Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: «Разработка системы умный дом»**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 31ИС-23** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **И.А. Гуров** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л.Б. Гусятинер** |

**Москва 2024**

Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 31ИС-23 Гуров Иван**

**ТЕМА: «Разработка системы умный дом»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

Оглавление

[Введение 4](#_Toc184915513)

[Глава 1. Теоретическая часть 5](#_Toc184915514)

[1.1 Описание предметной области 5](#_Toc184915515)

[1.1.1 Интернет вещей 5](#_Toc184915516)

[1.1.2 Основные компоненты 5](#_Toc184915517)

[1.1.3 Применение Интернет вещей 6](#_Toc184915518)

[1.1.4 Будущее Интернет вещей 6](#_Toc184915519)

[1.2 Описание существующих разработок 6](#_Toc184915520)

[1.3 Выбор инструментов 6](#_Toc184915521)

[Глава 2. Проектная часть 8](#_Toc184915522)

[2.1 Диаграмма прецедентов 8](#_Toc184915523)

[2.2 Проектирование предварительного дизайна 9](#_Toc184915524)

[2.3 Проектирование сценария 10](#_Toc184915525)

[2.4 Описание главного модуля 12](#_Toc184915526)

[2.5 Описание тестовых наборов модулей 18](#_Toc184915527)

[2.5 Отладка 24](#_Toc184915528)

[Глава 3. Эксплуатационная часть 25](#_Toc184915529)

[3.1 Руководство оператора 25](#_Toc184915530)

[3.1.1 Назначение программы 25](#_Toc184915531)

[3.1.2 Условия выполнения программы 25](#_Toc184915532)

[Список литературы и интернет-источников 28](#_Toc184915533)

[Приложение 30](#_Toc184915534)

[Приложение 1 main.c 30](#_Toc184915535)

[Приложение 2 hash.c 38](#_Toc184915536)

[Приложение 3 hash.h 39](#_Toc184915537)

[Приложение 4 login.c 40](#_Toc184915538)

[Приложение 5 login.h 41](#_Toc184915539)

[Приложение 6 sketch\_nov30a.ino 42](#_Toc184915540)

# Введение

Целью данного курсового проекта является написание системы умного дома, которая обеспечит автоматизацию повседневных обязанностей в повседневной жизни основанная на базе Arduino ESP8266 с Wi-Fi модулем. Система умного дома в современном мире появилась относительно недавно и с каждым днем набирает все больше популярности среди современного населения по всему земному шару. Возможности умного дома ограничиваются только идеями самого пользователя например: удалённое управление освещением, просмотр данных всех различных датчиков, открытие и закрытие окон и дверей, просмотр камер видео наблюдения и многое другое.

В первой части будут рассмотрены предметная область данной темы и несколько готовых решений.

Во второй части будут рассмотрены разработанные модули и инструменты, а также листинги некоторых частей программы.

В третьей части будет рассмотрено руководство оператора.

# Глава 1. Теоретическая часть

## 1.1 Описание предметной области

Умный дом или известный еще как домашняя автоматизация - кластер устройств способных выполнять и анализировать множество команд и задач без прямого участия человека. Наиболее распространение сценарии использования систем умного дома:

1. автоматическое включение и выключение освещения
2. управление отопительными и сплит системами
3. системы уведомления разных датчиков

Домашняя автоматизация это одна из тем интернет вещей.

### 1.1.1 Интернет вещей

Интернет вещей (Internet of Things, IoT) — это множество физических объектов, подключенных к интернету и обменивающихся данными. Концепция IoT может существенно улучшить многие сферы нашей жизни и помочь нам в создании более удобного, умного и безопасного мира. Примеры Интернета вещей варьируются от носимых вещей, таких как умные часы, до умного дома, который умеет, например, контролировать и автоматически менять степень освещения и отопления. Также ярким примером служит так называемая концепция умного предприятия (Smart Factory), которое контролирует промышленное оборудование и ищет проблемные места, а затем перестраивается так, чтобы не допустить поломок. Интернет вещей занимает важное место в процессе цифровой трансформации в компаниях.

### 1.1.2 Основные компоненты

Устройства и датчики — это физические приборы например: умные браслеты, системы видео и аудио наблюдения, которые могут собирать данные и отправлять их на сервер или устройства.

Сеть: Устройства подключаются к интернету через протоколы связи такие как Wi-Fi, Bluetooth, Zigbee, Meter-Bus что дает им обмениваться данными в реальном времени.

Обработка данных:

Собранные данные анализируются с помощью локальных серверов, что позволяет выполнять сценарии при различных входных данных.

### 1.1.3 Применение Интернет вещей

Интернет вещей находит применение в различных сферах такие как умные дома, здравоохранение, промышленность

### 1.1.4 Будущее Интернет вещей

По прогнозам, рынок Интернет вещей будет продолжать расти и развиваться. Это связано с увеличением числа подключённых устройств и развитием технологий, таких как 5G обеспечивающие более быструю и надёжную связь.

## 1.2 Описание существующих разработок

На рынке программного обеспечения (дальше именуемым ПО) управления системами умного дома существует множество вариантов, безусловными лидерами данной области на российском рынке являются:

1. Яндекс с системой управлением умным домом именуемый “Алиса”
2. Сбер с системой управлением умным домом именуемый “Салют”

Обе компании предлагают мобильное приложение, которое позволяет управлять умными устройствами не только тех, которые производит сама компания, но и сторонних производителей.

Так же к системе умного дома они предлагают устройство для голосового управления системой умного дома, а именно колонку с голосовым ассистентом.

## 1.3 Выбор инструментов

При выборе инструмента и языка программирования было проведено сравнение по критериям, которые представлены в таблицах 1 и 2.

Степень важности будет оцениваться по 5 системе, где 5 — это наибольшая важность, а 1 — это необязательный фактор.

Таблица 1. Критерии выбора инструмента.

|  |  |
| --- | --- |
| Критерий | Важность критерия |
| Функционал | 3 |
| Удобство использования | 2 |
| Скорость разработки | 3 |

Таблица 2. Оценка языков программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий  Язык  программирования | C++ | Python | C | bash |
| Функционал | 4 | 5 | 4 | 3 |
| Удобство использования | 3 | 2 | 5 | 2 |
| Скорость разработки | 3 | 5 | 4 | 3 |
| Итого | 10 | 12 | 13 | 8 |

По итогам проделанной работы я выбрал для себя язык программирования C.

Для собственного удобства я выбрал редактор кода VIM и Arduino IDE для обновления прошивки платы ESP8266. Выбор обоснован тем, что я уже имел дело работать с данными редакторами кода.

# Глава 2. Проектная часть

## 2.1 Диаграмма прецедентов

Для определения вариантов использования к проекту была построена диаграмма прецедентов

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок . Диаграмма прецедентов для проекта

## 2.2 Проектирование предварительного дизайна

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

## 2.3 Проектирование сценария

В данном разделе приведен пример использования программы пользователем.

Изображение выглядит как текст, диаграмма, чек, Параллельный

Автоматически созданное описание

Рисунок 2. Сценарий использования

После запуска программы пользователь должен войти в учетную запись. После пользователь для изучения функционала может написать команду help которая выведет перечень команд. Подробнее можно узнать в Главе 3.

## 2.4 Описание главного модуля

Функция find предназначена для поиска строки в массиве строк и возвращает индекс найденной строки или -1, если строка отсутствует. Она перебирает элементы массива до тех пор, пока не встретит NULL, который обозначает конец массива, и сравнивает каждую строку с искомой строкой word с помощью функции strcmp. Если строки совпадают, функция возвращает индекс текущего элемента, в противном случае — -1.

Листинг 1. Функция для поиска команды в массиве.

int find(char \*word, char \*mass[]) {

for (int i = 0; mass[i] != NULL; i++) {

if (strcmp(mass[i], word) == 0) {

return i;

}

}

return -1;

}

Изображение выглядит как диаграмма, текст, линия

Автоматически созданное описание

Рисунок 3. Функция для поиска команды в массиве

Функция WriteCallback используется в контексте работы с библиотекой, например, libcurl, для обработки данных, получаемых в ходе выполнения HTTP-запроса. Она принимает блоки данных (contents) и добавляет их к буферу, на который указывает userp, объединяя их с помощью функции strncat. Перед добавлением данных в буфер функция завершает его нулевым символом, чтобы избежать переполнения. Возвращаемое значение равно общему размеру обработанных данных (size \* nmemb), что сообщает библиотеке об успешной обработке.

Листинг 2. Функция для отправки HTTP-запроса и получения ответа

size\_t WriteCallback(void \*contents, size\_t size, size\_t nmemb, void \*userp) {

((char \*)userp)[size \* nmemb] = '\0';

strncat(userp, contents, size \* nmemb);

return size \* nmemb;

}

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 4. Функция для отправки HTTP-запроса и получения ответа

Функция control\_led предназначена для управления состоянием светодиодов, подключенных к устройству, например, Arduino. Она формирует URL-запрос, включающий номер светодиода (led\_num) и его состояние (state), на основе предопределенных массивов led\_pins и states. Сформированный URL передается в функцию send\_http\_request, которая отправляет запрос на устройство. В случае успешного выполнения запроса функция выводит сообщение о состоянии светодиода с использованием цветовой разметки, а в случае ошибки — сообщение об ошибке.

Листинг 4. Функция для управления светодиодами

void control\_led(int led\_num, int state) {

const char \*led\_pins[] = {"14", "12", "13", "15"};

const char \*states[] = {"0", "1"};

char url[100];

snprintf(url, sizeof(url), "%s/led?pin=%s&state=%s", ARDUINOIP, led\_pins[led\_num - 1], states[state]);

if (send\_http\_request(url)) {

printf("led %d %s\n", led\_num, state ? COLOR\_GREEN "on" COLOR\_RESET : COLOR\_RED "off" COLOR\_RESET);

} else {

printf("Failed to control LED %d\n", led\_num);

}

}

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 5. Функция для управления светодиодами

Функция get\_led\_status предназначена для получения текущего состояния светодиодов, подключенных к устройству, например, Arduino. Она отправляет HTTP-запрос на устройство по адресу, указывающему на конечную точку /status, и получает ответ, содержащий информацию о состоянии светодиодов. Если запрос выполнен успешно, функция выводит полученный статус на экран и освобождает память, выделенную для ответа. В случае ошибки выводится сообщение об ошибке.

Листинг 4. Функция для получения статуса светодиодов

void get\_led\_status() {

char \*response = send\_http\_request(ARDUINOIP "/status");

if (response) {

printf("Status: %s\n", response);

free(response);

} else {

printf("Failed to get LED status\n");

}

}

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 6. Функция для получения статуса светодиодов

Функция ping используется для проверки доступности устройства, например, Arduino, путем отправки HTTP-запроса на конечную точку /ping. Если устройство отвечает, функция выводит полученный ответ на экран и освобождает память, выделенную для хранения ответа. В случае, если запрос не удался, выводится сообщение об ошибке. Эта функция полезна для диагностики подключения к устройству.

Листинг 5. Функция для проверки соединения с сервером

void ping() {

char \*response = send\_http\_request(ARDUINOIP "/ping");

if (response) {

printf("Ping response: %s\n", response);

free(response);

} else {

printf("Failed to ping\n");

}

}

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 7. Функция для проверки соединения с сервером

## 2.5 Описание тестовых наборов модулей

В этом разделе будут продемонстрированы результаты тестирования “черного ящика”.

Тест 1. Проверка работоспособности управления светодиодом при наличии соединения.

Модуль: отправка запроса

Действия: Включение первого светодиода командой led\_1 1

Ожидаемый результат: led 1 on

Фактический результат:

Изображение выглядит как текст, Шрифт, рукописный текст, число

Автоматически созданное описание

Рисунок 8. Проверка работоспособности управления светодиодом при наличии соединения.

Тест 2. Проверка работоспособности управления светодиодом при отсутствии соединения.

Модуль: Отправка запроса

Действия: Включение первого светодиода командой led\_1 1

Ожидаемый результат: led 1 on

Фактический результат: Failed to control LED 1

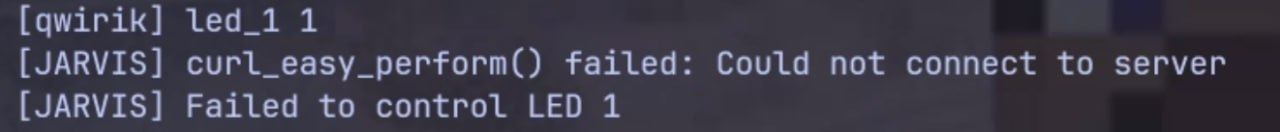


Рисунок 9. Проверка работоспособности управления светодиодом при отсутствии соединения.

Тест 3. Проверка работоспособности получения статуса светодиодов при наличии соединения.

Модуль: Получение ответа

Действия: Получение ответа после ввода команды status

Ожидаемый результат: Возвращение статуса светодиодов

Фактический результат: Status: {“led\_1”:1, “led\_2”:0, “led\_3”:0, “led\_4”:0}

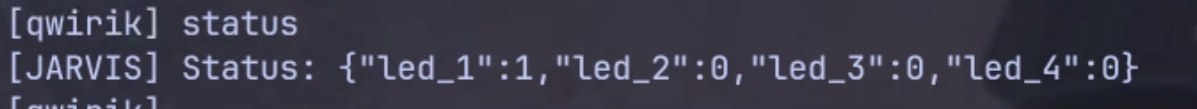


Рисунок 10. Проверка работоспособности получения статуса светодиодов при наличии соединения.

Тест 4. Проверка работоспособности получения статуса светодиодов при отсутствии соединения.

Модуль: Получение ответа

Действия: Получение ответа после ввода команды status

Ожидаемый результат: Failed to get LED status

Фактический результат: Failed to get LED status

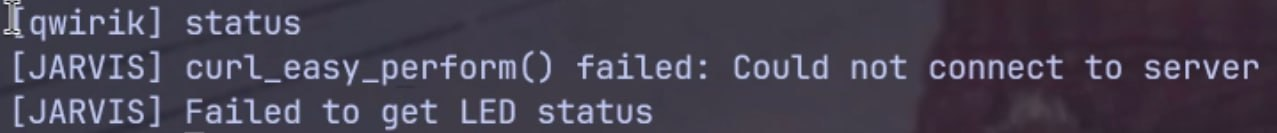


Рисунок 11. Проверка работоспособности получения статуса светодиодов при наличии соединения.

Тест 5. Проверка работоспособности получения статуса соединения при отсутствии соединения.

Модуль: Получение ответа

Действия: Получение ответа после ввода команды ping

Ожидаемый результат: Ping response: Pong

Фактический результат: Ping response: Pong

Изображение выглядит как текст, грифельная доска, рукописный текст, Шрифт

Автоматически созданное описание

Рисунок 12. Проверка работоспособности получения статуса светодиодов при наличии соединения.

Тест 6. Проверка работоспособности получения статуса соединения при отсутствии соединения.

Модуль: Получение ответа

Действия: Получение ответа после ввода команды ping

Ожидаемый результат: Failed to ping

Фактический результат: Failed to ping

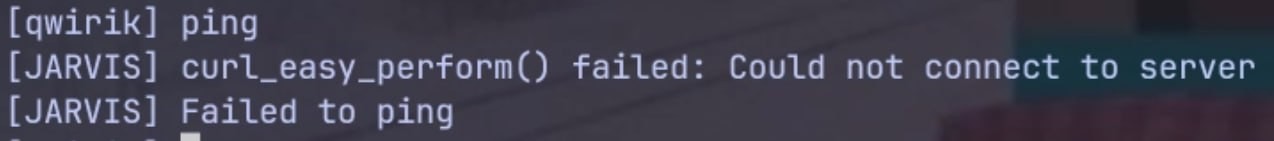


Рисунок 13. Проверка работоспособности получения статуса соединения при отсутствии соединения.

Тест 7. Проверка работоспособности очистки терминала.

Модуль: Команды

Действия: Очистка терминала командой clear.

Ожидаемый результат: “Очистка поля”

Фактический результат: “Очистка поля”

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 14. Проверка работоспособности очистки терминала (1).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 15. Проверка работоспособности очистки терминала (2).

Тест 8. Проверка выхода из программы.

Модуль: Команды

Действия: Выход из программы командой exit и согласием со следующим вопросом

Ожидаемый результат: exiting…

Фактический результат: exiting…

Изображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Автоматически созданное описание

Рисунок 16. Проверка выхода из программы.

Тест 9. Проверка входа в учетную запись пользователя при верных выходных данных.

Модуль: Входа

Действия: Вход в учетную запись пользователя при верных выходных данных.

Ожидаемый результат: access successful

Фактический результат: access successful

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 17. Проверка входа в учетную запись пользователя.

Тест 10. Проверка входа в учетную запись пользователя при не верном логине.

Модуль: Входа

Действия: Вход в учетную запись пользователя при неверном логине.

Ожидаемый результат: invalid login

Фактический результат: invalid login

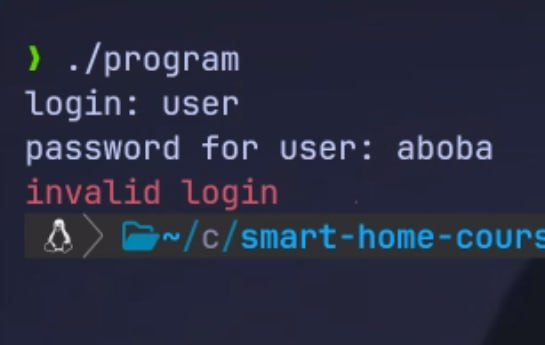


Рисунок 18. Проверка входа в учетную запись пользователя при не верном логине.

Тест 11. Проверка входа в учетную запись пользователя при не верном пароле.

Модуль: Входа

Действия: Вход в учетную запись пользователя при неверном пароле.

Ожидаемый результат: access denied

Фактический результат: access denied

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 19. Проверка входа в учетную запись пользователя при не верном пароле.

## 2.5 Отладка

При написании курсового проекта у меня возникла ошибка в неспособности выйти из программы с помощью команды exit.

Изображение выглядит как текст, Шрифт, рукописный текст, грифельная доска

Автоматически созданное описаниеРисунок 20. Логическая ошибка команды exit.

Методом анализом я выявил что ошибка в отсутствии запроса на ответ и в следствии программа не завершалась. Это исправилось когда я добавил строку запроса ответа.

# Глава 3. Эксплуатационная часть

## 3.1 Руководство оператора

### 3.1.1 Назначение программы

**Функциональное назначение программы**

Основной функцией проекта "smart-home" является автоматизация процесса управления устройствами умного дома.

**Эксплуатационное назначение программы**

Программное обеспечение "smart-home" может быть использовано в любом коммерческом и не коммерческом здании для управления устройствами умного дома.

**Состав функций**

Функции оператора:

1. Включение\выключение освещения
2. Просмотр статус устройства
3. Авторизация

### 3.1.2 Условия выполнения программы

Минимальный состав аппаратных средств

Минимальный состав используемых технических (аппаратных) средств:

* OC Windows 7/8.1/10/11 , GNU Linux
* Процессор с тактовой частотой не ниже 500 МГц
* ОЗУ 128КБ
* Видеоадаптер: Любой
* Место на жестком диске 7КБ
* Доступ в интернет

**Минимальный состав программных средств**

Программа компиляции программы.

**Требования к персоналу (оператору)**

Элементарные умения работы с терминалом.

**Загрузка и запуск программы**

Для запуска необходимо запустить скомпилированную программу под вашу ОС.

При запуске программы должно открыться следующее окно:

Далее требуется ввести логин и пароль если пароль логин и пароль введен не верен, то программа завершиться . После успешной авторизации пользователь может узнать какие функции поддерживает программа, введя команду “help”.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 21. Демонстрация входа в учетную запись пользователя

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 22. Демонстрация справочной информации

# Заключение

В результате выполнения курсового проекта была написана программа "smart-home" для автоматизации процесса управления умным домом.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки.

Так же планируется продолжить работу над данным проектом с целью расширения возможностей и удобства для пользователей. Планы по доработке представлены ниже.

1. Улучшение системы управления
2. Расширение функционала программы
3. Реализация голосового помощника .

# Список литературы и интернет-источников

Ли П. Архитектура интернета вещей / пер. с анг. М. А. Райтмана. – М.: ДМК Пресс, 2019. – 454 с.: ил

Умный дом на базе Arduino. Руководство пользователя

Умный дом своими руками. Тесля Е.В.

Кокунин П.А. Введение в Интернет вещей. Казань: Издательство Казанского университета, 2022. – 147 с. – Систем. требования: Adobe Acrobat Reader. – URL: https://kpfu.ru/portal/docs/F\_378200975/IOT.pdf. –

ISBN 978-5-00130-649-8

Глушак, Елена Владимировна. Введение в Интернет вещей: учебное пособие – Самара: Издательство Самарского университета, 2023. – 104 с.: ил.

ISBN 978-5-7883-2010-6

ИНТЕРНЕТ ВЕЩЕЙ, УМНЫЙ ДОМ И УМНЫЕ ГОРОДА. Кузяшев Азат Нургалеевич, Смолин Артем Евгеньевич

Папуловская, Наталья Владимировна. Основы интернета вещей : учебно-методическое пособие; М-во науки и высшего образования РФ.— Екатеринбург : Изд-во Урал. ун-та, 2022.— 104 с

То ли дом, то ли лаборатория: как устроен умный дом профессионала

https://habr.com/ru/companies/wirenboard/articles/821347/

Секреты умного дома от профессионала: что скрывает Андрей в своем особняке. https://habr.com/ru/companies/wirenboard/articles/850612/

Как работает умный дом и чем он может быть полезен

https://ichip.ru/tekhnologii/kak-rabotaet-umnyj-dom-854522

Основы построения системы "умный дом"

https://intuit.ru/studies/courses/644/500/info

Устелемова Основы построения системы "умный дом"

Что такое Умный Дом, общая структура Умного Дома и роль MajorDoMo при построении Умного Дома. https://rutube.ru/video/c0fafdeae14c3bb02f90232223209ad8/

Средства домашней автоматизации. Теория и практика «Умных домов». Часть первая. https://habr.com/ru/articles/381647/

Средства домашней автоматизации. Теория и практика «Умных домов». Часть вторая. https://habr.com/ru/articles/381649/

Система «Умный дом». Что это такое и почему он нужен каждому?

https://blog.eldorado.ru/publications/sistema-umnyy-dom-chto-eto-takoe-i-pochemu-on-nuzhen-kazhdomu-37927

Принципы построения современной системы «Умный дом»

https://www.elec.ru/publications/tsifrovye-tekhnologii-svjaz-izmerenija/1885/

Умный дом с нуля. https://stepik.org/course/125483/syllabus

Управление домом при помощи KNX: освещение

https://habr.com/ru/articles/427669/

С.В. Богданов - Умный Дом

Элсенпитер, Велт - Умный Дом строим сами

В.А. Петин - Создание умного дома на базе Arduino

А.П. Кашкаров - Электронные схемы для "умного дома"

Интернет вещей  
https://habr.com/ru/companies/otus/articles/549550/

# Приложение

## Приложение 1 main.c

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include <stdlib.h>

#include <curl/curl.h>

#include <readline/readline.h>

#include <readline/history.h>

#include <sys/time.h>

#include "hash.h"

#include "login.h"

#define ARDUINOIP "http://192.168.1.34" // Исправлено на правильный URL

// Функция для поиска команды в массиве

int find(char \*word, char \*mass[]) {

for (int i = 0; mass[i] != NULL; i++) {

if (strcmp(mass[i], word) == 0) {

return i;

}

}

return -1;

}

// Функция для отправки HTTP-запроса и получения ответа

size\_t WriteCallback(void \*contents, size\_t size, size\_t nmemb, void \*userp) {

((char \*)userp)[size \* nmemb] = '\0';

strncat(userp, contents, size \* nmemb);

return size \* nmemb;

}

char\* send\_http\_request(const char \*url) {

CURL \*curl;

CURLcode res;

char \*response = malloc(1024);

memset(response, 0, 1024);

curl\_global\_init(CURL\_GLOBAL\_DEFAULT);

curl = curl\_easy\_init();

if(curl) {

curl\_easy\_setopt(curl, CURLOPT\_URL, url);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, WriteCallback);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, response);

res = curl\_easy\_perform(curl);

if(res != CURLE\_OK) {

fprintf(stderr, "[JARVIS] curl\_easy\_perform() failed: %s\n", curl\_easy\_strerror(res));

curl\_easy\_cleanup(curl);

curl\_global\_cleanup();

free(response);

return NULL;

}

curl\_easy\_cleanup(curl);

}

curl\_global\_cleanup();

return response;

}

// Функция для управления светодиодами

void control\_led(int led\_num, int state) {

const char \*led\_pins[] = {"14", "12", "13", "15"};

const char \*states[] = {"0", "1"};

char url[1024];

snprintf(url, sizeof(url), "%s/led?pin=%s&state=%s", ARDUINOIP, led\_pins[led\_num - 1], states[state]);

if (send\_http\_request(url)) {

printf("led %d %s\n", led\_num, state ? COLOR\_GREEN "on" COLOR\_RESET : COLOR\_RED "off" COLOR\_RESET);

} else {

printf("Failed to control LED %d\n", led\_num);

}

}

// Функция для получения статуса светодиодов

void get\_led\_status() {

char \*response = send\_http\_request(ARDUINOIP "/status");

if (response) {

printf("Status: %s\n", response);

free(response);

} else {

printf("Failed to get LED status\n");

}

}

// Функция для измерения времени в миллисекундах

long millis() {

struct timeval tv;

gettimeofday(&tv, NULL);

return (tv.tv\_sec \* 1000 + tv.tv\_usec / 1000);

}

// Функция для проверки соединения с сервером и возврата задержки

void ping() {

char url[1024];

snprintf(url, sizeof(url), "%s/ping", ARDUINOIP);

CURL \*curl;

CURLcode res;

char \*response = malloc(1024);

memset(response, 0, 1024);

curl\_global\_init(CURL\_GLOBAL\_DEFAULT);

curl = curl\_easy\_init();

if (curl) {

// Засекаем время начала запроса

long start\_time = millis();

curl\_easy\_setopt(curl, CURLOPT\_URL, url);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, WriteCallback);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, response);

res = curl\_easy\_perform(curl);

// Засекаем время окончания запроса

long end\_time = millis();

if (res != CURLE\_OK) {

fprintf(stderr, "[JARVIS] curl\_easy\_perform() failed: %s\n", curl\_easy\_strerror(res));

} else {

// Вычисляем задержку в миллисекундах

long latency = end\_time - start\_time;

printf("%ld ms\n", latency);

}

curl\_easy\_cleanup(curl);

}

curl\_global\_cleanup();

free(response);

}

int main() {

int led\_status[] = {0, 0, 0, 0};

char login[MAX\_LEN\_STR];

char password[MAX\_LEN\_STR];

static char \*commands[] = {"exit", "help", "status", "led\_1 0", "led\_1 1", "led\_2 0", "led\_2 1", "led\_3 0", "led\_3 1", "led\_4 0", "led\_4 1", "clear", "ping", NULL};

static char \*yes[] = {"yes", "y", "yea", "yup", "yep"};

bool a = true;

// Запрос логина

printf("login: ");

scanf("%s", login);

// Запрос пароля

printf("password for %s: ", login);

scanf("%s", password);

// Попытка входа

bool access = log\_in(login, password);

getchar();

// Цикл команд

while (access) {

if (a){

printf("For a short summary of all commands, run 'help'\n");

a = false;

}

char prompt[MAX\_LEN\_STR];

snprintf(prompt, sizeof(prompt), "[%s] ", login);

char \*input = readline(prompt);

if (input && \*input) {

add\_history(input);

}

// Поиск команды

int command\_index = find(input, commands);

// Обработка команд

printf("[JARVIS] ");

switch (command\_index) {

case 0: // "exit"

char temp[MAX\_LEN\_STR];

printf("you are sure? (yes or no):");

scanf("%s",temp);

if (find(temp, yes) != -1) {

printf("exiting... \n");

access = false;

} else {

getchar();

}

break;

case 1: // "help"

printf("\n\

exit for exit from terminal\n\

status for print status pin\n\

led\_<num of pin> <status\_pin> for control pin\n\

clear for clear terminal\n\

ping for check connection\n");

break;

case 2: // "status"

get\_led\_status();

break;

case 3: case 4: case 5: case 6: case 7: case 8: case 9: case 10 : //control led

{

char \*token = strtok(input, " ");

int led\_num = token[4] - '0';

token = strtok(NULL, " ");

int state = atoi(token);

led\_status[led\_num - 1] = state;

control\_led(led\_num, state);

}

break;

case 11: // "clear"

system("clear");

printf("\n");

break;

case 12: //"ping"

ping();

break;

default: // Неизвестная команда

printf("unknown command\n");

}

free(input);

}

return 0;

}

## Приложение 2 hash.c

#include "hash.h"

int hash(const char \*str) {

unsigned long hash = 5381; // Стандартное начальное значение для DJB2

int c;

while ((c = \*str++)) {

hash = ((hash << 5) + hash) + c; // hash \* 33 + c

}

return hash;

}

## Приложение 3 hash.h

#ifndef HASH\_H

#define HASH\_H

// Функция для вычисления хеша строки

int hash(const char \*str);

#endif // HASH\_H

## Приложение 4 login.c

#include <stdio.h>

#include <string.h>

#include "hash.h"

#include "login.h"

int log\_in(char \*login\_i, char \*password\_i) {

unsigned long login\_h = hash(login\_i);

unsigned long password\_h = hash(password\_i);

unsigned long long password = 252832602ULL; // Используем ULL для больших чисел

unsigned long long login = 404414076; // Используем ULL для больших чисел

if (login != login\_h) {

printf(COLOR\_RED "invalid login" COLOR\_RESET "\n");

return false;

}

if (password != password\_h) {

printf(COLOR\_RED "access denied" COLOR\_RESET "\n");

return false;

} else {

printf(COLOR\_GREEN "access successful" COLOR\_RESET "\n");

return true;

}

}

## Приложение 5 login.h

#ifndef LOGIN\_H

#define LOGIN\_H

#include <stdbool.h>

// Цвета для вывода в терминал

#define COLOR\_RED "\x1b[31m"

#define COLOR\_GREEN "\x1b[32m"

#define COLOR\_YELLOW "\x1b[33m"

#define COLOR\_BLUE "\x1b[34m"

#define COLOR\_MAGENTA "\x1b[35m"

#define COLOR\_CYAN "\x1b[36m"

#define COLOR\_RESET "\x1b[0m"

// Константы

#define MAX\_LEN\_STR 255

// Функция для проверки логина и пароля

int log\_in(char \*login\_i, char \*password\_i);

#endif // LOGIN\_H

## Приложение 6 sketch\_nov30a.ino

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

const char\* \_ssid = "Paranoia";

const char\* \_password = "Paranoia\_Sarmat";

ESP8266WebServer server(80);

#define COLOR\_RED "\x1b[31m"

#define COLOR\_GREEN "\x1b[32m"

#define COLOR\_YELLOW "\x1b[33m"

#define COLOR\_BLUE "\x1b[34m"

#define COLOR\_MAGENTA "\x1b[35m"

#define COLOR\_CYAN "\x1b[36m"

#define COLOR\_RESET "\x1b[0m"

// Определение пинов кнопок и светодиодов

const int btn\_pin1 = 3;

const int btn\_pin2 = 5;

const int btn\_pin3 = 4;

const int btn\_pin4 = 2;

const int led\_pin1 = 14;

const int led\_pin2 = 12;

const int led\_pin3 = 13;

const int led\_pin4 = 15;

bool btn\_state1 = false;

bool btn\_state2 = false;

bool btn\_state3 = false;

bool btn\_state4 = false;

void setup() {

Serial.begin(9600); // Инициализация последовательного порта

// Настройка статического IP-адреса

IPAddress local\_IP(192, 168, 1, 34);

IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 255, 0);

if (!WiFi.config(local\_IP, gateway, subnet)) {

Serial.println("STA Failed to configure");

}

WiFi.begin(\_ssid, \_password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

pinMode(btn\_pin1, INPUT\_PULLUP);

pinMode(btn\_pin2, INPUT\_PULLUP);

pinMode(btn\_pin3, INPUT\_PULLUP);

pinMode(btn\_pin4, INPUT\_PULLUP);

pinMode(led\_pin1, OUTPUT);

pinMode(led\_pin2, OUTPUT);

pinMode(led\_pin3, OUTPUT);

pinMode(led\_pin4, OUTPUT);

server.on("/led", HTTP\_GET, handleLed);

server.on("/status", HTTP\_GET, handleStatus);

server.on("/ping", HTTP\_GET, handlePing); // Регистрация обработчика для /ping

server.begin();

}

void loop() {

server.handleClient();

// Проверка состояния каждой кнопки

checkButton(btn\_pin1, &btn\_state1, led\_pin1);

checkButton(btn\_pin2, &btn\_state2, led\_pin2);

checkButton(btn\_pin3, &btn\_state3, led\_pin3);

checkButton(btn\_pin4, &btn\_state4, led\_pin4);

}

void checkButton(int btn\_pin, bool \*btn\_state, int led\_pin) {

int currentBtnState = digitalRead(btn\_pin);

if (currentBtnState == LOW && \*btn\_state == true) {

delay(50);

\*btn\_state = false;

digitalWrite(led\_pin, !digitalRead(led\_pin));

} else if (currentBtnState == HIGH && \*btn\_state == false) {

\*btn\_state = true;

}

}

void handleLed() {

if (server.hasArg("pin") && server.hasArg("state")) {

int pin = server.arg("pin").toInt();

int state = server.arg("state").toInt();

// Проверка, что пин является одним из допустимых

if (pin == led\_pin1 || pin == led\_pin2 || pin == led\_pin3 || pin == led\_pin4) {

digitalWrite(pin, state);

server.send(200, "text/plain", "OK");

} else {

server.send(400, "text/plain", "Invalid pin");

}

} else {

server.send(400, "text/plain", "Missing arguments");

}

}

void handleStatus() {

String response = "\n";

response += "led\_1:" + String(digitalRead(led\_pin1) ? "\x1b[32m on \x1b[0m" : "\x1b[31m off \x1b[0m") + "\n";

response += "led\_2:" + String(digitalRead(led\_pin2) ? "\x1b[32m on \x1b[0m" : "\x1b[31m off \x1b[0m") + "\n";

response += "led\_3:" + String(digitalRead(led\_pin3) ? "\x1b[32m on \x1b[0m" : "\x1b[31m off \x1b[0m") + "\n";

response += "led\_4:" + String(digitalRead(led\_pin4) ? "\x1b[32m on \x1b[0m" : "\x1b[31m off \x1b[0m");

response += "";

server.send(200, "application/json", response);

}

void handlePing() {

String ipAddress = WiFi.localIP().toString();

server.send(200, "text/plain", ipAddress);

}