Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. каф. ЭВМ

\_\_\_\_\_\_\_\_\_\_ Б.В. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

ANDROID-ПРИЛОЖЕНИЕ ДЛЯ МОНИТОРИНГА И УПРАВЛЕНИЯ УСТРОЙСТВАМИ УМНОГО ДОМА

БГУИР ДП 1–40 02 01 01 xxx ПЗ

Студент Е.В. Максимчик

Руководитель Ю.А. Луцик

Консультанты:

от кафедры ЭВМ Ю.А. Луцик

по экономической части В.Г. Горовой

Нормоконтролер А.И Стракович

Рецензент

МИНСК 2025

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Благодаря развитию передовых технологий повседневная жизнь человека становится удобнее и эффективнее. Развитие систем «умный дом» позволил автоматизировать большинство бытовых процессов, обеспечивая комфорт для пользователей. Современные решения позволяют управлять освещением, климатом, системами безопасности и другими элементами, создавая интеллектуальную и адаптивную среду.

Крупные технологические компании сосредоточены на разных аспектах систем: одни разрабатывают инновационные решения в области умных замков и систем безопасности, другие занимаются созданием интеллектуальных термостатов и климат-контроля, а третьи предлагают комплексные решения для автоматизированного освещения.

Развитие универсальных стандартов связи значительно упростило взаимодействие устройств разных производителей, обеспечивая их совместимость и позволяя пользователям создавать гибкие, масштабируемые экосистемы для управления. В результате современные системы автоматизации становятся все более функциональными и доступными.

Несмотря на все преимущества, вопрос стоимости остается одним из ключевых факторов, ограничивающих массовое внедрение систем умный дом. Высокие цены на оборудование, необходимость приобретения дополнительных шлюзов и контроллеров, а также расходы на установку и настройку могут стать серьезным барьером для пользователей. Кроме того, некоторые устройства требуют регулярного обновления или подписки на облачные сервисы, что увеличивает общую стоимость использования внедряемой системы.

Однако, помимо готовых коммерческих решений, все большую популярность приобретают DIY-устройства (Do It Yourself – «сделай сам»), позволяющие пользователям самостоятельно создавать и настраивать системы умного дома под свои нужды. Такой подход не только снижает затраты, но и дает возможность полной кастомизации, позволяя интегрировать исполнительные устройства в единую интеллектуальную сеть.

Целью данного дипломного проекта является разработка Android-приложения, которое обеспечит мониторинг и управление устройствами умного дома, предоставляя пользователю интуитивно понятный интерфейс для эффективного контроля, настройки и автоматизации различных устройств.

Для достижения поставленной цели дипломного проекта важно разбить ее на конкретные задачи:

* прошивка и настройка шлюза умного дома;
* проектирование архитектуры системы «умный дом»;
* разработка Android-приложения для yправление системой «умный дом»;
* тестирование и отладка разработанного приложения.

# 1 ОБЗОР ЛИТЕРАТУРЫ

* 1. Обзор существующих аналогов
     1. **Система умный дом «Xiaomi»**
     2. **Система умный дом «Tuya»**
     3. **Система умный дом «Яндекс»**
     4. **Система умный дом «Aqara»**
  2. Беспроводные протоколы связи
     1. **Z-Wave**
     2. **Bluetooth Low Energy**
     3. **Zigbee**
  3. Протоколы обмена сообщений
     1. **HTTP REST API**
     2. **MQTT**
  4. Интегрированная среда разработки Android Studio
  5. Архитектура приложений
     1. **MVC**

* + 1. **MVVM**

1. СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После анализа предметной области разрабатываемого приложения были сформулированы основные требования, которые необходимо учитывать при его реализации. Поскольку приложение включает обширный функционал, включая работу с базой данных, обработку и отправку данных MQTT-брокеру, работа в локальной сети с использованием HTTP, возникает необходимость в проектировании структурной схемы программного средства.

Структура проекта состоит из следующих блоков:

* блок MQTT клиента;
* блок соединения с брокером;
* блок коммуникации MQTT;
* блок обработки и генерации сообщений;
* блок базы данных;
* блок моделей;
* блок бизнес-логики;
* блок моделей представления;
* блок представления.

Структура системы была организована таким образом, чтобы каждый блок выполнял строго определенную задачу и обеспечивал стабильную работу приложения в целом.

Далее будет рассмотрен принцип работы каждого из выделенных блоков, их основные функции, а также взаимодействие между ними. Взаимосвязь между основными блоками проекта отражена на структурной схеме ГУИР.400201.ххх С1.

* 1. Блок MQTT клиента

Блок MQTT клиента отвечает за первичную настройку соединения, обеспечивая корректное взаимодействие с брокером. В рамках этой настройки задаются основные параметры подключения, включая адрес брокера, номер порта, уникальный идентификатор клиента, а также параметры безопасности, такие как логин и пароль.

Дополнительно можно задать интервал Keep Alive, который определяет, как часто клиент будет отправлять сигналы активности брокеру, предотвращая разрыв соединения. Также можно настроить механизм Last Will and Testament (LWT) – заранее заданное сообщение, которое брокер опубликует в случае неожиданного отключения клиента. Это позволяет другим участникам системы получать актуальную информацию о его состоянии.

Кроме того, при настройке соединения можно указать, должна ли сессия быть очищаемой (Clean Session). Если параметр включен, при каждом новом подключении клиент начинает работу с нуля, без сохраненных подписок. Если отключен, брокер запоминает подписки и недоставленные сообщения, обеспечивая более надежную связь при временных разрывах соединения.

При первом подключении системы «умного дома» данные для подключения вводятся пользователем через блок представления. Вводимые параметры, такие как адрес брокера, логин, пароль и другие настройки, проверяются на корректность и сохраняются в базе данных только при успешном вводе и установлении соединения. Это обеспечивает их последующее использование и предотвращает сохранение некорректных данных. При следующем запуске приложения блок MQTT клиента загружает сохраненные параметры из базы данных и автоматически устанавливает соединение с брокером, исключая необходимость повторного ввода данных. Такой подход повышает удобство работы пользователя и снижает вероятность ошибок, связанных с ручной настройкой.

Благодаря этому механизму обеспечивается не только удобное управление подключением, но и стабильность работы системы, так как параметры сохраняются и могут быть использованы для восстановления соединения в случае сбоя.

* 1. Блок соединения с брокером MQTT

Блок соединения с MQTT-брокером отвечает за установление и поддержание стабильного соединения между клиентом и брокером. Его главная задача – инициировать подключение, следить за его состоянием и восстанавливать связь в случае разрыва.

При запуске системы блок загружает сохраненные параметры подключения, такие как адрес брокера, номер порта, логин, пароль, уникальный идентификатор клиента, а также дополнительные настройки, включая Keep Alive, Clean Session и Last Will and Testament (LWT).

Далее создается MQTT-клиент с заданными параметрами, после чего он отправляет запрос на подключение к брокеру, используя аутентификацию по логину и паролю. Если соединение успешно установлено, блок уведомляет другие компоненты системы, в частности блок коммуникации MQTT, который затем подписывается на нужные топики и начинает обмен сообщениями.

Для поддержания соединения используется механизм Keep Alive, который позволяет клиенту периодически отправлять сигналы активности, предотвращая разрыв связи. Если же соединение все же прерывается, блок оперативно обнаруживает это и инициирует процедуру переподключения. При повторных неудачах используется механизм увеличения задержки между попытками восстановления связи, чтобы снизить нагрузку на сеть и брокер.

При каждом успешном подключении блок соединения может сохранять актуальные параметры в базе данных, чтобы использовать их при следующих запусках системы. Это позволяет исключить необходимость повторного ввода данных вручную.

Таким образом, блок соединения с MQTT-брокером выполняет ключевую функцию в системе, обеспечивая надежное подключение и автоматическое восстановление связи при сбоях.

* 1. Блок коммуникации MQTT

Блок коммуникации MQTT играет ключевую роль в системе «умного дома», обеспечивая обмен сообщениями между приложением и удаленным брокером. Его основная задача – установление защищенного соединения, подписка на топики и публикация сообщений, что позволяет передавать данные о состоянии устройств и отправлять им управляющие команды.

После подключения к брокеру с использованием аутентификации блок подписывается на определенные топики, что дает возможность получать актуальную информацию о состоянии устройств. При поступлении сообщения от брокера блок передает его в блок обработки и генерации сообщений, который анализирует содержимое и определяет дальнейшие действия. Если данные содержат информацию о новом состоянии устройства, они передаются в пользовательский интерфейс для отображения актуальной информации.

Кроме приема сообщений, блок коммуникации выполняет отправку данных, обеспечивая управление устройствами. Когда пользователь изменяет параметры в приложении, например, включает освещение или задает температуру, блок обработки формирует соответствующую команду и передает ее в блок коммуникации, который публикует сообщение в нужный топик MQTT. Это позволяет системе обеспечивать двусторонний обмен данными, необходимый для стабильной работы умного дома.

Для поддержания соединения блок включает механизмы обработки ошибок и восстановления связи. В случае разрыва соединения он уведомляет блок соединения с брокером, который предпринимает попытки восстановления. Также он контролирует целостность передаваемых данных, предотвращая некорректные сообщения и дублирование команд.

* 1. Блок обработки и генерации собщений

Блок обработки и генерации сообщений играет самую важную роль в системе, обеспечивая корректное взаимодействие между приложением и устройствами умного дома через MQTT-протокол. Он выполняет обработку входящих сообщений от брокера, анализируя их содержимое, и формирует исходящие команды для управления устройствами. Основная цель этого блока — преобразование сырых данных в понятный формат, фильтрация лишней информации и передача обработанных данных в соответствующие компоненты системы.

При получении сообщения от MQTT-брокера блок обработки и генерации анализирует его, проверяет на наличие ошибок и определяет, к какому устройству оно относится. Если данные содержат информацию о состоянии устройства, например, текущую температуру, уровень освещенности или статус реле, они передаются в пользовательский интерфейс для отображения актуальной информации.

Кроме обработки входящих сообщений, этот блок также отвечает за генерацию исходящих данных. Когда пользователь в приложении изменяет параметры работы устройства, блок обработки получает соответствующий запрос, преобразует его в формат MQTT-сообщения и передает в блок коммуникации для отправки брокеру. Это позволяет осуществлять двусторонний обмен данными между клиентом и устройствами умного дома.

Дополнительно блок включает механизмы фильтрации и валидации данных, предотвращая отправку некорректных команд и обеспечивая корректную работу системы. В случае обнаружения ошибки он может отправлять уведомления в интерфейс пользователя или запускать механизмы повторной передачи сообщения. Все это делает блок обработки и генерации сообщений важным элементом системы, который обеспечивает стабильную и надежную работу обмена данными через MQTT.

* 1. Блок базы данных

Информация, получаемая из блока бизнес-логики после обработки сообщений от брокера, записывается в базу данных, где она сохраняется в специальных таблицах, предназначенных для хранения данных о системе и подключенных устройствах. В качестве хранилища используется SQLite с библиотекой Room.

Блок базы данных отвечает за сохранение данных о подключении к удаленному брокеру сообщений, включая адрес брокера, номер порта, логин, пароль, уникальный идентификатор клиента, а также дополнительные настройки, такие как Keep Alive, Clean Session и Last Will and Testament. Это позволяет сохранять параметры подключения и восстанавливать их при последующем запуске приложения.

В таблице «устройства» хранятся данные о подключенных устройствах: уникальный сетевой адрес Zigbee, последние полученные значения от MQTT-брокера, выбранное пользователем изображение устройства, каналы управления (например, для переключения реле или управления освещением), тип устройства и его дружественное имя.

* 1. Блок моделей

Блок моделей отвечает за структуру данных, используемых в приложении. Он содержит классы, описывающие подключенные устройства умного дома, а также параметры подключения к MQTT-брокеру. Модели обеспечивают единообразное представление данных, используемых в бизнес-логике, базе данных и пользовательском интерфейсе.

Данные об устройствах включают уникальный идентификатор, тип устройства, текущее состояние, доступные команды и другие параметры, необходимые для взаимодействия. Также в моделях хранятся сведения о подключении к MQTT-брокеру, такие как адрес сервера, логин, пароль и настройки соединения.

* 1. Блок бизнес-логики

Блок бизнес-логики отвечает за обработку данных, выполнение основной логики работы приложения и взаимодействие с другими модулями. Он обеспечивает управление сообщениями MQTT, взаимодействие с базой данных, обработку данных перед отправкой в представление и выполнение пользовательских сценариев умного дома.

* 1. Блок представлений

Модели представления являются промежуточным слоем между моделями данных и представлениями, выполняя преобразование данных в формат, удобный для отображения. Они обрабатывают информацию, специфичную для пользовательского интерфейса, тогда как общие вычисления и бизнес-логика выносятся в другой блок.

Основное назначение моделей представления – подготовка данных для удобного использования в представлении. Например, если в модели устройства умного дома хранятся только время его включения и выключения, то модель представления может вычислять общее время работы устройства и передавать уже готовое значение для отображения пользователю.

Модель представления связана как с моделью данных, так и с представлением, обеспечивая их взаимодействие и отделяя обработку данных от логики интерфейса.

* 1. Блок моделей представлений

В Jetpack Compose корневым представлением является композиционная функция (Composable), с которой начинается работа приложения. Это представление содержит ссылки на другие представления, формируя иерархическую структуру интерфейса. При этом всегда возможна навигация назад, а также циклические переходы, когда одно представление может ссылаться на себя.

В Jetpack Compose любое отображаемое на экране содержимое является представлением (Composable). Например, кнопка – это отдельное представление, описанное декларативным способом, а содержащий ее список также является представлением. Таким образом, одно представление может включать в себя множество других, а благодаря механизму параметров при инициализации обеспечивается гибкость и переиспользуемость компонентов.

Представления должны работать одинаково на всех устройствах Android и адаптироваться под разные размеры экранов. Для этого используется механизм адаптивного пользовательского интерфейса, обеспечивающий корректное отображение элементов на смартфонах, планшетах и других устройствах.

Представление взаимодействует с моделью представления (ViewModel), получая из нее данные и отправляя события на обработку. Однако допускается выполнение простых вычислений непосредственно в представлениях. В некоторых случаях, когда представление содержит только статические данные или не требует сложной логики, оно может работать напрямую с моделью данных без использования ViewModel. Это особенно актуально для простых компонентов, таких как кнопки или отдельные элементы списка.

Представления также могут взаимодействовать с Composition Local, получая данные из общего контекста, что упрощает управление состоянием и передачу зависимостей. Функциональность представлений в системе "умного дома":

* пользовательский опыт. Важно учитывать удобство использования интерфейса: логичную навигацию, интуитивные элементы управления и четкую обратную связь при взаимодействии с системой;
* визуализация данных. Jetpack Compose позволяет создавать динамические и адаптивные интерфейсы для отображения данных. Это могут быть графики, диаграммы, карточки или таблицы, отображающие текущее состояние системы "умного дома";
* элементы управления, такие как Button, Switch, Slider, TextField, позволяют пользователю управлять устройствами "умного дома". Использование состояний (State) и событий (Event) обеспечивает интерактивность и отзывчивость интерфейса;
* адаптивность и поддержка разных экранов Jetpack Compose позволяет легко адаптировать интерфейс под различные размеры экранов.

Таким образом, использование Jetpack Compose в разработке интерфейса приложения "умного дома" позволяет создавать гибкие, адаптивные и удобные представления, обеспечивая качественный пользовательский опыт.

1. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ
2. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ
3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
4. ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ
5. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И РЕАЛИЗАЦИИ НА РЫНКЕ ANDROID-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ И МОНИТОРИНГА УСТРОЙСТВАМИ УМНОГО ДОМА

* 1. Характеристика программного средства, разрабатываемого для реализации на рынке

Созданный дипломный проект представляет собой нативное приложение для операционной системы Android, которое позволяет пользователям управлять устройствами «умного дома» через протокол MQTT для DIY систем.

Целью разработки проекта является упрощение управления устройствами «умного дома» и обработки данных о их состоянии. Приложение предназначено для пользователей, позволяя им эффективно контролировать устройства, настраивать автоматизации и оперативно получать информацию об их работе.

Целевой аудиторией данного приложения являются пользователи систем «умного дома», которым необходим удобный инструмент для управления устройствами через протокол MQTT. Также потенциальными пользователями могут быть энтузиасты DIY-решений, использующие платформы, такие как Home Assistant, а также владельцы экосистем Aqara, Xiaomi, Tuya и Яндекс.

На момент разработки существует большое количество решений для управления устройствами «умного дома», однако большинство из них ориентированы на конкретные экосистемы, такие как Aqara, Xiaomi, Tuya и Яндекс. Существующие приложения часто имеют ограниченный функционал и не поддерживают интеграцию с DIY-системами.

Планируется распространение приложения через Google Play с возможностью монетизации, включая бесплатную и расширенную платную версии.

А

А

А

* 1. Расчет инвестиций в разработку программного средства
     1. **Расчет зарплат на основную заработную плату разработчиков**

Расчет затрат на основную заработную плату разработчиков производится исходя из количества людей, которые занимаются разработкой программного продукта, месячной зарплаты каждого участника процесса разработки и сложности выполняемой ими работы. Затраты на основную заработную плату рассчитаны по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.1) |

где n – количество исполнителей, занятых разработкой конкретного ПО;

Кпр – коэффициент, учитывающий процент премий;

Зч.i – часовая заработная плата i-го исполнителя, р.;

ti – трудоемкость работ, выполняемых i-м исполнителем, ч.

Разработкой всего приложения занимается инженер-программист, обязанности тестирования приложения лежат на инженере-тестировщике. Задачами инженера-программиста, который занимается являются создание модели данных, графического интерфейса, связи между моделью данных и графическим интерфейсом. Инженер-тестировщик занимается выявлением неработоспособных частей приложения, а также оценивает пользовательский опыт, получаемый от использования приложения.

Месячная заработная плата основана на медианных показателях для Junior инженера-программиста за 2024 год по Республике Беларусь, которая составляет примерно 807 долларов США в месяц, а для Junior инженера-тестировщика – 466 долларов США [зп]. По состоянию на 21 февраля 2025 года, 1 доллар США по курсу Национального Банка Республики Беларусь составляет 3,2239 белорусских рубля.

В перерасчете на белорусские рубли месячные оклады для инженера-программиста и инженера-тестировщика составляют 2601,68 и 1502,33 белорусских рублей соответственно.

Часовой оклад исполнителей высчитывается путем деления месячного оклада на количество рабочих часов в месяце, то есть 160 часов.

За количество рабочих часов в месяце для инженера-программиста и инженера-тестировщика принято соответственно 196 и 32 часа. Коэффициент премии приравнивается к единице, так как она входит сумму заработной платы. Затраты на основную заработную плату приведены в таблице:

Таблица 7.1 – Затраты на основную заработную плату сотрудников

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р | Часовой оклад, р | Трудоемкость работ, ч | Итого, р |
| Инженер-программист | 2601,68 | 16,26 | 196 | 3186,96 |
| Инженер-тестировщик | 1502,33 | 9,39 | 32 | 300,46 |
| Итого | | | | 3487,42 |
| Премия и стимулирующие зарплаты (0%) | | | | 0 |
| Общие затраты на основную заработную плату разработчиков | | | | 3487,42 |

* + 1. **Расчет затрат на дополнительную заработную плату разработчиков**

Расчет дополнительных выплат, предусмотренных законодательством о труде, осуществлялся по формуле 7.2:

|  |  |  |
| --- | --- | --- |
|  |  | (7.2) |

где Зо – затраты на основную заработную плату, р.;

– норматив дополнительной заработной платы.

Норматив дополнительной заработной платы был принят равным 10%.

Размер дополнительной заработной составил:

* + 1. **Расчет отчислений на социальные нужды**

Расчет производился в соответствии с действующими законодательными актами по формуле 7.3:

|  |  |  |
| --- | --- | --- |
|  |  | (7.3) |

где Нсоц – норматив отчислений от фонда оплаты труда.

Норматив отчислений от фонда оплаты труда дополнительной заработной платы был принят равным 35%.

Размер дополнительной заработной платы составил:

* + 1. **Расчет прочих расходов**

Прочие расходов были рассчитаны по формуле 7.4:

|  |  |  |
| --- | --- | --- |
|  |  | (7.4) |

где Нпр – норматив прочих расходов.

Норматив прочих расходов был принят равным 30%.

Размер прочих расходов составил:

* + 1. **Расчет расходов на реализацию**

Расходы на реализацию продукта рассчитаны по формуле 7.5:

|  |  |  |
| --- | --- | --- |
|  |  | (7.5) |

где Нр – норматив расходов на реализацию

Норматив расходов на реализацию был принят 3%.

Размер расходов на реализацию составил:

Полная сумма затрат на разработку программного средства представлена в таблице 7.2.

Таблица 7.2 – Полная сумма затрат на разработку программного средства

|  |  |
| --- | --- |
| Наименование статьи затрат | Сумма, р. |
| Основная заработная плата разработчиков | 3 487,42 |
| Дополнительная заработная плата разработчиков | 348,724 |
| Отчисления на социальные нужды | 1 342,65 |
| Прочие расходы | 1 046,226 |
| Расходы на реализацию | 104,622 |
| Общая сумма инвестиций (затрат) на разработку | 6 329,642 |

## ЗАКЛЮЧЕНИЕ

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

## ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода

## ПРИЛОЖЕНИЕ Б

*(обязательное)*

Спецификация

## ПРИЛОЖЕНИЕ В

*(обязательное)*

Ведомость документов