) {
                    portList.add(deviceName);
                }
            }
        }
    }
    public List<String> getPortList() {
        return portList;
    }
}
```

А.6 Фрагмент класса «UsbDevicePermissionReceiver.java»

```
public class UsbDevicePermissionReceiver extends BroadcastReceiver {
    private static final String USB_PERMISSION_ACTION =
"com.android.example.USB_PERMISSION";
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action)) {
            UsbDevice usbDevice = intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
            if (usbDevice != null) {
                UsbManager usbManager = (UsbManager)
context.getSystemService(Context.USB_SERVICE);
                PendingIntent permissionIntent = PendingIntent.getBroadcast(context, 0,
new Intent(USB_PERMISSION_ACTION), 0);
                usbManager.requestPermission(usbDevice, permissionIntent);
            }
        } else if (USB_PERMISSION_ACTION.equals(action)) {
            synchronized (this) {
                UsbDevice usbDevice = intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
                if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                    if (usbDevice != null) {
                        UsbManager usbManager = (UsbManager)
context.getSystemService(Context.USB_SERVICE);
                        UsbDeviceConnection connection =
usbManager.openDevice(usbDevice);
                        if (connection != null) {
                            connection.close();
                        }
                    }
                } else {
                    Log.d("UsbDevicePermissionReceiver", "Permission denied for USB
device " + usbDevice);
                }
            }
        }
    }
}
```

А.7 Фрагмент класса «MyApplication.java»

```
public class MyApplication extends Application {

private UsbSerialDriver usbSerialDriver;
    private UsbDeviceConnection usbDeviceConnection;

    public UsbSerialDriver getUsbSerialDriver() {
        return usbSerialDriver;
    }

    public UsbDeviceConnection getUsbDeviceConnection() {
        return usbDeviceConnection;
    }

}
```

ПРИЛОЖЕНИЕ Б

Окна интерфейса программного модуля

Б.1 Фрагмент файла «activity_main.xml»

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <ScrollView
        android:layout_width="389dp"
        android:layout_height="508dp"
        android:layout_marginStart="13dp"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="13dp"
        android:layout_marginBottom="100dp"
        android:layout_weight="1"
        app:layout_constraintBottom_toTopOf="@+id/linearLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints">
        <TextView
            android:id="@+id/sent_commands_text_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:rotationX="-360"
            android:text="@string/sent_commands"
            android:textSize="18sp"
            android:textStyle="bold" />
    </ScrollView>
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent">
        <EditText
            android:id="@+id/commandEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:autofillHints=""
            android:hint="@string/enter_command"
            android:inputType="text"
            android:minHeight="48dp"
            tools:ignore="VisualLintTextFieldSize" />
        <Button
            android:id="@+id/sendCommandButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:layout_gravity="end"
            android:text="@string/send"
            tools:ignore="VisualLintButtonSize" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Б.2 Фрагмент файла «settings_activity.xml»

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="top"
        android:orientation="vertical">
        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/choose_device" />
        <Spinner
            android:id="@+id/resultTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:entries="@array/UsbDevice"
            android:minHeight="48dp" />
        <EditText
            android:id="@+id/boud_rate"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:hint="@string/boud_rate"
            android:inputType="text"
            android:minHeight="48dp"
            android:autofillHints=""
            tools:ignore="VisualLintTextFieldSize" />
        <EditText
            android:id="@+id/data_bits"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:hint="@string/data_bits"
            android:inputType="text"
            android:minHeight="48dp"
            android:autofillHints=""
            tools:ignore="VisualLintTextFieldSize" />
        <EditText
            android:id="@+id/stop_bits"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:hint="@string/stop_bits"
            android:inputType="text"
```

```
            android:minHeight="48dp"
            android:autofillHints=""
            tools:ignore="VisualLintTextFieldSize" />
        <EditText
            android:id="@+id/parity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:hint="@string/parity"
            android:inputType="text"
            android:minHeight="48dp"
            android:autofillHints=""
            tools:ignore="VisualLintTextFieldSize" />
        <Button
            android:id="@+id/connect_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/connect"
            android:layout_gravity="center"/>
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```