ГБОУ ЛИЦЕЙ №470 КАЛИНИСКОГО РАЙОНА

Проектная работа

По теме: «Система и интерфейс автоматизации умного дома»

Работу выполнил:

Рязанов Даниил Дмитриевич

ученик 11 «А» класса

ГБОУ лицея № 470 Калининского района Санкт-Петербурга

Научные руководители работы:

Григорик Алексей Русланович

учитель Информатики и ИКТ

Погорелов Константин Алексеевич

учитель Информатики и ИКТ

Санкт-Петербург

2024-2025

|  |  |
| --- | --- |
| Тема проекта | Система и интерфейс автоматизации умного дома. |
| Автор проекта | Рязанов Даниил Дмитриевич, ученик 11 А класса |
| Руководители проекта | Григорик Алексей Русланович, учитель Информатики и ИКТ  Погорелов Константин Алексеевич, учитель Информатики и ИКТ |
| Тип проекта | Прикладной |
| Гипотеза проекта | Используя недорогие электронные компоненты, программирование и радиоэлектронику, можно создать многофункциональную систему умного дома. |
| Объект исследования | Системы автоматизации быта и производств. |
| Предмет исследования | Микроконтроллеры и цифро-аналоговая электроника, программная инженерия, пользовательские интерфейсы. |
| Проблема проекта | Умный дом – одна из технологий автоматизации, которая входит в жизнь общества, наравне с автоматизацией производства и бизнеса. Несмотря на наличие продвинутых систем умного дома, такие технологии среди многих людей пока не пользуются большой популярностью.  Готовые системы умного дома имеют ограниченный функционал и высокую стоимость. Разработка собственной системы умного дома требует значительных затрат на предметное изучение программирования, радиоэлектроники, приобретение разного рода профессиональных навыков, а также времени на создание электронных схем и плат. |
| Актуальность проекта | На рынке представлены разнообразные решения для создания умных домов. Данный проект позволяет реализовать систему самостоятельно и предлагает ряд готовых инструментов и решений. Самостоятельная сборка системы позволяет реализовать только необходимый функционал и снизить денежные расходы. |
| Цель проекта | Разработать фреймворк, набор решений и инструментов, для создания универсальных систем автоматизации дома и производств. |
| Задачи проекта | 1. Создание инструментов и решений для разработки системы умного дома под любой функционал и условия.  2. Создание рабочего прототипа.  3. Разработка документации и рекомендаций по применению проекта.  4. Создание бренда и экосистемы. |
| Востребованность проекта | Решения и инструменты проекта упростят самостоятельную разработку системы умного дома, позволят спроектировать сеть модулей для многофункциональной системы и дадут возможность наладить коммерческое производство. |
| Этапы реализации проекта | Этап I. Проектирование автоматических модулей.   * + Определение устройств, процессов и параметров, которые необходимо контролировать.   + Проектирование электронной части модулей.   + Определение общей структуры всей системы.   Этап II. Проектирование ядра (бизнес-логики).   * + Дискретизация задач.   + Создание шаблонов процессов, задач и конфигураций.   + Создание шаблонов обработки данных и объектов.   + Создание протоколов взаимодействия и API.   Этап III. Разработка библиотеки.   * + Создание удобных инструментов, на основе ядра.   + Создание удобного интерфейса для программистов.   + Создание инструментов для автоматической конфигурации системы.   Этап IV. Объединение модулей в единую систему.   * + Настройка стабильной связи между модулями.   + Настройка протоколов передачи данных.   + Проектирование API модулей.   + Проектирование моделей взаимодействия между модулями.   + Настройка систем обработки ошибок.   Этап V. Взаимодействие с пользователем.   * + Определение методов взаимодействия с пользователем.   + Реализация удобных для пользователя методов управления системой.   + Разработка GUI.   Этап VI. Обучение системы саморегулированию.   * + Определение факторов, влияющих на поведение системы.   + Проектирование сценариев поведения системы.   + Создание нейронной сети, для автоматического регулирования системы.   Этап VII. Настройка системы.   * + Отладка всех датчиков и модулей.   + Оптимизация прошивок.   + Обучение нейронной сети.   + Тестирование системы на стабильность.   + Проверка обработки системой критических ситуаций. |
| Проектный продукт | <https://github.com/MrRyabena/SmartHomeSystem> |
| Ожидаемые результаты | Функционирующая система, база для разработки дальнейших систем и развития проекта. |

Оглавление

[**1.** **Введение** 5](#_Toc191581936)

[**1.1** **Микроэлектроника вокруг нас** 5](#_Toc191581937)

[**1.2** **Актуальность и обоснование проекта** 5](#_Toc191581938)

[**1.3** **Цели и задачи** 6](#_Toc191581939)

[**2.** **Обзор литературы** 7](#_Toc191581940)

[**2.1** **Alex Gyver** 7](#_Toc191581941)

[**2.2** **Виктор Петин** 7](#_Toc191581942)

[**2.3** **Нил Кэмерон** 7](#_Toc191581943)

[**2.4** **Макс Шлее** 7](#_Toc191581944)

[**3.** **Структура проекта** 7](#_Toc191581945)

[**4.** **Методы и этапы проектирования** 8](#_Toc191581946)

[**4.1** **Принцип работы** 8](#_Toc191581947)

[**4.2** **Этапы разработки** 10](#_Toc191581948)

[**4.3** **Подход к проектированию** 12](#_Toc191581949)

[**4.4** **Функции** 13](#_Toc191581950)

[**5.** **Выбор компонентов** 14](#_Toc191581951)

[**5.1** **Микроконтроллеры** 14](#_Toc191581952)

[**5.2** **Датчики** 17](#_Toc191581953)

[**5.3** **Силовые компоненты** 18](#_Toc191581954)

[**5.4** **Обвязка** 19](#_Toc191581955)

[**6.** **Программная часть** 21](#_Toc191581956)

[**6.1** **Семантика *Smart Home System*** 21](#_Toc191581957)

[**6.2** **Инструменты и среды разработки** 21](#_Toc191581958)

[**6.3** **Компоненты ядра** 22](#_Toc191581959)

[**7.** **Объединение модулей в единую систему** 24](#_Toc191581960)

[**7.1** **Организация соединения** 24](#_Toc191581961)

[**7.2** **Поиск модулей в сети** 25](#_Toc191581962)

[**7.3** **API** 26](#_Toc191581963)

[**8.** **Взаимодействие с пользователем** 26](#_Toc191581964)

[**8.1** **Кнопки и датчики** 26](#_Toc191581965)

[**8.2** **Telegram bot** 27](#_Toc191581966)

[**8.3** **Графическое приложение** 27](#_Toc191581967)

[**8.4** **Голосовое** **управление** 27](#_Toc191581968)

[**9.** **Заключение** 28](#_Toc191581969)

[**10.** **Список литературы** 29](#_Toc191581970)

[**Приложение** 30](#_Toc191581971)

# **1. Введение**

## **1.1** **Микроэлектроника вокруг нас**

Мы живем в эпоху кремниевой лихорадки, когда на каждом шагу нас окружает огромное количество высокотехнологичных устройств с микрочипами внутри. Даже самые обыденные бытовые приборы, такие как электрочайник, светодиодная лампа или обогреватель, оказываются оборудованы умными «мозгами», превращая наши дома в настоящие технологические чудеса. Что уж там говорить про смартфоны, компьютеры, серверы… Такая массовая распространенность микроэлетроники привела к росту уровня технологий и увеличению масштабов производства. Появилось много микросхем, доступных обычным радиолюбителям. Сейчас, всего за цену одной порции шаурмы, можно заказать из Поднебесной модуль, способный решать задачи с эффективностью компьютера на i486 (процессор Intel 80486, 1989г.), при этом сам он поместится в чайной ложке!

Системой умного дома в наши дни мало кого можно удивить. На рынке представлены промышленные решения от именитых брендов. Компании готовы спроектировать и построить дом под любые запросы, либо предоставляют линейку модулей, соединяя которые можно собрать систему под свои потребности. Для таких продвинутых решений не нужны знания программирования и инженерии, все продумано разработчиками, требуется только подключить модули согласно инструкции и настроить их взаимодействие в удобном приложении. Линейки модулей достаточно обширные и позволяют контролировать температуру, освещение, включать и выключать приборы по расписанию, общаться с помощью чат-ботов и голосовых помощников и многое другое.

## **1.2 Актуальность и обоснование проекта**

Глядя на разнообразие существующих решений, может показаться, что создавать еще одну подобную систему совершенно бессмысленно. Однако, не все так однозначно. Посмотрим, какие же имеются плюсы для тех, кто намерен заняться самостоятельной разработкой?

* Цена. Готовые модули стоят достаточно дорого и собрать из них полноценную систему выйдет далеко не дешево.
* Функциональность. Индивидуальная разработка позволяет реализовать все полностью под себя, вывести доступ к любым настройкам и иметь возможность в любой момент исправить или модернизировать систему.
* Опыт. В процессе разработки нужно будет изучить программирование на низком и высоком уровнях, приобрести знания по радиоэлектронике, созданию электронных схем и плат, пайке, 3D-моделированию и печати и пр.
* Возможность наладить коммерческое производство. Иметь в портфолио объемный проект.
* В системе собраны алгоритмы, применение которым можно найти в других проектах и сферах деятельности.
* Проект с открытыми исходным кодом и материалами будет полезен для других разработчиков, с которыми возможна совместная работа и эффективное развитие.

Да, такие мысли приходили в голову многим, и аналогичных проектов с открытым кодом и подробной документацией множество на просторах интернета. Все они достаточно разнообразны и написаны, в основном, под конкретную ситуацию. *Smart Home System* предлагает набор инструментов для создания автоматизированных систем под любые условия и задачи. Таким образом, проект рассчитан на использование в разных сферах.

## **1.3 Цели и задачи**

**Основная цель проекта**

Разработать комплекс решений и инструментов для создания универсальных систем умного дома.

**Задачи проекта**

1. Создать функциональные инструменты и решения, которые позволят разрабатывать системы автоматизации, адаптируемые под различные функциональные требования и условия эксплуатации.

2. Составить оптимальную подборку электронных компонентов, необходимых для реализации автоматизированных систем.

3. Спроектировать аппаратную часть, разработать принципиальные электрические схемы.

4. Разработать кроссплатформенное программное ядро, обеспечивающее гибкость и масштабируемость системы.

5. Создать эффективные и надежные прошивки для микроконтроллеров, обеспечивающие управление и взаимодействие с компонентами системы.

6. Разработать графическое пользовательское приложение (GUI) для управления системой, совместимое с операционными системами Linux, Windows и Android, обеспечивающее интуитивно понятный интерфейс и удобство использования.

7. Создать специализированные модули и устройства.

8. Подготовить подробные руководства и документацию, которые помогут пользователям и разработчикам в установке, настройке, разработке и эксплуатации системы.

# **2. Обзор литературы**

## **2.1 Alex Gyver**

Во время разработки проекта самыми главными стали материалы, предоставленные в открытом доступе известным в кругах самодельщиков и ардуинщиков блогером AlexGyver. На его сайте (https://AlexGyver.ru) есть немало полезных уроков, гайдов и статей по микроконтроллерам. В *Smart Home System* многие нюансы не представлены подробно, однако почти все они в деталях грамотно изложены в уроках Gyver’a. Вместе со своим помощником(ами) Alex создал базу легких, оптимизированных и качественно сделанных библиотек, которые используются в *Smart Home System*: https://github.com/GyverLibs.

Все материалы и разработки Гайвера можно смело рекомендовать для изучения и использования. Большое ему спасибо за огромный и безвозмездный вклад в сообщество программистов-радиолюбителей.

## **2.2 Виктор Петин**

Петин В. А. "Создание умного дома на базе Arduino" – в книге подробно рассматривается конструирование умного дома на Arduino Mega (atmega) и NodeMCU (esp8266). Обстоятельно описаны все этапы: от установки ПО, до подключения датчиков, управления нагрузкой и интернета вещей. Даны примеры подключения практически всех существующих типов датчиков.

## **2.3 Нил Кэмерон**

Neil Cameron. «Electronics Projects with the ESP8266 and ESP32: Building WebPages, Applications, and WiFi Enabled» – в книге приведен ряд полезных примеров и интересных проектов. Есть переводной вариант от российских издательств.

## **2.4 Макс Шлее**

Макс Шлее. "Qt 5.10. Профессиональное программирование на C++" – книга-руководство по использованию Qt, созданию десктопных и мобильных приложений. Её рекомендуется освоить для разработки GUI-приложений, используемых в *Smart Home System.*

# **3. Структура проекта**

Проект выложен в репозитории на GitHub и доступен по ссылке: [https://github.com/MrRyabena/*Smart Home System*](https://github.com/MrRyabena/SmartHomeSystem).

Структура репозитория:

* **SHSdocumentation** – документация к проекту.
* **Code** – документация к компонентам, микроконтроллерам, техническим устройствам.
* **Hardware** – документация по микроконтроллерам, приборам и электронным компонентам.
* **ProjectOverview** – обзор проекта, особенности разработки, методы и подходы.
* **Usage** – руководства по сборке библиотек и использованию *Smart HomeSystem*, информация для пользователей и разработчиков.
* **pitch** – презентации проекта и вспомогательные файлы.
* **schemes** – схемы и чертежи.
* **SHSlogos** – логотипы и символика проекта.
* **src** – исходный программный код.
* **debugging\_sketches** – наброски отладочных прошивок.
* **examples** – примеры использования программного кода.
* **SHSapp** – разработки графических приложений.
* ***SHScore* –** ядро (бизнес-логика), на основе которого разрабатываются библиотеки, приложения, прошивки и остальное ПО.
* **SHSdevice** – устройства, которые можно добавлять в модули системы.
* **SHSlibrary** – SHSlibrary – библиотека с удобными инструментами, основанная на ядре.
* **SmartModules –** [beta] устройства и модули.
* **SmartModulesAPI –** [beta] команды устройств и модулей.
* **make.sh** – Shell-скрипт для сборки всех исходников проекта (см. документацию: build manual).

# **4. Методы и этапы проектирования**

## **4.1 Принцип работы**

Все много раз слышали, что работа электроники основана на принципе нулей и единиц (Булева алгебра), где сами значения “0” и “1” – на первый взгляд, очень абстрактные понятия “нет сигнала” и “есть сигнал”. Чтобы понять как, основываясь на этом принципе, построить умный дом, надо копнуть чуть поглубже. Рассмотрим упрощенную схему модуля (**см. Приложение 1**).

Первое что нужно – это подать на схему питание. Линий питания две:

* COM (GND, VSS, земля) – общий вывод питания, относительно него измеряются все остальные потенциалы в схеме.
* +V (VСС, VIN) – положительная линия питания. Их может быть несколько, рассчитанных на разные напряжения, например, +12V, +5V, +3.3V. Еще бывает отрицательное напряжение (относительно GND, опять же), но с ним обычно работают усилители и компараторы. В большинстве архитектур ЭВМ для логических цепей оно не применяется.

Вторая часть схемы, которая будет управлять модулем – это набор логических элементов. Наш дом – умный, поэтому схема из таких элементов будет непростая. Собрать самостоятельно ее, используя лишь базовые компоненты (резисторы, транзисторы, диоды…) крайне сложно, объемно и не рационально, да к тому же работать стабильно она вряд ли будет. К счастью, проблему уже давно решили и все необходимые цепи собраны в одном небольшом радиокомпоненте – микросхеме.

**Интегральная схема (микросхема)** – это электронная схема, помещенная на полупроводниковой (чаще всего кремниевой) подложке, с помощью фотолитографии. Небольшой корпус может содержать внутри как несложный набор логических элементов, так и целый процессор или контроллер, последний нам и нужен.

**Микроконтроллер** – это микросхема, которая содержит в себе процессор, ОЗУ, ПЗУ и периферийные устройства. Это целый небольшой компьютер, который может выполнять математические операции и управлять другими устройствами с помощью периферии.

Чтобы микроконтроллер мог принимать и выводить какие-то сигналы он оснащен выводами (контактами, пинами) с интерфейсом GPIO (general-purpose input/output). Такие пины могут работать в двух режимах: INPUT (вход) и OUTPUT (выход).

* В режиме INPUT микроконтроллер сравнивает входящий сигнал с землей (GND) и принимает его за 1, если его потенциал больше GND.
* Аналогично в режиме OUTPUT микроконтроллер может формировать на определенном контакте "0" или потенциал, равный его напряжению питания.

Теперь логический сигнал от микроконтроллера надо усилить с помощью силовой части схемы (например, транзистора или реле), что позволит управлять нагрузкой (светом, отоплением, чайником…) в режиме on/off (вкл/выкл).

Для того, чтобы управлять мощностью (интенсивностью работы) нагрузки (т.е. яркостью света, температурой нагревателя), необходимо регулировать подаваемое на нее напряжение. В цифровой электронике для этого применяется PWM (ШИМ – широтно-импульсная модуляция). Проще говоря, микроконтроллер очень быстро включает и выключает нагрузку на разные микропромежутки времени, а за счет ее инертности получается плавное регулирование.

Выше был описан принцип работы цифровой электроники, но в арсенале многих микроконтроллеров есть блоки для работы с аналоговой электроникой – ADC (АЦП) и DAC (ЦАП).

* **Аналого-цифровой преобразователь** позволяет микроконтроллеру измерять потенциал входного сигнала в диапазоне от 0, до опорного напряжения (либо задается от отдельного источника, либо совпадает с напряжением питания микроконтроллера) с некоторой точностью, которая зависит от разрядности АЦП. Он используется для считывания информации с датчиков, которые за счет физических эффектов (фотоэффектов, термоэффектов, эффекта Холла и пр.) изменяют напряжение на своем выходе. (Прим. цифровые датчики имеют встроенный АЦП и микроконтроллер для передачи информации по интерфейсам связи).
* **Цифро-аналоговый преобразователь** позволяет изменять потенциал сигнала в некотором диапазоне, он обычно служит для звуковых сигналов либо в качестве "цифровых потенциометров".

Теперь, вдохновившись идеей и понимая принцип работы, можно попробовать создать свою систему. На одной теории дом не построишь, поэтому разработка требует постоянных экспериментов, которые подробно описаны в основной части документации.

## **4.2 Этапы разработки**

Первым делом необходимо создать физические устройства для решения задач: управления светом, температурой, измерения параметров микроклимата и пр. Затем реализуются методы взаимодействия и управления. В итоге – получаем автоматизированную систему. Последний шаг в проектировании – "научить" систему обрабатывать данные и регулировать все устройства корректно в любой ситуации с минимальным вмешательством пользователя.

* **Этап I.** Проектирование автоматических модулей
* Определение устройств, процессов и параметров, которые необходимо контролировать.
* Проектирование электронной части модулей.
* Определение общей структуры всей системы.
* **Этап II.** Проектирование ядра (бизнес-логики).
* Дискретизация задач.
* Создание шаблонов процессов, задач и конфигураций.
* Создание шаблонов обработки данных и объектов.
* Создание протоколов обработки и передачи данных.
* Создание протоколов взаимодействия и API.
* **Этап III.** Разработка библиотеки.
* Создание удобных инструментов, на основе ядра.
* Создание удобного интерфейса для программистов.
* Создание инструментов для автоматической конфигурации системы.
* **Этап IV.** Объединение модулей в единую систему.
* Настройка стабильной связи между модулями.
* Настройка протоколов передачи данных.
* Проектирование API модулей.
* Проектирование моделей взаимодействия между модулями.
* Настройка систем обработки ошибок.
* **Этап V.** Взаимодействие с пользователем.
* Определение методов взаимодействия с пользователем.
* Реализация удобных для пользователя методов управления системой.
* Разработка GUI.
* **Этап VI.** Обучение системы саморегулированию.
* Определение факторов, влияющих на поведение системы.
* Проектирование сценариев поведения системы.
* Создание нейронной сети, для автоматического регулирования системы.
* **Этап VII.** Настройка системы.
* Отладка всех датчиков и модулей.
* Оптимизация прошивок.
* Обучение нейронной сети.
* Тестирование системы на стабильность.
* Проверка обработки системой критических ситуаций.

## **4.3 Подход к проектированию**

При проектировании выявлено два способа реализации *Smart Home System*:

1. Создается один большой модуль, который включает в себя несколько микроконтроллеров, их обвязку, систему питания и подключения устройств. Он реализует все необходимые функции.
2. Создается несколько небольших модулей, каждый из которых контролирует один небольшой блок устройств и процессов, имеет собственную систему питания. Все модули связываются между собой по WiFi и образуют единую систему.

Преимущества первого подхода заключаются в удобстве обслуживания: все находится в одном корпусе, не нужно бегать по всему дому, чтобы что-то подключить или поправить. На этом, как оказалось, плюсы заканчиваются, начинаются проблемы. При создании такого модуля получается большая печатная плата, очень большая и сложная, возникает много трудностей при пайке и выявлении ошибок. Самая главная проблема – большое количество проводов, которые необходимо протянуть по всему дому. Они постоянно отовсюду вылезают, мешают, стоят дорого и наводят помехи друг на друга.

Второй способ оказался более практичным. Мы можем постепенно создавать небольшие схемы, добавлять, менять или переделывать их. Сигнальные и силовые линии не нужно тянуть по всему дому, все получается аккуратно и компактно.

Первый способ имеет смысл только для реализации каких-то небольших систем, по типу контроллера теплицы или какого-то небольшого помещения. Поэтому *Smart Home System* основан на втором способе.

## **4.4 Функции**

Выше уже описаны цели и идеи их реализации, здесь речь пойдет более конкретно о задачах, которые можно решить с помощью *Smart Home System*. Проект предоставляет широкий спектр инструментов, поэтому варианты его применения никак не ограничены. Вот лишь некоторые примеры:

* Первое что приходит в голову, когда речь идет об умном доме – автоматическое включение и регулировка освещения. Свет должен включаться, когда это необходимо, выключаться, но только тогда, когда он действительно никому не нужен и плавно регулироваться, постоянно поддерживая одинаковое значение яркости. Еще одна дополнительная функция точно не оставит равнодушными людей, которым приходится рано вставать – будильник-рассвет – комната плавно заливается теплым светом, имитируя восход солнца.

Задача достаточно простая и реализуется даже без дополнительных библиотек.

* Если уж речь зашла об освещении, то можно добавить красоты и технологичности – RGB- подсветку. Тут необходимы инструменты для работы со светодиодами типа RGB (обычные четыреxвыводные, где требуется только регулировать яркость каждого канала цвета) и ARGB (адресные светодиоды, которые позволяют управлять цветом каждого элемента ленты, независимо от всех остальных). Если первыми можно управлять «вручную», то для адресных точно потребуются дополнительные библиотеки. Когда есть хорошие решения, писать заново библиотеки смысла особого нет, поэтому, используя готовую основу в виде пары библиотек, останется только создать различные эффекты и режимы работы.
* Климат-контроль. С помощью современных датчиков можно с легкостью отслеживать главные факторы домашнего микроклимата: температуру, уровень влажности и концентрацию углекислого газа. Обрабатывая данные, можно настроить регулировку котла или подачу горячей воды, работу увлажнителей воздуха и открывать форточки (или включать систему вентиляции).
* Метеостанция. Измерение температуры, влажности, скорости и направления ветра, атмосферного давления. Обработка полученных результатов и составление прогноза погоды на ближайшее время, рекомендации по одежде и ожидания на день.
* Контроллер теплицы. В доме наверняка есть комнатные растения, а может быть целая теплица или сад. Необходимо реализовать автополив, контроль влажности почвы и досвечивание растений.
* IoT. Реализовать возможность получать данные из интернета, выкладывать их, обмениваться с пользователями различных чат-ботов и другие паттерны Интернета Вещей.
* Smart Bar. Холодильная камера на элементах Пельтье, для охлаждения напитков, чайник с поддержанием температуры воды, автоналиватор напитков.
* Аудиосистема. Качественное проигрывание звука, переключение между колонками, автоматическое воспроизведение, эффекты полного погружения.
* Фотостудия. Система светильников и приборов для спецэффектов, управляющаяся через приложение. Такая разработка будет крайне полезна фотографам, видеооператорам и художникам по свету.

# **5. Выбор компонентов**

## **5.1 Микроконтроллеры**

Для управления всеми модулями от микроконтроллеров требуется наличие достаточного количества выводов GPIO, поддержка (желательно аппаратная) интерфейсов UART, I2C, SPI, PWM, WiFi, и встроенный ADC.

Сейчас на рынке представлено много линеек микроконтроллеров, доступных простым радиолюбителям:

* Компания Atmel представляет семейства на архитектуре AVR – Attiny и Atmega. Первые микроконтроллеры совсем крохотные и использовать их можно только для небольших задач, например, управления одним светильником и связи с остальными. Семейство Atmega более известно под маркой Arduino, т.к. лежит в основе большинства их плат. Эти камни уже могут похвастаться большим арсеналом GPIO, ADC, высоким выходным током с пина. На их основе можно собирать полноценные модули, не хватает только WiFi или другой беспроводной связи.
* Китайцы, в лице Espressif Systems разработали свою линейку микроконтроллеров с WiFi на борту, flash-памятью и достаточно мощными (для задач умного дома) вычислительными ядрами.
* В сети много гайдов по созданию сервера умного дома на основе Raspberry Pi. Это семейство представляет собой целые миниатюрные компьютеры на архитектуре ARM, позволяющие работать с операционной системой Linux и выводить изображение на монитор. Тема на самом деле достаточно крутая и найти им применение не составит проблем – можно собрать домашний сервер и это выйдет компактнее, чем переделывать какой-нибудь старый компьютер.

Мощности Малинки для *Smart Home System* будут избыточные, да и стоят такие платы достаточно дорого.

В итоге выбор пал на 3 основных микроконтроллера: Atmega328P, ESP8266, ESP32.

**5.1.1 ATmega328P**

Микроконтроллер удобен в использовании, крайне неприхотлив в эксплуатации, имеет много аналоговых пинов, аппаратную поддержку ШИМ. WiFi на борту нет, поэтому его придется связывать с ESP по UART.

В проекте используется “голый” микроконтроллер (особенно там, где нужна компактность), либо платы Arduino Nano, Arduino Pro Mini. (**см. Приложение 2**)

|  |  |
| --- | --- |
| **Parametr** | **Value** |
| CPU type | 8-bit AVR |
| CPU speed | 16 MHz |
| Сurrent consumption | <<100 mA |
| Flash memory | 32 KB |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| I/O pins | 23 |
| Max current from the pin | 40 mA |
| PWM pins | 6 |
| ADC pins | 8 channels, 10-bit |
| DAC | none |
| External interrupts | 3 |
| I2C | 1 |
| I2S | none |
| SPI | 1 |
| UART | 1 |
| WiFi | none |
| Bluetooth | none |
| Operating Voltage Range | 1.8 – 5.5 V |

**5.1.2 ESP8266**

Имеет WiFi, достаточно много оперативной и постоянной памяти – шустрый процессор. Хорошо подходит для взаимодействия с другими цифровыми устройствами и интернетом. Из минусов: слабый ток с пина (12 mA), рабочее напряжение 3.3 V, нет аппаратной поддержки PWM (реализована программно) и всего 1 канал АЦП. “Голый” микроконтроллер – ESP12F. Платы – NodeMCU или WemosD1 mini. (**см. Приложение 3**)

|  |  |
| --- | --- |
| **Parametr** | **Value** |
| CPU type | Xtensa L106, 32 bit |
| CPU speed | 80/160 MHz |
| Сurrent consumption | 300 mA |
| Flash memory | 1-16 MB |
| SRAM | 82 KB |
| EEPROM | 4 KB |
| I/O pins | 11 |
| Max current from the pin | 12 mA |
| PWM pins | 10 (software!) |
| ADC pins | 1 |
| DAC | none |
| External interrupts | 10 |
| I2C | 1 (software) |
| I2S | 1 |
| SPI | 1 |
| UART | 1.5 |
| WiFi | 802.11 b/g/n 2,4 GHz |
| Bluetooth | none |
| Operating Voltage Range | 2.2–3.6V |

**5.1.3 ESP32**

Имеет двухъядерный процессор, большой объем памяти, WiFi, PWM, аппаратную поддержку цифровых протоколов, сенсорные пины, много каналов ADC, DAC и выход звукового сигнала. Мощный чип, на котором будет собран сервер. В проекте используется “голый” микроконтроллер, но также можно взять платы (аналогично ESP8266). (**см. Приложение 4**)

|  |  |
| --- | --- |
| **Parametr** | **Value** |
| CPU type | Xtensa LX6, dual-core, 32 bit |
| CPU speed | 160/240 MHz |
| Сurrent consumption | 300 mA |
| Flash memory | 1-16 MB |
| SRAM | 512 KB |
| EEPROM | 4 KB |
| I/O pins | 34 |
| Max current from the pin | 12 mA |
| PWM pins | 16 |
| ADC pins | 18, 12-bit |
| DAC | 2, 8-bit |
| External interrupts | 34 (10 touch sensors) |
| I2C | 2 |
| I2S | 2 |
| SPI | 4 |
| UART | 3 |
| WiFi | 802.11 b/g/n 2,4 GHz |
| Bluetooth | v4.2 BR/EDR and BLE |
| Operating Voltage Range | 2.2–3.6V |

## **5.2 Датчики**

Чтобы *Smart Home System* могла получать информацию об окружающей среде, в ней предусмотрены датчики, позволяющие измерять разнообразные параметры. Датчики можно разделить на две категории: цифровые и аналоговые, о чем было сказано выше. Цифровые в основном подключаются по шинам I2C и SPI, аналоговые – к АЦП через обвязку (если требуется).

**5.2.1 Temperature**

* **Терморезистор (термистор)** – самый простой способ измерить температуру окружающей среды с приемлемой точностью. Изменяет свое сопротивление с изменением температуры. Подключается в схеме делителя напряжения к ADC и обрабатывается по уравнению Стейнхарта- Харта.
* **DS18B20** – цифровой, достаточно точный датчик температуры, подключается по интерфейсу I2C.
* **Термопара + MAX6675** – термопара и драйвер для измерения ее показаний, измерение температуры в несколько сотен градусов.

**5.2.2 Humidity**

* **DHT22** – цифровой датчик температуры и влажности воздуха. Не особо точный, в системе не используется.
* **HTU21D** – гораздо более точный датчик температуры и влажности, рекомендуется использовать его.

**5.2.3 Pressure**

* **BME280/BMP280** – цифровой датчик измерения атмосферного давления. По сути, это целая метеостанция, т.к., помимо давления, измеряет температуру и влажность воздуха (для последнего нужно брать BME280).

**5.2.4 Carbon dioxide**

* **MH-Z19b** – датчик углекислого газа.

**5.2.5 Illumination**

* **Фоторезистор** – аналоговый компонент, изменяет свое сопротивление, в зависимости от интенсивности падающего на него света. Подключается к ADC в схеме делителя напряжения.

**5.2.6 Other**

* **Кнопка/энкодер** – можно организовать удобное управление системой.
* **TTP223** – сенсорная кнопка. Подключается и работает точно так же, как обычная механическая. Можно подпаяться к плате и вывести “антенну” в удобное место. Пробивает толщину в ~0.5mm.
* **Потенциометр** – подключается к ADC, через него тоже можно регулировать настройки системы.

## **5.3 Силовые компоненты**

По управлению мощной нагрузкой есть хорошая статья (https://alexgyver.ru/lessons/arduino-load), в ней написано грамотно и подробно, ниже лишь указаны некоторые детали.

**5.3.1 Транзисторы**

* **IRLB8743** – полевой транзистор (мосфет) для вкл./выкл. или ШИМ управления нагрузкой в несколько ампер до 30 V. Может быть не самый лучший вариант (высоковатое сопротивление в открытом состоянии, сильно греется при большом токе или высокой частоте), но дешевый и практичный. Заказать лучше всего сразу пачку-две в Поднебесной.
* **2N7000** – небольшой полевой транзистор для слаботочных цепей (тянет катушку небольшого реле или клапана, кусочки светодиодных лент).
* **BC547** – небольшой биполярный транзистор, полезен для подтяжки цепей к земле, управления слаботочными нагрузками.

**5.3.2 Симисторы**

**ВНИМАНИЕ! Работа с высоким напряжением опасна для жизни! Следует строго соблюдать все меры предосторожности и убеждаться, что цепи отключены от питания, а все конденсаторы разряжены!**

В *Smart Home System* основная часть нагрузок и модулей работает от низковольтного напряжения 3.3V–24V. Не рекомендуется внедрять в систему высоковольтные цепи, однако, если есть необходимость, в системе разработаны решения для управления нагрузкой с помощью реле или симистора. Все низковольтные цепи необходимо гальванически развязать с высоковольтными (с помощью трансформаторов или оптронов).

* **BTA208X**-1000С – хорошо подойдет для регулирования осветительных приборов.
* **BTA41**-600BRG – очень мощный симистор, когда-то можно было купить в 8 раз дешевле.

## **5.4 Обвязка**

**5.4.1 Резисторы**

* **150Om 0.25W** – для ограничения тока с пинов микроконтроллеров c логическим уровнем 5V (Atmega328P, ток с пина ~40mA).
* **330Om 0.25W** – для ограничения тока с пинов микроконтроллеров с логическим уровнем 3.3V (ESP8266, ESP32, ток с пина ~10mA).
* **10kOm 0.25W** – для логической подтяжки пинов.

**5.4.2 Конденсаторы**

* **0.1uF** – керамический конденсатор, нужно ставить как можно ближе к микроконтроллерам, для поглощения помех и пульсаций питания.
* **470–2000uF** – стоит ставить рядом с микроконтроллерами и другими чувствительными устройствами (адресными светодиодными лентами, датчиками…), особенно если используется некачественный блок питания или в этой же цепи есть мощная нагрузка.

**5.4.3 Диоды**

* **1N4007** – практически на все случаи жизни.
* **1N5817/1N5819** – диоды Шоттки, могу пригодиться, когда мешает падение напряжения на классическом диоде (например, для реализации защиты схемы от неправильной полярности входного питания, без потерь напряжения, что особенно критично для 3.3V).

**5.4.4 Стабилитроны**

* **1N4728** – 3.3V, подойдет для выравнивания логических уровней.
* **BZX55C5V1** – 5V, можно сделать источник опорного напряжения.

**5.4.5 Стабилизаторы напряжения**

* **LD1117/L78L33** – 3.3V 1.5/0.1A. L7805/L78L05 – 5V 1.5A/0.1A. L7812C – 12V.
* **LM317** – регулируемый выход ~1–24V.

**5.4.6 Оптопары (оптроны)**

* **АОТ101ГС** – транзисторная, двухканальная. Удобна для гальванической развязки цепей связи типа UART.
* **PC814/FOD814** – транзисторная, хорошо подходит для цепей постоянного тока.
* **PC817/EL817C** – симисторная, хорошо подходит для цепей переменного тока.
* **MOC3063** – симисторная, с детектором нуля (для управления нагрузкой в режиме on/off).
* **MOC3021** – симисторная, без детектора нуля (для диммера).

**5.4.7 Регистры**

* **SN74HC595** – восьмиканальный сдвиговый регистр, можно использовать как расширитель пинов.
* **CD4051** – аналоговый восьмиканальный мультиплексор, можно использовать как переключатель аналоговых входов/выходов (особенно актуально для ESP8266).

# **6. Программная часть**

## **6.1 Семантика *Smart Home System***

Одна из важнейших задач, стоящих перед разработчиками – создать правильные абстракции, которые удобно использовать и модернизировать. *Smart Home System* разделен на несколько уровней абстракции, позволяющих организовать удобный программный код:

1. **Component** – неделимая составляющая. Под этим термином подразумевается класс, процесс, вычислительный блок или малое устройство (отдельный датчик, соединение, протокол...).
2. **Device** – устройство системы, отвечающее за выполнение определенной задачи (физической или виртуальной). Обычно устройство состоит из нескольких компонентов, например, датчиков и нагрузок, и вычислительной (программной) логики, которая обрабатывает данные и формирует управляющие сигналы.
3. **Module** – самостоятельный узел *Smart Home System*, состоящий из одного или нескольких микроконтроллеров, блока питания, обвязки для управления нагрузками и датчиками для сбора данных. Модуль может объединять внутри себя одно или несколько устройств. Отдельные программы, например, сервер или мобильное приложение, также являются модулями, хотя и не имеют физической основы (имеется ввиду, что телефон или компьютер непосредственно не являются элементами системы).

## **6.2 Инструменты и среды разработки**

**6.2.1 Языки программирования**

Основная часть исходного кода написана на языке C++. В редких случаях используется чистый C и ассемблерные вставки. При этом *Smart Home System* предлагает хорошо оптимизированный под микроконтроллеры код, использующий современные паттерны и инструменты C++.

Для разработки графических приложений используется фреймворк Qt. Он предоставляет возможность программировать на C++ и Qml – специально разработанном языке графической разметки. Ядро (*SHScore*) *Smart Home System* интегрируется в Qt, что позволяет применять общие алгоритмы и удобно организовывать взаимодействие элементов системы.

**6.2.2 Среды разработки**

Для работы с C/C++ рекомендуется использовать Microsoft Visual Studio. В качестве редактора кода лучше всего подходит Cursor – аналог Visual Studio Code со встроенными нейросетями, ускоряющими процесс разработки и отладки.

Работа с микроконтроллерами требует специальных компиляторов и загрузчиков. Наиболее доступными средами разработки являются Arduino IDE и Platform IO. Имея возможность, стоит воспользоваться более профессиональными инструментами.

При работе с Qt рекомендуется использовать специально созданную для этой библиотеки среду разработки QtCreator. Для разработки мобильных приложений под Android потребуется дополнительно установить Android Studio.

**6.2.3 Инструменты**

Проект использует систему контроля версий Git.

Для сборки программного кода используются Shell-скрипты и система сборки CMake. Скрипты Shell рассчитаны на запуск в командной строке Linux, поэтому пользователям Windows необходимо включить компонент WSL, позволяющий открыть виртуальную Linux-среду.

*Более подробно ознакомиться с рекомендуемым для разработки ПО и инструкцией по сборке проекта можно в документации в файле* **build\_manual***.*

## **6.3 Компоненты ядра**

*SHScore* – главная составляющая *Smart Home System*. Ядро отвечает за абстракции, алгоритмы обработки и передачи данных, программные инструменты и интерфейсы. Кроссплатформенная реализация позволяет применять одни и те же алгоритмы на разных устройствах, поэтому разработчикам не требуется каждый раз дублировать код или разрабатывать решения заново.

В текущей версии *Smart Home System* ядро получило глобальное обновление. Его функционал был изменен, дополнен и протестирован. Ниже описаны основные компоненты, которые предлагает для разработчиков *SHScore*.

Чтобы избежать возможные конфликты имен между кодом *SHScore* и другими разработками, все классы и функции обернуты в namespace shs (пространство имен shs). Оно также позволяет другим разработчикам четко понимать происхождение используемого ими интерфейса.

Со схемой архитектуры *SHScore* можно ознакомиться в **Приложении 5** или в папке *schemes* в репозитории проекта на гитхаб (https://github.com/MrRyabena/*Smart Home System*)

**6.3.1 Containers**

Контейнер – тип, позволяющий инкапсулировать в себе объекты других типов. *SHScore* предоставляет 3 важных контейнера, активно используемых при разработке автоматизированных систем:

* **ByteCollector** – контейнер, представляющий любые типы данных в виде последовательности байтов. Он активно применяется в протоколах передачи данных. ByteCollector имеет собственные итераторы и совместим с функциями STL.
* **SimpleFixed** – класс, хранящий в целочисленной переменной вещественное число. Преобразует числа с плавающей точкой в фиксированные и обратно. Предусмотрена возможность изменить точность.
* **SortedBuf** – класс-оболочка, поддерживающий std::vector в сортированном виде. Выполняет поиск и вставку элементов за O(logN).

**6.3.2 Protocols**

* **SHSDTP** – *Smart Home System* *Data Transfer Protocol*, протокол передачи данных, предоставляющий мощный набор инструментов для поиска устройств в локальной сети (DTPdiscover), установления и поддерживания соединений с ними, упаковки и распаковки данных (DTPpacket), отправления и приема сообщений (DTPbus). Главный класс (shs::DTP) соединяет внутри себя шины данных и обработки API, что позволяет автоматически настроить все взаимосвязи для дистанционного управления устройствами.A
* **Network** – набор инструментов для работы с сетевыми протоколами. Имеет готовые реализации TCP и UDP серверов и клиентов.
* **API** – набор интерфейсов и команд для управления и взаимодействия с устройствами. Реализует команды устройств, виртуальные объекты и обработчики сообщений.

**6.3.3 OS**

Собственная кроссплатформенная операционная система позволяет управлять процессами и запускать их в параллельных потоках. Разработаны инструменты для работы со временем, что позволяет удобно организовывать паттерны событийно-ориентированного программирования. Большая часть функционала операционной системы на данный момент находится в разработке и требует отладки.

**6.3.4 System**

В этом разделе ядра реализованы основные абстракции системных компонентов и инструменты для работы с ними.

* **Load** – абстракция нагрузки, на основе которой создаются все классы, управляющие физическими или виртуальными устройствами. При создании пользовательских компонентов, необходимо наследоваться от этого класса. Регулирование нагрузки может происходить в разных режимах (например, SWITCH или PWM). Для дистанционного управления предусмотрены класс-интерфейс shs::LoadVirtual.
* **Sensor** – абстракция датчика, позволяющая производить измерения, обрабатывать их, и получать результат. Для работы с удаленных устройств, аналогично нагрузке, создан инструмент shs::SensorVirtual.
* **settings** – предусмотрены флаги и настройки компиляции. Для пользовательской настройки кода следует изучить соответствующую документацию.

**6.3.5 Algorithms**

* **CRC** – набор классов для подсчета контрольной суммы 8, 16 и 32 бит. Используется в протоколах передачи данных и файловой системе.
* **Random** – модуль для генерации случайных чисел.
* **shs\_algorithm –** набор шаблонных функций для работы с последовательностями данных.

**6.3.6 Utils**

* **debug** – макрофункции для вывода отладочной информации.
* **version** – структура, хранящая текущие версии компонентов проекта.

# **7. Объединение модулей в единую систему**

## **7.1 Организация соединения**

**7.1.1 Покрытие сети**

Все модули *Smart Home System* обмениваются между собой данными по WiFi. Для стабильной работы всех устройств необходимо обеспечить высокий уровень сигнала локальной сети для каждого модуля.

Если помещение небольшое и установлен мощный маршрутизатор (WiFi-роутер), то скорее всего никаких дополнительных действий не потребуется. В противном случае, проблему можно решить путем установки WiFi-repeater’ов. Они подключаются в сеть, увеличивают ее радиус покрытия и уровень сигнала.

* + 1. **Подключение** **модулей**

После настройки маршрутизатора, остается подключить и настроить сами микроконтроллеры. Для этого в *SHScore* есть набор статических функций, реализованных в класс *shs::ControlWiFi*. Подробно о работе с сетью и подключении модулей следует изучить в документации или статье-руководстве по использованию проекта – system\_creation\_guide.

* + 1. **Сетевые протоколы**

**TCP/IP** – основной протокол передачи данных в Интернете. Через него организуется соединение между всеми модулями. Для этого в ядре ESP есть классы WiFiClient и WiFiServer. На их основе в *SHScore* реализованы собственные классы, которые, к тому же, поддерживают и другие платформы. Таким образом создан единый для всех платформ интерфейс для работы с сетевыми протоколами.

**UDP –** еще один сетевой протокол, поддерживаемый *Smart Home System*. Он выигрывает TCP/IP в скорости, но не гарантирует целостную доставку пакетов. Рекомендуется не примерять UDP без острой необходимости.

* 1. **Поиск модулей в сети**

После того как модули подключены к сети и функционируют, необходимо чтобы они нашли друг друга. Поскольку IP-адрес при меняется при каждом новом подключении устройства, необходим инструмент для поиска нужных модулей в сети. Можно, конечно, задать статическую адресацию в настройках роутера, но это требует дополнительных усилий и знаний от пользователя и не удобно, при добавлении новых модулей. Такой вариант может не сработать с некоторыми роутерами или мобильными точками доступа. Поэтому в *SHScore* разработан специальный инструмент для поиска устройств – *shs::DTPdiscover.* Он отправляет широковещательное сообщение о поиске модуля с соответствующим ID, в ответ на который искомый модуль отправляет свой IP-адрес.

* 1. **API**

Все устройства связаны и имеют доступ друг к другу. Чтобы они могли запрашивать и принимать данные, нужно определить соответствующие команды и обработчики для них. API каждого модуля состоит из перечисления (enum), где каждой команде соответствует числовой код. В пакет данных передается команда, а затем дополнительные параметры, если требуется. В таком же порядке данные и будут расшифровываться на стороне приемника. Поскольку вручную отправлять данные не очень удобно, создаются классы виртуальных объектов с переопределенными функциями. Они предоставляют интерфейс, аналогичный реальному устройству, однако работают с ним удаленно.

Для расшифровки команд на стороне управляемого модуля создается API-handler (обработчик API) и подключается к *shs::DTP,* который передает на обработку адресованные ему сообщения.

# **8. Взаимодействие с пользователем**

В *Smart Home System* реализованы четыре метода взаимодействия с пользователем:

1. Кнопки и датчики.
2. Чат-боты.
3. GUI (приложение).
4. Голосовое управление.

Каждый из них позволяет изменять набор параметров и регулировать систему.

* 1. **Кнопки и датчики**

Датчики движения позволяют оценить нахождение пользователя в пространстве и скорректировать поведение системы. Таким образом, когда пользователь вне дома, включается досвечивание растений, шумные приборы (например, вытяжки, фильтры, насосы).

Кнопки лучше всего расположить в наиболее удобных местах: у входов в комнаты, около рабочего места, рядом с кроватью. Таким образом, выходя из комнаты, можно дать команду перейти в ждущий режим, за рабочим местом – добавить света, а лежа в кровати – включить тихий режим и подготовиться ко сну.

* 1. **Telegram bot**

Разработка, которая позволяет не только управлять умным домом, но и пользоваться дополнительными функциями для составления расписания, напоминаний, получения различной информации. Кроме того, бот может предупредить об ухудшении погоды или присутствии посторонних рядом с домом.

Бот умеет показывать меню с кнопками, нажимая на которые можно очень быстро управлять системой. По специальным алгоритмам меню корректируются и высвечиваются актуальные на данный момент кнопки. Например – к вечеру меню предложит гасить свет и переводить систему в бесшумный режим; когда пользователь возвращается домой, меню может предложить кнопки для включения чайника, света, отопления и пр. (**см. Приложение 7**)

Бот запускается на сервере и имеет быстрый доступ к информации хранилища и управлению *Smart Home System*.

* 1. **Графическое приложение**

**GUI** – графический интерфейс пользователя, т.е. различные приложения и web-страницы. Один из самых удобных способов организовать управление системой – визуализировать графики и настройки, сделать меню для конфигурации системы и настройки связей. Разработка хорошего приложения требует много времени, но результат того стоит.

Приложение, ядро и инструменты для его разработки написаны на C++ с использованием фреймворка Qt и вспомогательного языка QML.

* 1. **Голосовое** **управление**

Построено на специальном контроллере распознавания голоса. Библиотека для него достаточно неудобная и нефункциональная, поэтому в ближайших версиях в проекте появится своя. Разработки уже ведутся.

Модуль имеет хранилище распознавателя и общее хранилище команд. Таким образом одновременно он сравнивает 7 команд, затем передает код распознанной команды, основываясь на ней, управляющий микроконтроллер загружает в распознаватель следующий набор команд. Таким образом можно строить цепочки сложных команд и организовать своего голосового помощника.

# **9. Заключение**

*Smart Home System* представляет собой масштабный проект, предлагающий удобные инструменты и руководства для создания автоматизированных систем умного дома. С каждой новой версией проект становится более доступным и удобным даже для пользователей, не обладающих глубокими знаниями в программировании и проектировании.

В ходе работы над проектом были достигнуты следующие результаты:

* Разработан комплекс решений и инструментов. Созданы универсальные инструменты, которые позволяют разрабатывать системы автоматизации, адаптируемые под различные функциональные требования и условия эксплуатации.
* Создано кроссплатформенное программное ядро, обеспечивающее гибкость и масштабируемость системы. Ядро позволяет применять одни и те же алгоритмы на различных устройствах, избегая дублирования кода.
* Разработаны эффективные и надежные прошивки для микроконтроллеров, а также графическое пользовательское приложение, совместимое с Linux, Windows и Android.
* Подготовлены подробные руководства и документация: Эти материалы помогут пользователям и разработчикам в установке, настройке и эксплуатации системы.

Проект имеет большой потенциал для дальнейшего развития и расширения. Подготовив линейку модулей и пользовательские интерфейсы, можно наладить коммерческое производство модульных систем. Многие инструменты проекта поддерживают индивидуальную работу, что позволяет использовать *Smart Home System* в отдельных проектах и устройствах, ускоряя процесс их разработки.

В ходе данной проектной работы удалось рассмотреть основные детали *Smart Home System*. Более подробно ознакомиться с проектом можно в GitHub-репозитории по ссылке: [https://github.com/MrRyabena/*Smart Home System*](https://github.com/MrRyabena/SmartHomeSystem) или qr-коду **(см. Приложение 8**)

# **10. Список литературы**

1. Петин, В. А. Создание умного дома на базе Arduino / В. А. Петин. — Москва : ДМК Пресс, 2018. — 180 с. ISBN 978-5-97060-620-9.

2. Cameron, N. Electronics Projects with the ESP8266 and ESP32: Building WebPages, Applications, and WiFi Enabled Devices / N. Cameron. — 1st ed. — Apress, 2020. ISBN-13: 978-1-4842-6336-5

3. Шлее, М. Qt 5.10. Профессиональное программирование на СПб.: БХВ-Петербург, 2018. – 1072 с. ISBN 978-5-9775-3678-3.

4. Головиц, Я. C++17 STL. Стандартная библиотека шаблонов / Я. Головиц. — СПб.: Питер, 2018. — 432 с. ISBN 978-5-4461-0680-6.

**Контактная информация**

Tg: <https://t.me/MrRyabena>

Mail: [daniilrazanov82349@gmail.com](mailto:daniilrazanov82349@gmail.com)

Блог разработки: https://t.me/SmartHomeSystem\_dev

Даниил Рязанов (Daniil Ryazanov)

# **Приложение**

|  |
| --- |
| **Приложение 1:** упрощенная схема модуля. |
| **Приложение 2:** схема GPIO микроконтроллера ATmega328P. |
| **Приложение 3:** схема GPIO микроконтроллера ESP8266. |
| **Приложение 4:** схема GPIO микроконтроллера ESP32. |
| **Приложение 5:** схема архитектуры *SHScore.* |
| **Приложение 6:** архитектура *SHSlibrary.* |
| **Приложение 7:** архитектура Telegram-бота. |
| **Приложение 8:** qt-код с ссылкой на GitHub-репозиторий *Smart Home System*. |
| **Приложение 9:** символика *Smart Home System* – логотип и подпись, стиль «mint». |
| **Приложение 10:** символика *Smart Home System* – логотип и подпись, стиль «green». |