МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Разработка веб-интерфейса системы управления умным домом

Обучающегося 4 курса
очной формы обучения
Салаватова Михаила Валерьевича

Руководитель выпускной квалификационной работы:
кандидат педагогических наук, доцент
кафедры информационных технологий и электронного обучения
Государев Илья Борисович

Рецензент:
Ученая степень *(при наличии)*, ученое звание *(при наличии)*, должность
Ф. И. О. *(указывается в именительном падеже)*

Санкт-Петербург
2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность выбранной темы:

Огромная часть жизни современного человека уделена повседневной деятельности. Согласно исследованиям сотрудников Института социальной политики НИУ ВШЭ [1], люди уделяют домашней работе порядка 3 часов в сутки. 1095 часов в год на выполнение рутинных обязанностей по дому. Или, если продолжить расчеты, 76650 часов в среднем за жизнь. Закономерным вопросом будет: «Как сократить время, затрачиваемое на домашние дела, без потери качества самой жизни?». Переложить часть обязанностей с себя на автоматизированную систему – умный дом, или сделать контроль за многочисленными приборами и управление ими более доступными и энергоемкими для человека. Умный дом – это единая система управления устройствами: вентиляцией, водоснабжением, безопасностью, - постоянно окружающими нас в повседневной жизни. Умный дом является частью более широкой области под названием – Internet Of Things (IoT) или Интернет вещей. Интернет вещей — это концепция сети передачи данных между физическими объектами («вещами»), оснащенными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой [2].

Перри Ли, автор книги «Архитектура интернета вещей», так описывает будущее: «Датчик сна или умная подушка зафиксировали то, как вы спите. Данные были отправлены шлюзу IoT, а затем переданы вашему бесплатному облачному сервису, который отправляет отчеты на информационную панель на вашем телефоне. Вы обходитесь без будильника, но, если у вас, например, самолет в 5 утра, вы все же поставите будильник, который, опять же, контролируется облачным сервисом, работающим на протоколе IFTTT (англ. «if this then that» - «если это, тогда то»). Ваш двухзонный котел отопления обслуживается другим облачным сервисом и подключен к вашей домашней 802.11 Wi-Fi-сети, как и датчики дыма, дверной звонок, система полива, дверь гаража, камеры наблюдения и система безопасности» [3, с.21]

Однако, для среднестатистического пользователя в нашей стране, порог вхождения в большинство систем управления умным домом - выше среднего. Это послужило основанием для выбора темы дипломного проекта.

Цель дипломного проекта: спроектировать веб-интерфейс для системы управления умным домом, который будет прост в управлении.

Задачи дипломного проекта: в соответствии с поставленной целью в процессе дипломного проектирования были поставлены следующие задачи:

1. Спроектировать и создать модуль умного дома – «умная розетка» для тестирования системы.
2. Разработать базовый дизайн веб-интерфейса
3. Спроектировать веб-интерфейс для объединения всех модулей умного дома и их управления.

ПОСТАНОВКА ЗАДАЧИ

Описание предметной области

Интернет вещей является концепцией, сформулированной в 1999 году в Массачусетском технологическим институте. Концепцией взаимодействия физических объектов посредством цифрового обмена данными. В отличии от стандартной модели, где устройство привязано к пользователю и способно обмениваться информацией только с ним, в идею интернета вещей заложена возможность объектам (вещам) обмениваться данными между собой, образуя единую сеть. С этой технологией связаны как большие надежды, так и серьезные риски безопасности, связанные с всепроникающей тенденцией превращения обычных вещей в интернет-узлы для передачи данных. Таких как: автомобиль, бытовая и электронная техника, мебель и даже товарные упаковки.

Для удобства и более четкой классификации, основатель Европейского совета по «Интернету вещей», ведущий эксперт в области цифровизации и автор концепции Интернета вещей Роб Ван Краненбург ввел четыре уровня Интернета вещей по охвату области [4]:

1. **BAN (Body Area Network)** — все, что так или иначе находится на человеке: умные часы, футболки, кроссовки, очки и так далее.
2. **LAN (Local Area Network)** — по сути это умный дом: различные устройства в доме, объединенные в одну сеть.
3. **WAN (Wide Area Network)** — умные города: общественный транспорт, объединенный в одну сеть и имеющий выход в интернет; электростанции, объединенные в одну сеть и автоматически перераспределяющие нагрузку и так далее.
4. **VWAN (Very Wide Area Network)** — умная планета, где каждое устройство может взаимодействовать с любым другим.

В данном распределение просматривается закономерность объединения мелких сетей в более крупные, до тех пор, пока они не сольются в единую глобальную сеть обмена данными.

Различные независимые сети на одном уровне объединяются с использованием защищенных протоколов шифрования, аналитики и контролем за передаваемыми данными. Например, на уровне WAN представляющем собой умные города, возникнет необходимость в связующем звене между обособленными системами: транспортными, энергетическими, бизнес, образовательными и другими. Данная идея хорошо передана на рисунке 1.

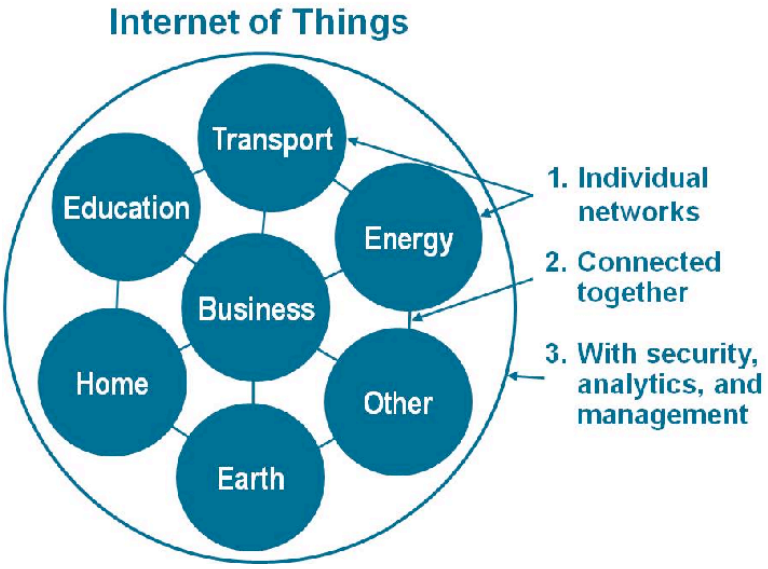

Рисунок 1 – Независимые системы IoT объединенные в единую сеть

Умный дом являет собой совокупность автоматизированных сценариев, а также «умных» вещей, связанных в единую сеть. В представленной иерархии, он занимает второй уровень – LAN (Local Area Network) или зону локальной сети. Это посредник между «умными» объектами в доме и человеком, получающим всю необходимую информацию в удобном виде: графика, кнопки переключения и так далее, - а также в удобном формате вывода: на персональном компьютере, телефоне, телевизоре или любом другом устройстве с дисплеем для отображения данных.

Для упрощения настройки подобных соединений, существуют платформы с интуитивно-понятным интерфейсом и быстрой установкой, что обеспечивает низкий порог вхождения.

Функции системы

Существует ряд требований, обеспечивающих будущую работоспособность создаваемой системы:

Масштабируемость. Система должна быть гибкой при добавлении новых устройств в сеть.

Зона покрытия. Должна охватывать весь участок дома.

Надежность соединения. Система управления должна обеспечивать бесперебойную работу для исправного функционирования всех устройств. И должна быть независимой от внешних устройств, подключенных к ней.

Простота управления. Порог вхождения для новых пользователей должен быть приемлемым.

Технологии и элементная база

Система умного дома представляет из себя связку интеллектуальных устройств, обменивающихся данными. Следовательно, для анализа системы управления, необходимо создать простейшую сеть объектов (вещей) способных передавать информацию в эту систему. Для создания подобных устройств нужна элементная база:

Микроконтроллеры

Микроконтроллер – это микросхема, предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. Строго говоря, это однокристальный компьютер, способный выполнять относительно простые задачи, такие как: открывания дверей, включения полива газонов, получение данных с датчиков, форматирование полученных данных для последующей отправки и многое другое.
Именно благодаря тому, что микроконтроллер способен выполнять столь разнообразный функционал, потребность в них ежегодно повышается, а

производство во много раз превышает производство процессоров. Вторым фактором их распространенности является нужда производителей в относительно недорогом и сбалансированном модуле для встроенных систем, игрушек, домашней технике, где встраивание процессора существенно повысит итоговую стоимость продукта.

В ходе работы микроконтроллер считывает команды из памяти или порта ввода и исполняет их. Что означает каждая команда, определяется системой команд микроконтроллера. Система команд заложена в архитектуре микроконтроллера и выполнение кода команды выражается в проведении внутренними элементами микросхемы определенных микроопераций. Микроконтроллеры дают возможность гибко управлять различными электронными устройствами. Некоторые модели микроконтроллеров настолько мощны, что могут непосредственно переключать реле (для включения и выключения лампочки, например).

В данной дипломной работе будет использоваться микроконтроллер ESP8266 установленный на плату NodeMcu v3 Lua Wi-Fi. Данная плата с микроконтроллером была выбрана ввиду невысокой стоимости, а также возможности подключиться к беспроводной сети и передавать данные на устройства через Wi-Fi или превратить саму плату в точку доступа, для подключения модулей к ней напрямую.

Датчики

Для проверки работы, системы управления умным домом будут получать информацию с датчиков и, обрабатывая ее, отображать в понятном человеку виде через пользовательский интерфейс.

Датчик температуры и влажности будет замерять показатели в комнате и передавать их на плату NodeMcu, а та в свою очередь, подключаясь к системе управления, отобразит полученные данные на локальном сервере. Был выбран датчик DHT11 представленный на рисунке 2.

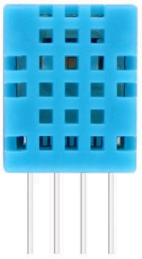

Рисунок 2 – Датчик температуры и влажности DHT11

Датчик движения. Принцип действия основан на регистрации изменения инфракрасного (ИК) излучения, вызванного перемещением или деятельностью человека. По физической природе видимый свет и ИК излучение одинаковы. ИК излучение также можно сфокусировать линзой, как обычный свет. При попадании ИК излучения на фотоэлемент, расположенный внутри датчика, он меняет свои параметры. При комнатной температуре в видимом свете тела не светятся, а в ИК диапазоне – сияют, что отчетливо видно на рисунке 3.

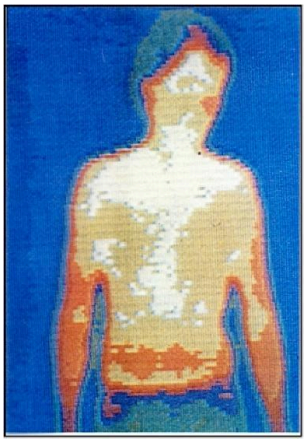

Рисунок 3 - Распределение температуры человеческого тела в инфракрасном спектре

Для данного проекта был выбран модуль инфракрасного датчика движения **HS-SR501** представленный на рисунке 4.

Рисунок 4 - Датчика движения HS-SR501

Релейный модуль

Реле — электромеханическое устройство, предназначенное для коммутации электрических цепей, цепей сигнализации и управления. Электромагнитное реле представляет из себя катушку. Она состоит из основания из немагнитного материала, на которое намотан медный провод, как правило покрытый диэлектрическим лаком.

При подаче напряжения на катушку происходит втягивание металлического сердечника, связанного с толкателем, который приводит в движение контакты.

Данное реле будет использоваться в «умной» розетке. NodeMcu будет принимать команды от системы управления и передавать на реле через pin IN, та в свою очередь будет включать или выключать питание розетки. Реле, используемое в этом проекте показано на рисунке 5.

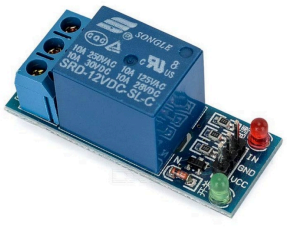

Рисунок 5 – Одноканальный модуль реле 5V

AC/DC преобразователь

Преобразователь представляют из себя устройство, предназначенное для преобразования входного напряжения. Он может повышать или понижать его, преобразовывать постоянный электроток в переменный и наоборот. Соответственно, принцип функционирования оборудования зависит от его типа. Для создания «умной» розетки понадобится понижающий преобразователь с входным напряжением 220V и выходным 5V для питания управляющей платы NodeMcu от розетки. На рисунке 6 представлен такой преобразователь.

Рисунок 6 - IRM-03-5 AC/DC преобразователь на 5V

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МОДУЛЯ УМНОГО ДОМА

В результате выполнения данного модуля, будет спроектирована розетка, управляемая пользователем удаленно. Схема представлена на рисунке 7. Блоком управления является плата NodeMcu. Питание подается от обычной розетки через вилку с заземлением. Напряжение от розетки преобразуется на понижающем ac/dc преобразователе со стандартных 220V до 5V, подходящих для питания платы. Управляющий модуль контролирует работу 1-канального реле. Реле контролирует подачу тока в розетку, к которой будет подключаться нагрузка в виде настольной лампы.

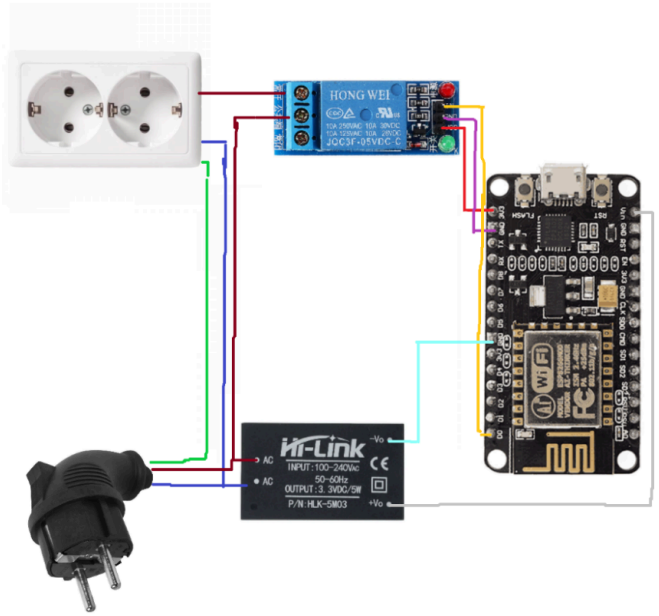

Рисунок 7 – Схема модуля умного дома - розетка

Платформа и язык программирования

Для написания кода была выбрана интегрированная среда разработки Arduino IDE, предназначенная для создания и загрузки программ на Arduino-совместимые платы, а также на платы других производителей. Интерфейс показан на рисунке 8. В данной IDE, в качестве языка программирования, используется стандартный C++ с упрощающими написание кода особенностями. Интерфейс программы представлен на рисунке 8.

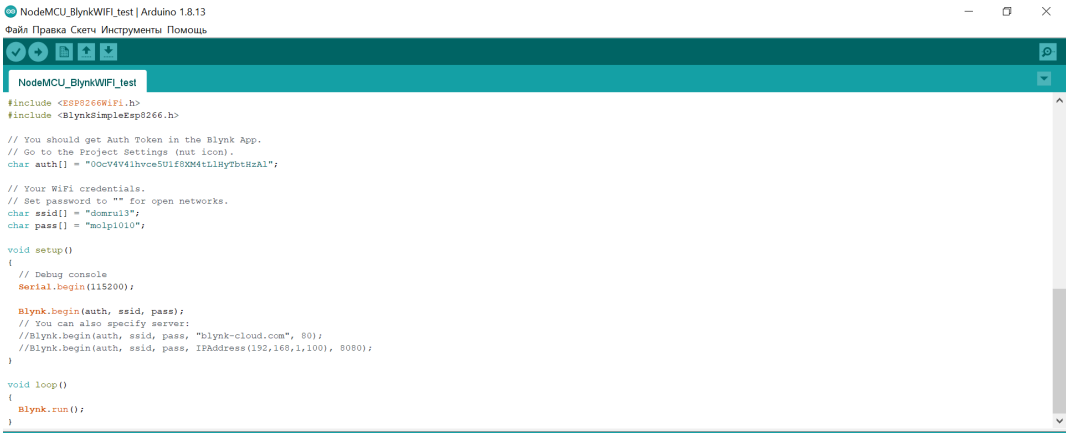

Рисунок 8 – Интерфейс Arduino IDE

Результат работы представлен на рисунках 9 и 10.

Рисунок 9 – Розетка с дистанционным управлением

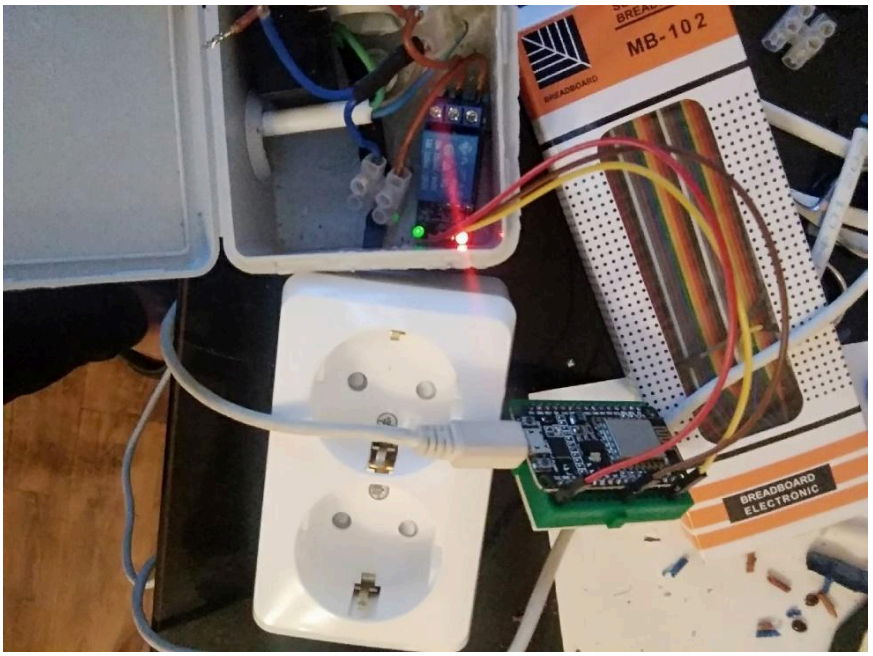

Рисунок 10 – Элементная составляющая розетки

Подключение датчиков

Для фиксирования данных, получаемых с датчиков, будет использована вторая плата NodeMcu аналогичная первой. Питание подается через micro-usb порт от обычной розетки с помощью переходника. К плате подключается датчик движения и датчик температуры и влажности. Схема подключения DHT11 показана не рисунке 11. Для тестирования работы датчиков был написан код, представленный в приложении Б.1. В результате выполнения программы, данные передаются на локальный сервер, обновляясь с помощью ajax запросов. Ajax – это подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

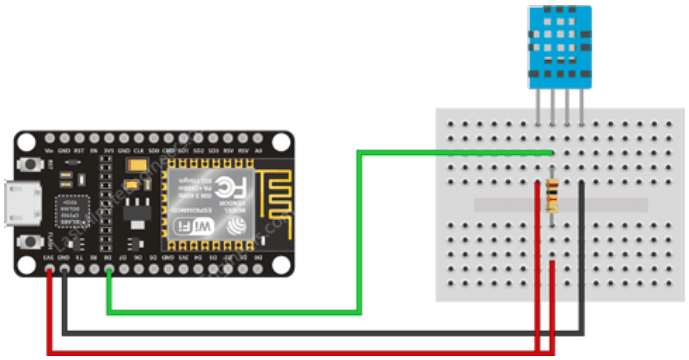

Рисунок 11 – Схема подключения DHT11 к плате NodeMcu

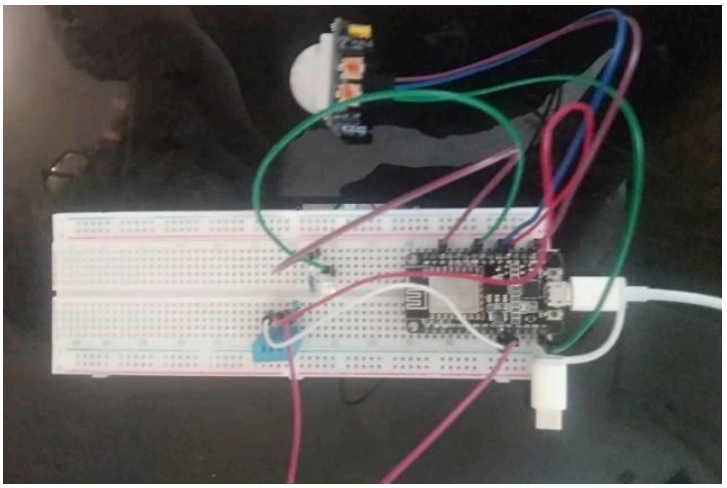

Рисунок 12 – Подключенные к плате датчики температуры и движения

Датчики, подключенные к плате показаны на рисунке 12.

Плата NodeMcu использует домашний wi-fi для создания локального сервера. Таким образом, все девайсы, находящиеся в одной сети с NodeMcu, получают доступ к сайту, где в виде наглядного интерфейса отображаются данные: температура и влажность комнаты, движения в квартире. Смотреть статистику можно как с телефона – рисунок 13, так и с компьютера – рисунок 14.

Рисунок 13 – Доступ к локальному серверу через телефон

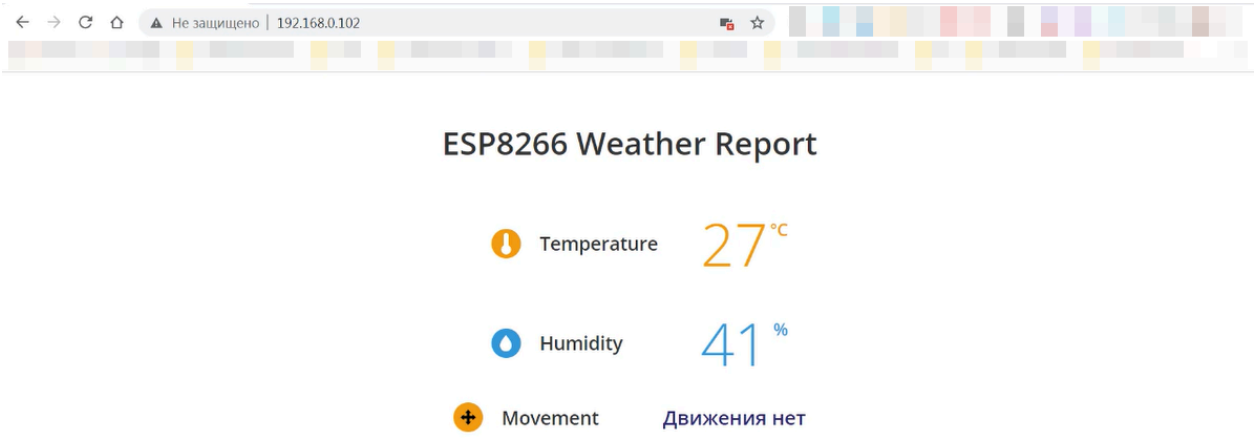

Рисунок 14 – Доступ к локальному серверу через пк

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ИНТЕРФЕЙСА СИСТЕМЫ

При проектировании поведения интерфейса был использован принцип цифровой доски или whiteboard: подключенные датчики визуально отображаются в виде прямоугольных блоков с информацией - фреймов. Фреймы могут быть помещены в любое место в пределах доски. Также они могут растягиваться на любую длину и ширину, чтобы вместить весь содержащийся внутри них контент. Один фрейм передает информацию только с одного подключенного модуля. Удаление отдельных фреймов не влияет ни на работу системы в целом, ни на отдельные ее части. Каждый фрейм содержит необходимый минимум информацию о подключенном модуле для корректной работы.

Рассмотрим систему более подробно.

Внешний вид интерфейса

Один из примеров визуального оформления интерфейса представлен на рисунке 15. Пользователь может изменить общий вид с помощью внутренних настроек, выбрав один из трех имеющихся стилей: классический, темный, светлый.

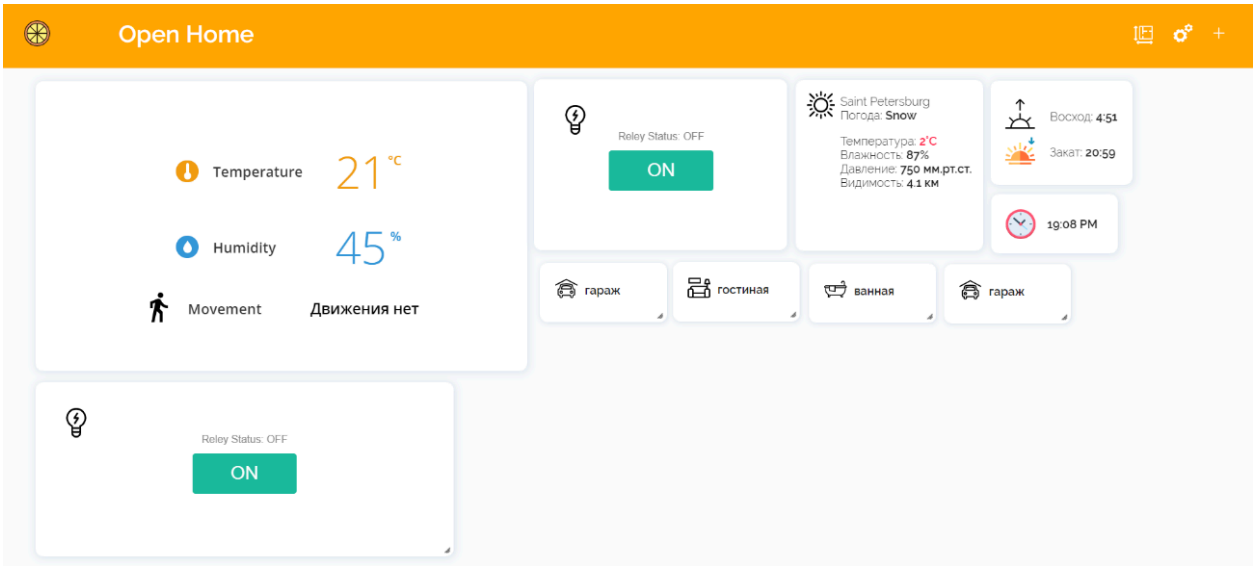

Рисунок 15 – Веб-интерфейс системы управления (классический стиль)

Другие варианты оформления показаны на рисунках 16 и 17.

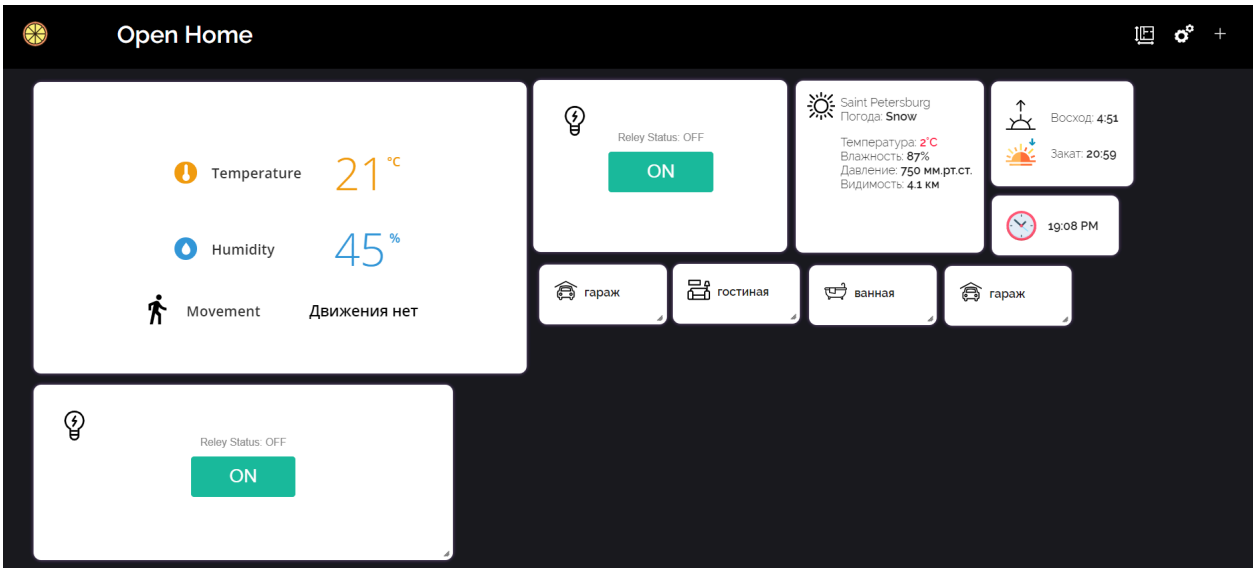

Рисунок 16 – веб-интерфейс (темный стиль)

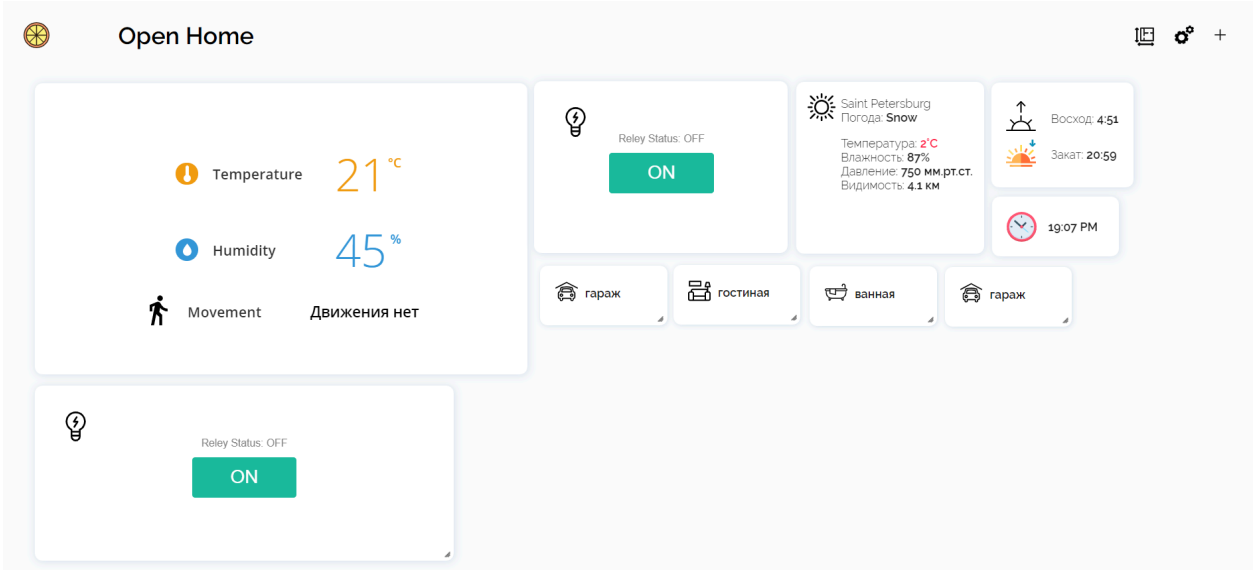

Рисунок 17 – веб-интерфейс (светлый стиль)

Переключение осуществляется при помощи radio-button во всплывающей вкладке настроек (рисунок 18). При подтверждении выбора, информация об id выбранного варианта отправляется в php файл, который отправляет sql-запрос в базу данных для обновления имеющейся в ней записи style в таблице pages. При загрузке html-страницы, в эту же таблицу отправляется запрос SELECT для получения данных о текущем стиле

веб-страницы (рисунок 19). В соответствии с полученным значением, загружается тот или иной css файл.

При изменении стиля на одной странице, он сохранится и будет отображаться на всех.

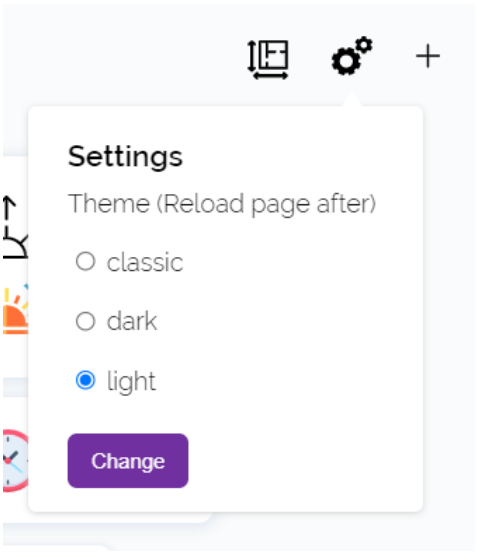

Рисунок 18 – Настройки стиля веб-страницы

Рисунок 19 – Запись в таблице pages для сохранения значения текущего стиля страниц

Фреймы

Фрейм — отдельный законченный HTML-документ, который вместе с другими HTML-документами может быть отображён в окне браузера. Фреймы по своей сути очень похожи на ячейки таблицы, однако более универсальны.

В данной работе, фреймы используются в качестве связующих звеньев между платами nodemcu, с подключенными к ним устройствами, и веб-интерфейсом, располагающимся на локальном веб-сервере управляющего блока (компьютера).

Принцип работы:

На модуле (плате nodemcu), который планируются подключить к системе, поднимается локальный сервер с прописанной в скетче html-страницей. Данная страница задает функционал конкретного модуля: переключение состояний реле при нажатии на html кнопку, отображение значений датчиков и т.п. Вся стилизация происходит на стороне модуля.

Плата переводится в WI-FI режим станции (STA) и подключается к существующей wi-fi сети, к которой уже подключен управляющий блок. Важно, чтобы все модули находились в единой, с блоком управления, сети.

В скетче модуля, при подключении к сети, прописывается следующая строка: Serial.println(WiFi.localIP()). При успешном подключении, она выведет в монитор порта IP адрес платы, который понадобиться для добавления нового фрейма.

Добавив папку с проектом в каталог доменов локального сервера, по адресу http://openhome.esp/ будет доступен веб-интерфейс.

Нажав на кнопку «+» в правом верхнему углу экрана, откроется окно для создания нового фрейма (рисунок 20). Пользователю необходимо будет придумать frameID устройства и указать ip-адрес платы, который он получил на шаге 3.

При совершении всех перечисленных действий, в веб-интерфейс автоматически добавится новый фрейм, ссылающийся на страницу подключенной платы. Если модуль подключен к питанию, страница должна быть доступна в локальной сети wi-fi и фрейм отобразит содержимое, будь то кнопка управления реле (рисунок 21) или визуально оформленные данные, получаемые с датчиков (рисунок 22).

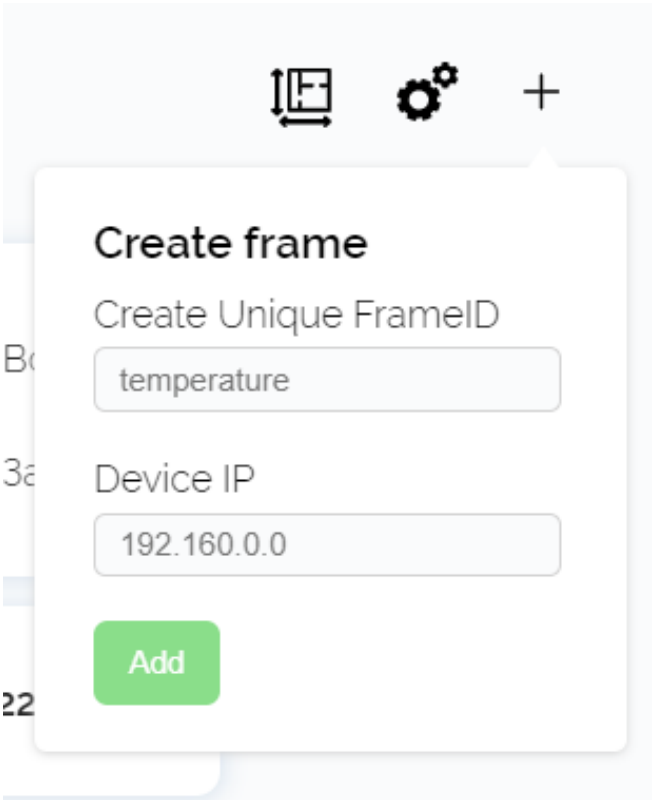

Рисунок 20 – Форма создания нового фрейма

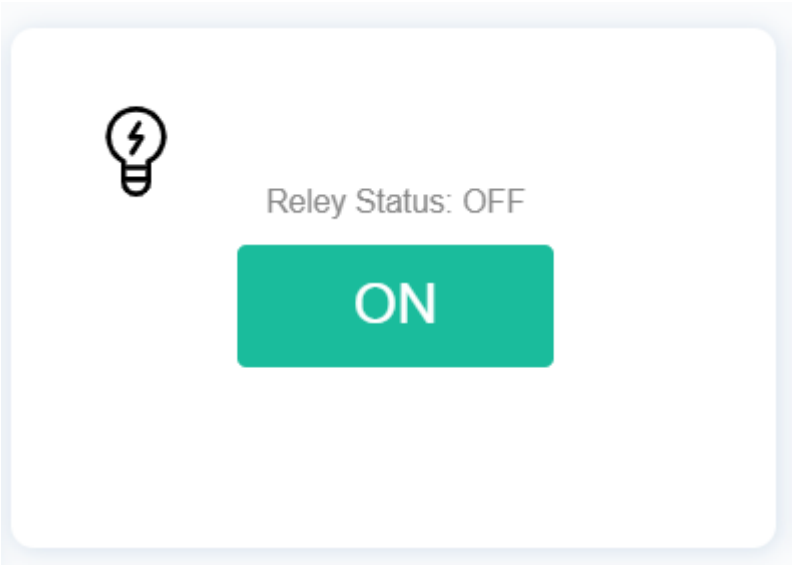

Рисунок 21 – Фрейм управления реле

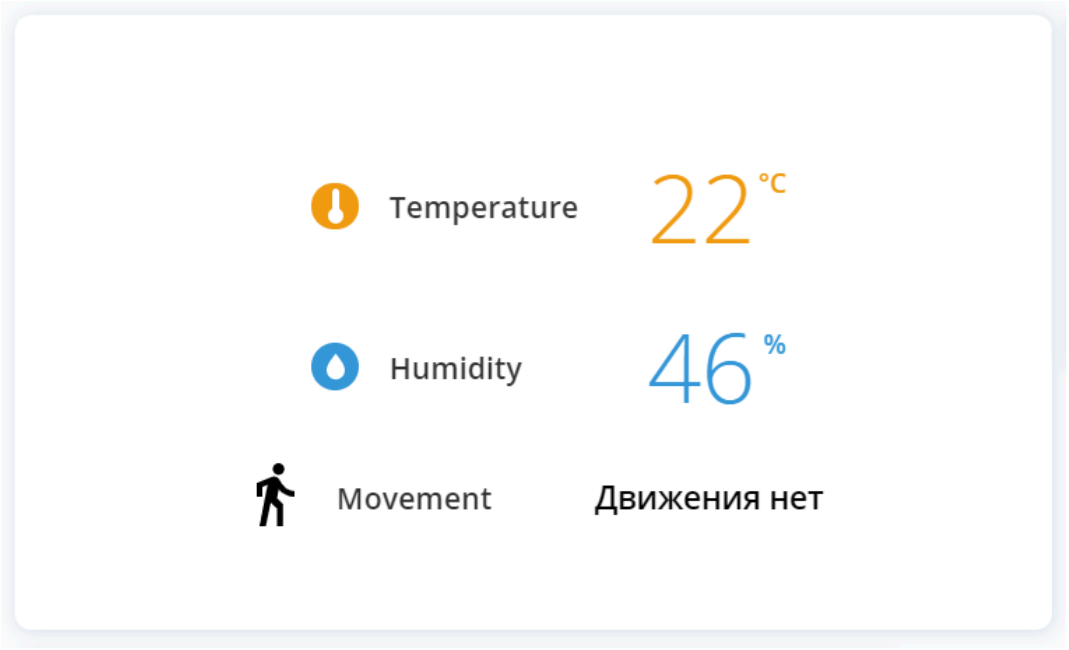

Рисунок 22 – Фрейм с данными датчиков

Если плата не подключена к питанию, ip-адрес будет ссылать фрейм на несуществующую в данный момент страницу, что не нарушит работы остальных фреймов и веб-интерфейса в целом.

Фрейм, ссылающийся на несуществующую страницу, будет поддаваться масштабированию и перемещению как все остальные блоки. При подключении модуля к сети и перезагрузке страницы, фрейм автоматически найдет и отобразит данные с платы. Пример выключенного фрейма представлен на рисунках 23.

Рисунок 23 – Фрейм ссылающийся на несуществующий адрес

Фреймы, добавляемые пользователем, можно перемещать и масштабировать (изменять ширину и высоту) в пределах родительского блока content. При последующей загрузке страницы, фреймы останутся в том же положении и в тех же пропорциях.

Перемещения и изменения размеров фреймов сохраняются в локальном хранилище (LocalStorage) устройства и не требуют выгрузки значений в базу данных, что существенно замедлило бы работу системы или вовсе сделало бы ее невозможной.

На каждом устройстве, имеющем доступ к локальному серверу, веб-интерфейс будет иметь свои уникальные положения и размеры фреймов, заданные пользователем.

Пример изменений размера фрейма относительно других элементов на странице: рисунок 24 и 25.

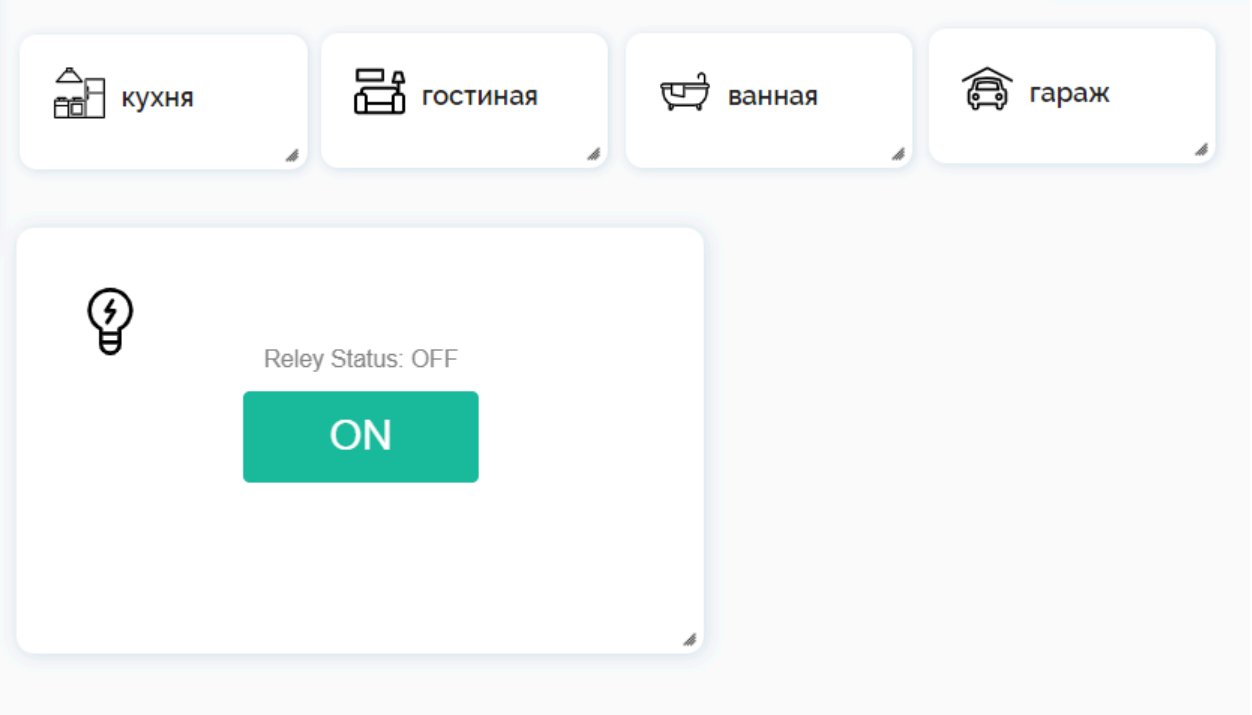

Рисунок 24 – Начальное положение и размер фрейма «реле»

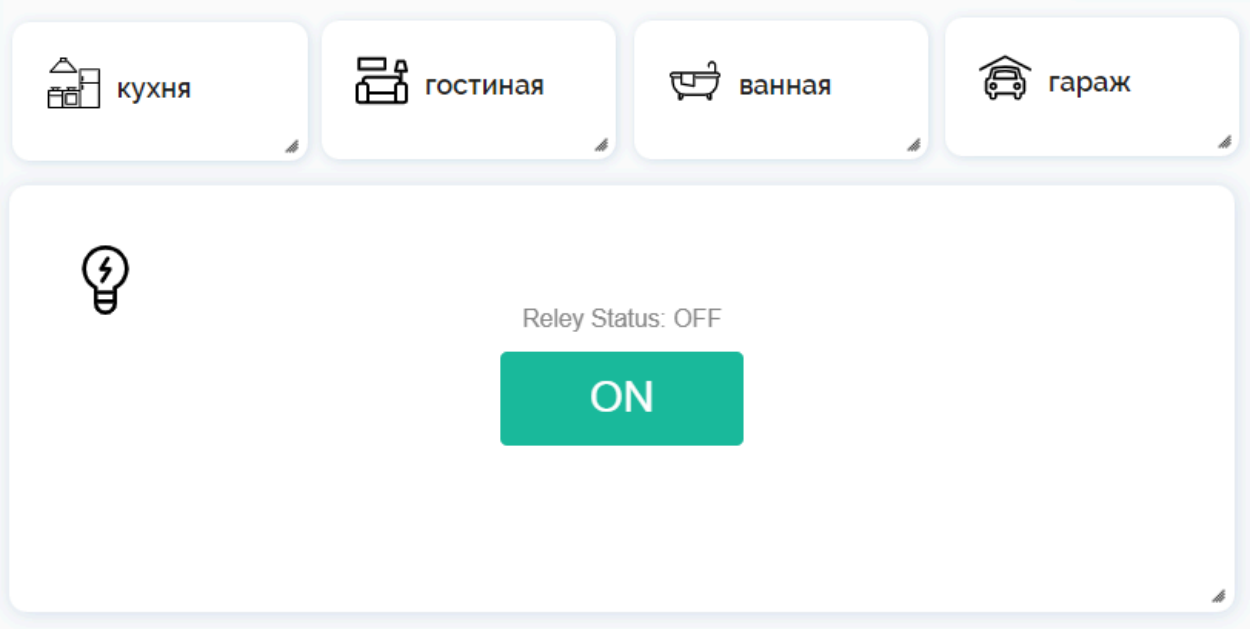

Рисунок 25 – Итоговый размер и положение фрейма «реле»

Данные фреймов, созданных пользователем, выводятся над ними в левом верхнем углу, при наведении на один из них (рисунок 26). Они включают в себя два пункта: url-адрес страницы, на которую ссылается фрейм и frameID, придуманное пользователем при инициализации.

Рисунок 26 – Данные фрейма «реле»

Эти данные будут полезны, при обновлении фрейма. Если пользователь захочет сохранить размер и позиционирование блока, но изменить содержимое, ему будет достаточно использовать функцию update.
При нажатии правой кнопкой мыши на фрейм, выпадет контекстное меню с возможностью: удалить, обновить фрейм (рисунок 27).

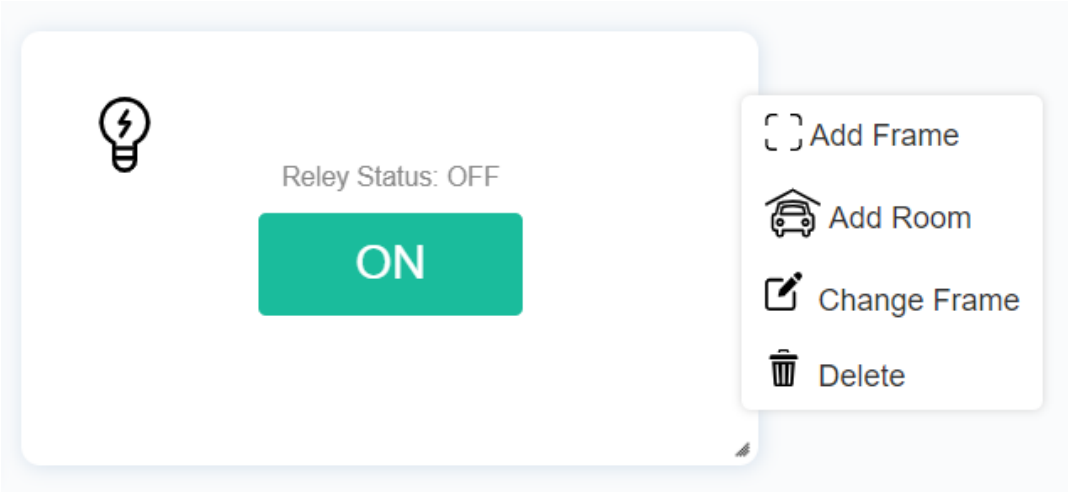

Рисунок 27 – Контекстное меню при нажатии на фрейм

Выбрав пункт Change Frame, в этом месте откроется всплывающая форма обновления фрейма (рисунок 28).

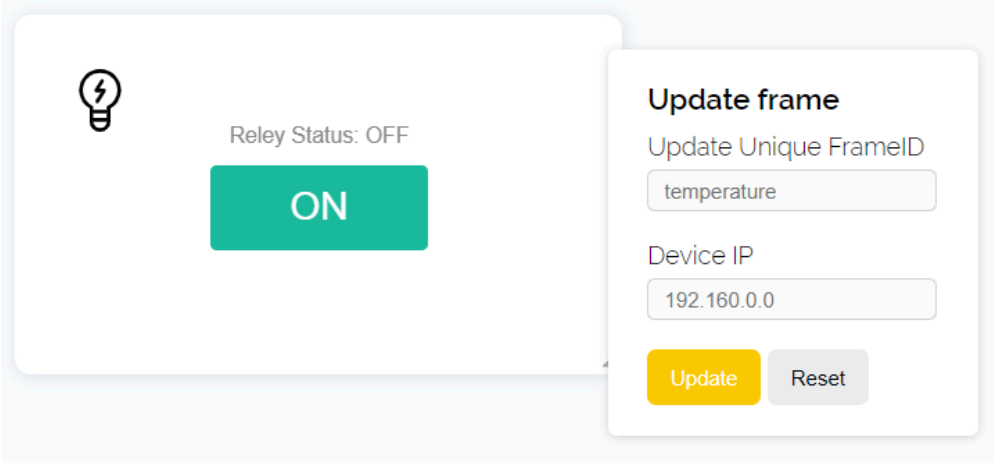

Рисунок 28 – Форма обновления фрейма

Введя новые данные и нажав кнопку update, в базу отправится sql-запрос для обновления последнего нажатого фрейма, страница обновится и php файл, получающий из базы все фреймы, принадлежащие данной

странице, заново отобразит фрейм, с уже новым именем и ip-адресом заданными пользователем (рисунок 29 и 30).

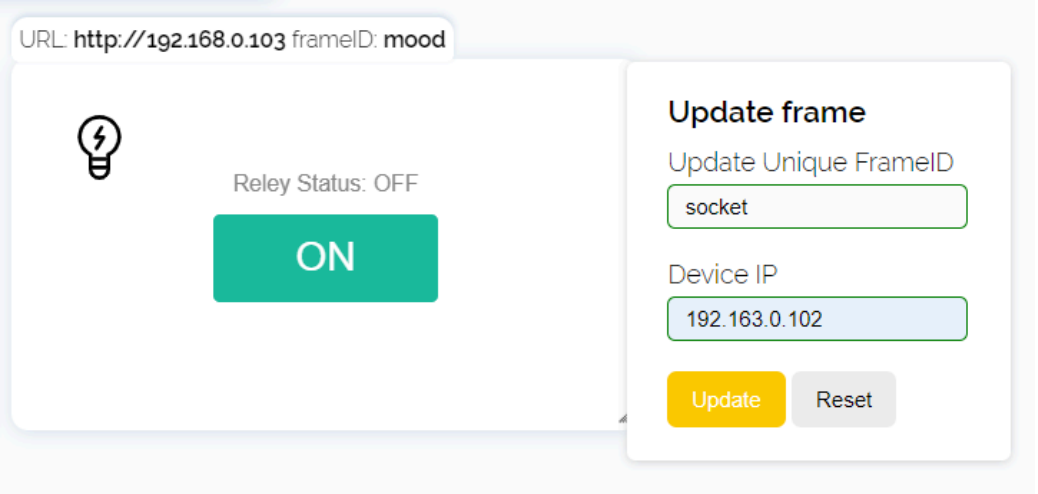

Рисунок 29 – Обновление фрейма с frameID «mood»

Рисунок 30 – Обновленный фрейм

Комнаты

Комнаты, в разрабатываемой системе, представляют из себя дополнительные whiteboard-ы для структурирования фреймов по их местоположению в пространстве. Комнаты способны объединять в себе все фреймы, относящиеся к одной конкретной категории или месту в квартире.

Пользователь сам задает имя комнаты, но предпочтительно исходить из названия мест, в которых располагаются модули: кухня, гостиная, ванная и т.д.

Пример комнат показан на рисунке 31.

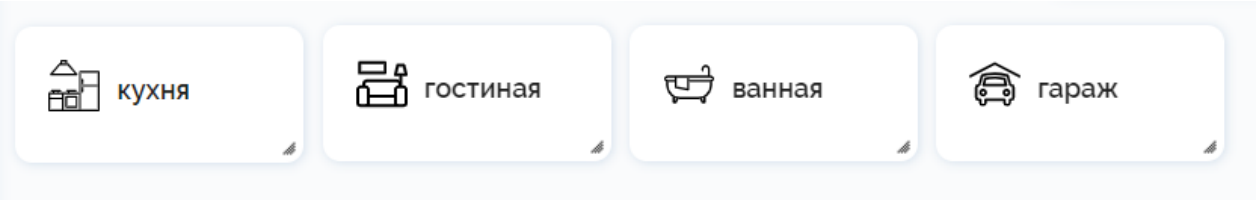

Рисунок 31 – Пример «комнат» в системе управления

Для создания новый комнаты, необходимо нажать на соответствующую иконку в шапке сайта, после чего откроется popup-форма создания комнаты (рисунок 32). Пользователю достаточно придумать название комнаты и выбрать любую иконку из предложенных. При нажатии на кнопку Add, введенные данные передадутся в php файл, который отправит в базу данных запрос на добавление (INSERT) новой комнаты в таблицу **rooms** (рисунок 33 и 34). Перед отправкой данных, файл сгенерирует уникальное значение timestamp используя, как базис, время, в которое был выполнен запрос, с включением миллисекунд и сформирует на его основе уникальный **room_id**. Благодаря этому, две или более комнат с одинаковыми именами не будут конфликтовать между собой.

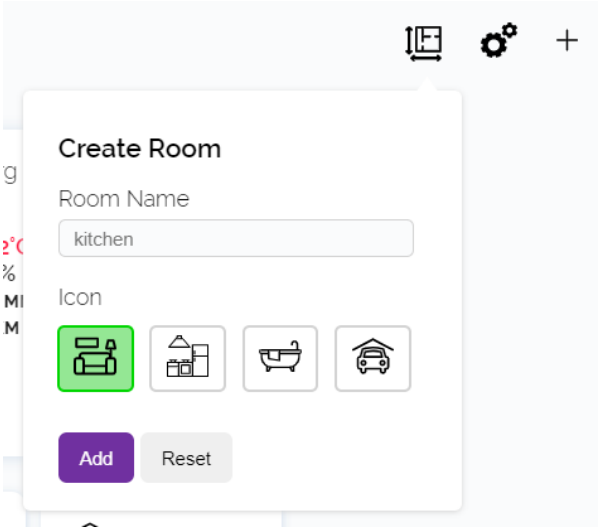

Рисунок 32 – Форма создания комнаты

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT	
2	name	varchar(255)	utf8_general_ci		Нет	*Нет*			
3	room_id	varchar(50)	utf8_general_ci		Нет	*Нет*			
4	icon_id	varchar(50)	utf8_general_ci		Нет	*Нет*			
5	room_file	varchar(255)	utf8_general_ci		Да	*NULL*			

Рисунок 33 – Структура таблицы rooms

id	name	room_id	icon_id	room_file
76	кухня	room_20214320193212	kitchen_1	kuhnya_20214320193212.php
77	гостиная	room_202143201917650	livingroom_1	gostinaya_202143201917650.php
78	ванная	room_202143201922509	bathroom_1	vannaya_202143201922509.php
79	гараж	room_202143201928371	garage_1	garazh_202143201928371.php
80	чердак	room_202143225617527	kitchen_1	cherdak_202143225617527.php

Рисунок 34 – Пример данных, получаемых таблицей rooms

При условии, что данные успешно добавлены в базу и не возникло ошибок в процессе, php-код создаст в папке: «data/rooms» новый файл с именем, сгенерированным по формуле (1):

$$f=n+t, \quad (1)$$

Где f – имя файла,
n – имя комнаты, преобразованное из кириллицы в латиницу используя функцию транслитерации,
t – уникальное значение timestamp (время отправки запроса).

Таблица rooms хранит в себе имена созданных файлов, на которые будет ссылаться при генерации веб-страницы. При загрузке html главной страницы, php-цикл foreach сгенерирует комнаты со встроенными ссылками (<a>) на данные файлы. Например, для комнаты с именем «кухня» будет

сгенерирован файл kuhnya_xxx.php, где xxx – уникальное значение timestamp. Данный файл будет доступен по адресу: «../data/rooms/kyhnya_xxx.php».

При создании новых файлов комнат, php-код использует в качестве шаблона файл room.php, расположенный в папке «../assets/templates/» в каталоге проекта. Файл задает базовые css и html будущих страниц (рисунок 35).

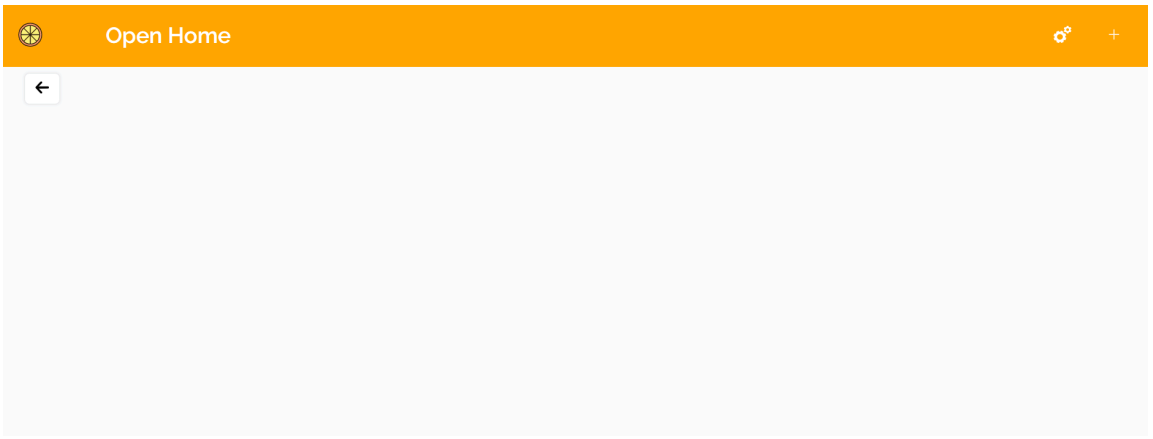

Рисунок 35 – Шаблон веб-страницы комнаты

При создании фреймов внутри комнаты, они будут доступны исключительно в ней и нигде более. Это позволит систематизировать интерфейс на более понятные подструктуры. Фреймы внутри комнаты никак не влияют на фреймы главной страницы или фреймы других комнат (рисунок 36).

Рисунок 36 – Фреймы внутри комнаты

Фреймы получают имя страницы, на которой они были созданы, через невидимый input хранящий статическое значение, уникальное для каждой страницы: для главной страницы – mainpage, для комнат – название файла, созданное по формуле (1). Структура и пример получаемых данных фреймов в базе данных представлены на рисунках 37 и 38.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT	
2	url	varchar(255)	utf8_general_ci		Нет	*Нет*			
3	frameID	varchar(50)	utf8_general_ci		Нет	*Нет*			
4	page_name	varchar(255)	utf8_general_ci		Да	*NULL*			

Рисунок 37 – Структура фреймов в таблице frames

id	url	frameID	page_name
99	http://192.168.0.103	new_weather	mainpage
117	http://192.163.0.102	socket	mainpage
118	http://192.168.0.0	car	garazh_202143201928371.php
119	http://192.168.0.103	light	garazh_202143201928371.php

Рисунок 38 – Пример получаемых данных в таблице frames

Комнаты также как и фреймы поддаются масштабированию и перемещению в пределах границ родительского блока content (рисунок 39).

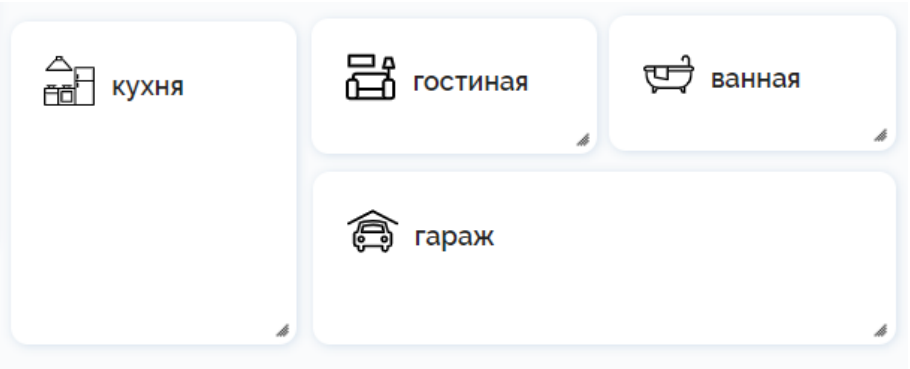

Рисунок 39 – Масштабирование и перемещение комнат

При удалении комнаты, в базу сначала посылается запрос на получение имени файла, затем комната удаляется из базы, вместе с тем, удаляются все связанные фреймы, у которых имя страницы совпадает с именем комнаты. И при успешном выполнении, php файл удалит из папки «../data/rooms» файл с именем, полученным из базы переде удалением. Это предотвращает переполнение папки ненужными файлами, а также исключает возможные конфликты имен.

Контекстное меню

В пределах блока content, нажатие правой кнопки мыши вызывает специальное меню (рисунок 40).

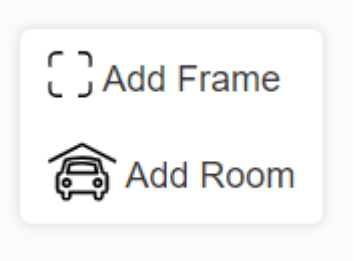

Рисунок 40 – Контекстное меню

При нажатии на Add Frame, страница прокручивается вверх и открывается форма создания фрейма. Аналогичное действие происходит по нажатию на Add Room. При нажатии правой кнопкой мыши по блокам внутри content (фреймы, комнаты), открывается расширенное контекстное меню с возможностью редактировать или удалить блок (рисунок 41 и 42).

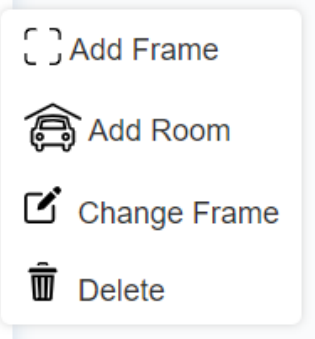

Рисунок 41 – Контекстное меню при нажатии на фрейм

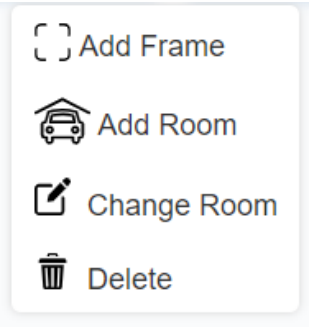

Рисунок 42 – Контекстное меню при нажатии на комнату

Виджеты

В интерфейс, для пользователя, предустановлено несколько виджетов: текущее время, время восхода и заката солнца, а также подробный прогноз погоды, включающий: температуру, влажность, давление и видимость (рисунок 43).

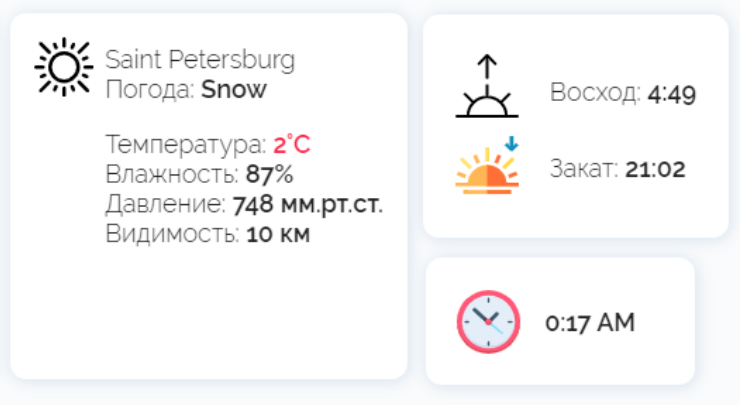

Рисунок 43 – Предустановленные виджеты

Для виджета времени, каждую секунду вызывается и обрабатывается JavaScript функция new Date().

Прогноз погоды и время восхода/заката передаются через сторонний **API** (application programming interface) принадлежащий **OpenWeatherMap**.

Используя уникальный **api key**, генерируемый для каждого зарегистрированного, в системе openweathermap, пользователя, открывается доступ к текущему прогнозу погоды в формате JSON.

Полученные данные обрабатываются и передаются в отдельный виджет.

API - Описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

JSON - текстовый формат обмена данными, основанный на JavaScript.

ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ

Расчет стоимости разработки

Стоимость дипломной работы, включающая проектирования модуля умного дома:

Минимальная стоимость самостоятельного проектирования и сборки дистанционно-управляемой розетки:

1-канальный модуль реле – 90 руб.

Понижающий ac/dc преобразователь IRM-03-5 – 400 руб.

NodeMcu v3 Lua WI-FI ESP8266 CP210 – 320 руб.

Розетка с заземлением – 150 руб.

Сетевой шнур и вилка с заземлением – 120 руб.

Провод марки ШВВП 3х0.75 – 30 руб.

Итоговая стоимость модуля: 1110 руб.

Стоимость датчиков и управляющей платы NodeMcu для передачи данных:

NodeMcu v3 Lua WI-FI ESP8266 CP210 – 320 руб.

Датчик температуры и влажности DHT11 – 90 руб.

Инфракрасный датчик движения HS-SR501 – 90 руб.

Макетная плата – 120 руб.

Резистор на 10 кОм – 2 руб./шт.

Итоговая стоимость: 622 руб.

Так как программная часть данной дипломной работы бесплатная, итоговая стоимость вычисляется из стоимости элементной базы проекта.

Итоговая стоимость дипломной работы: 1732 руб.

РАЗРАБОТКА UML ДИАГРАММ

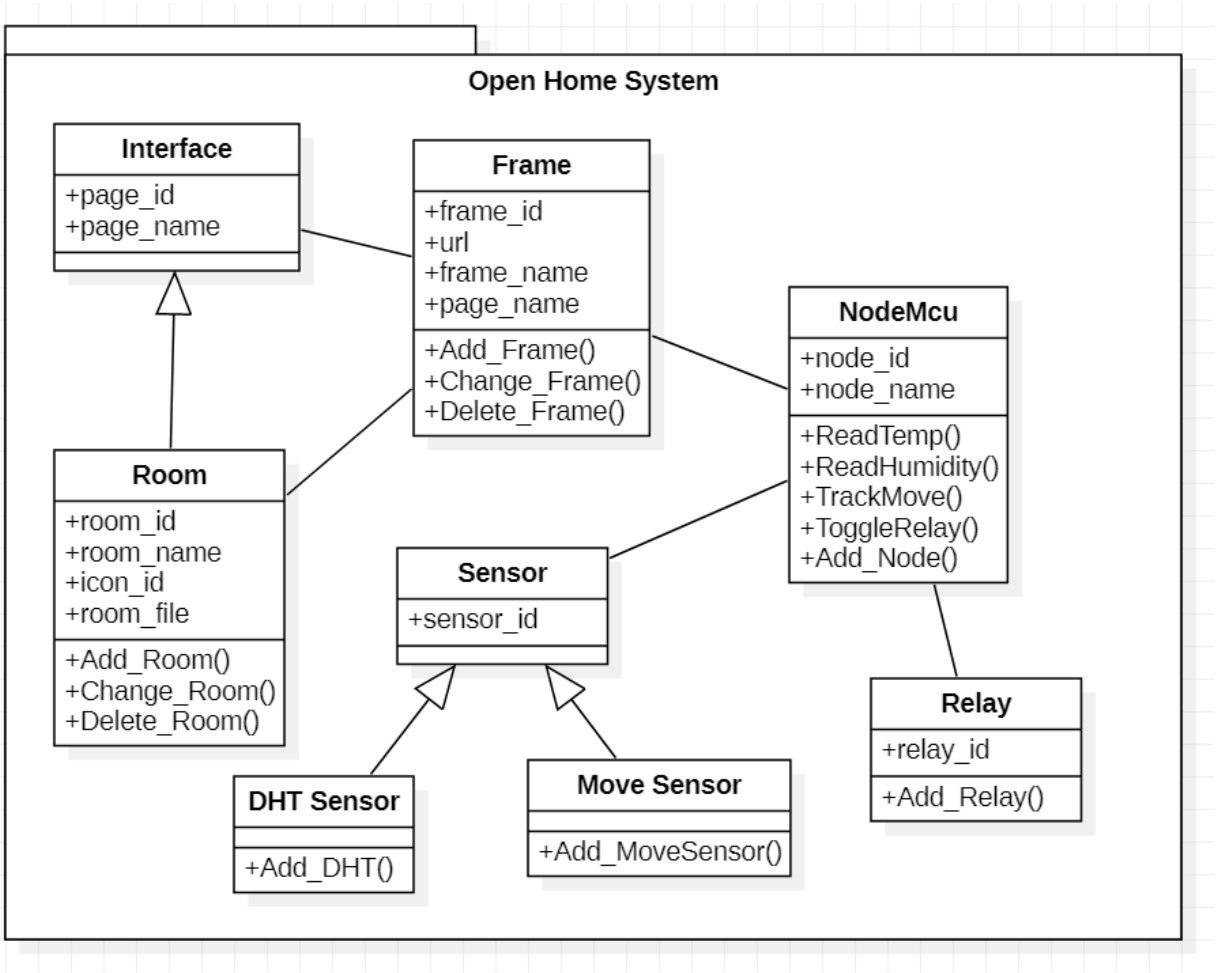

Рисунок 4.1 – UML диаграмма классов

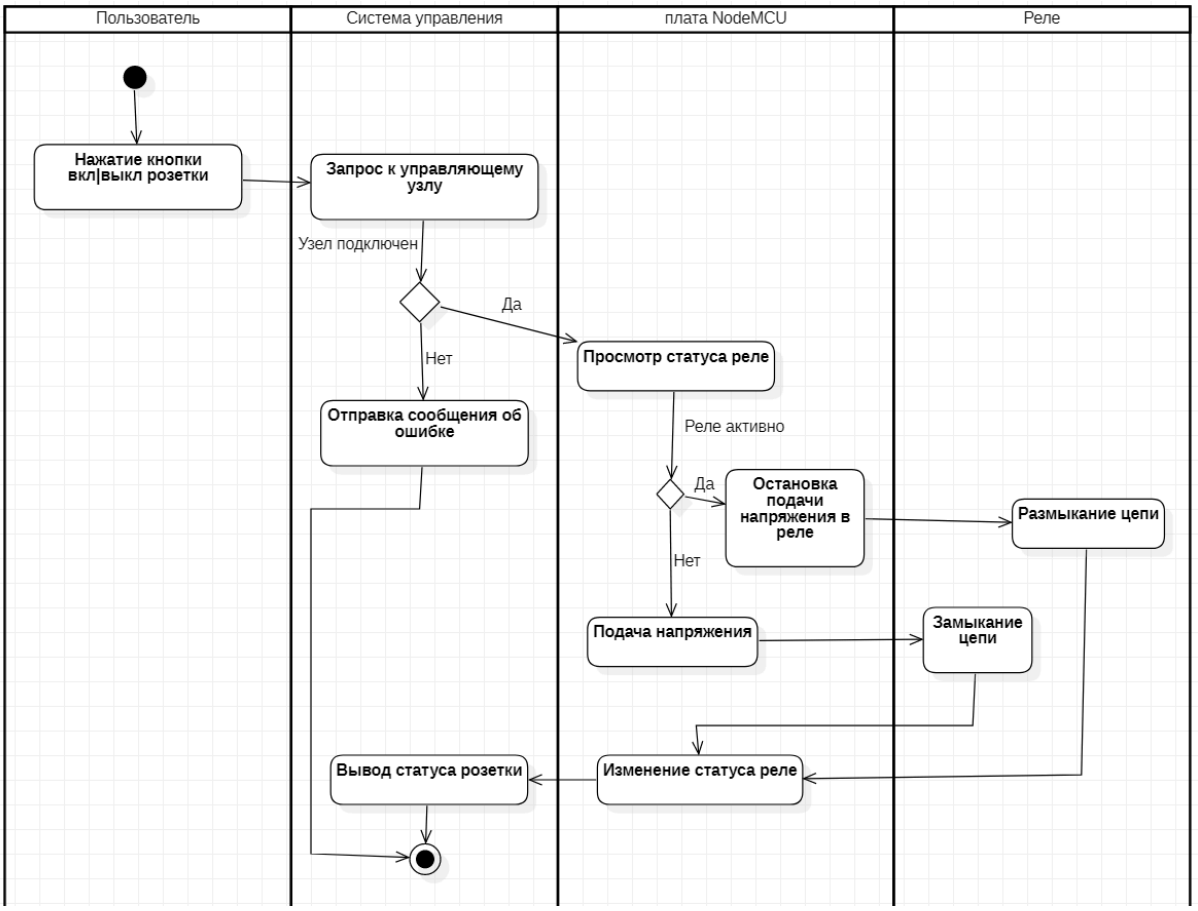

Рисунок 4.2 - UML диаграмма деятельности (Вкл./Выкл. Розетки)

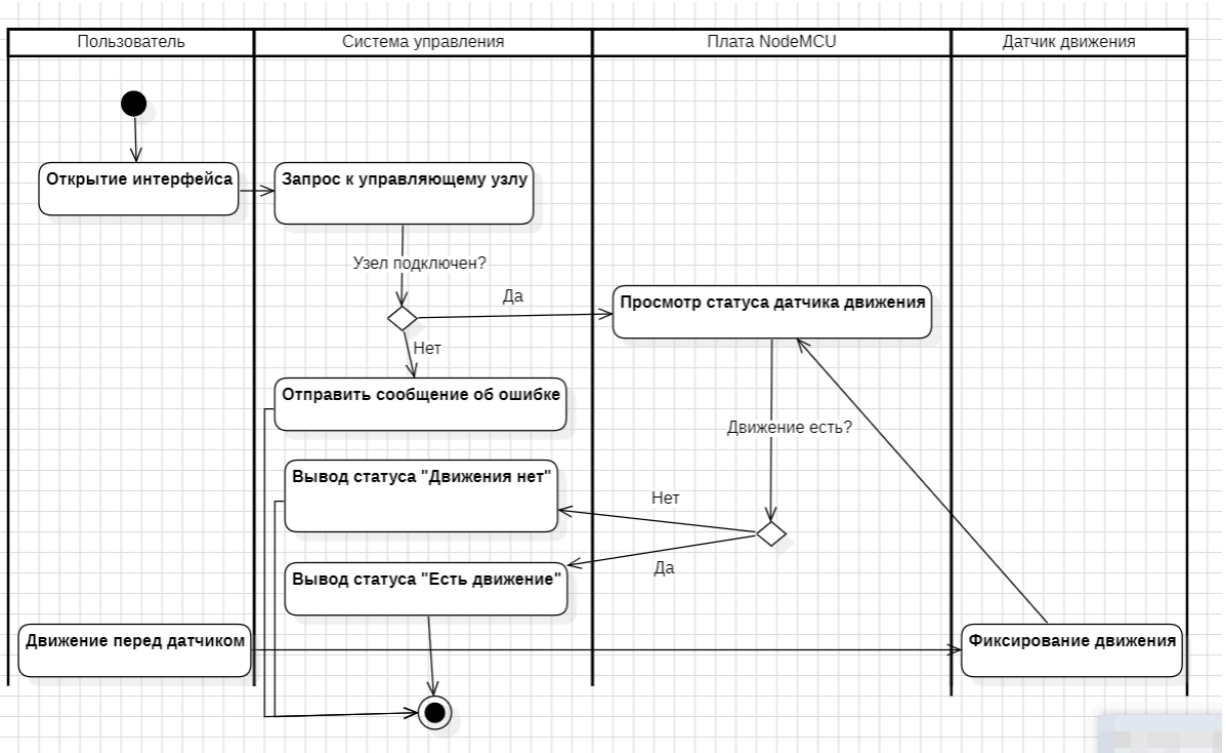

Рисунок 4.3 - UML диаграмма деятельности (Работа датчика движения)

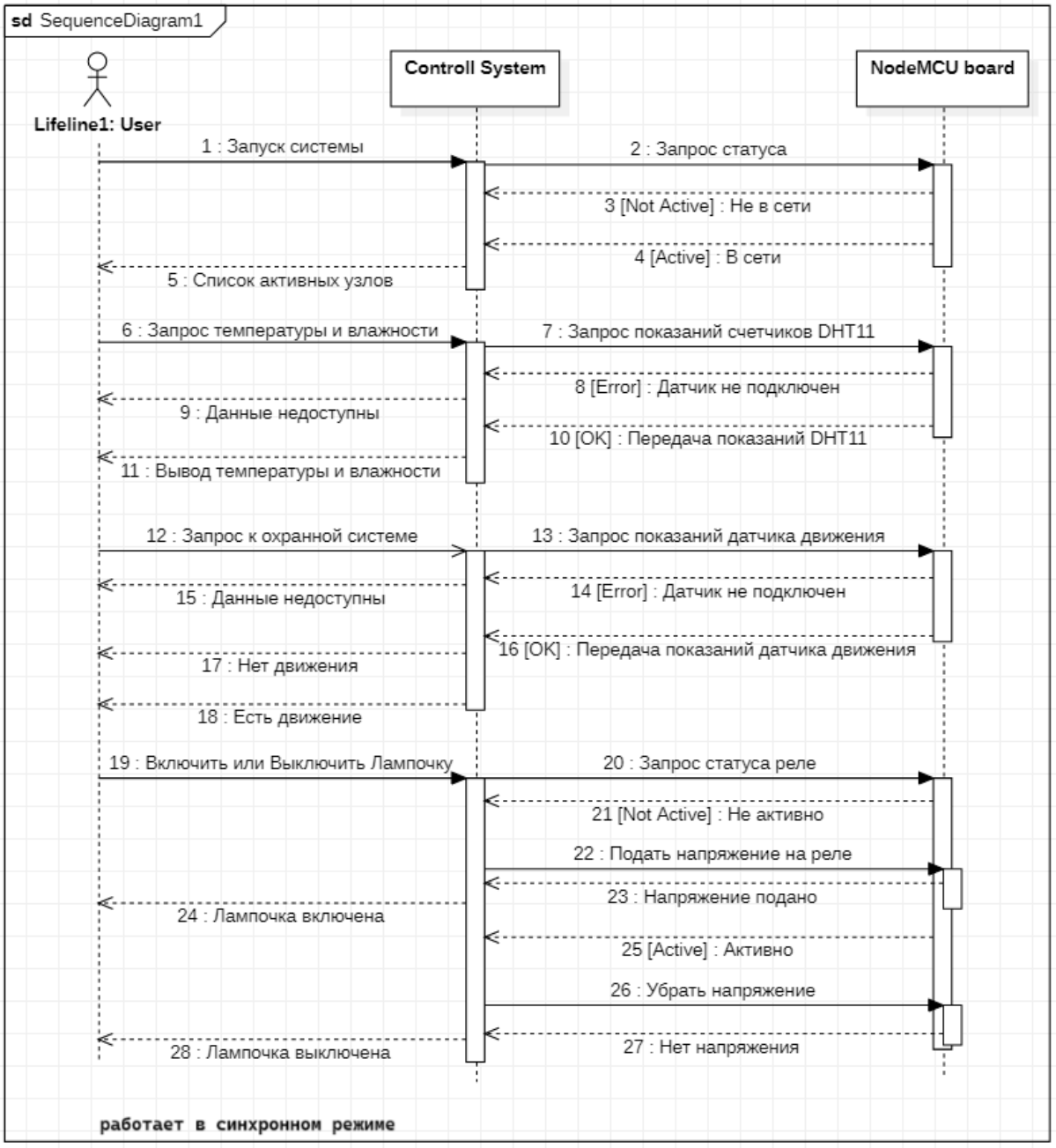

Рисунок 4.4 - UML диаграмма последовательности (Sequence Diagram)

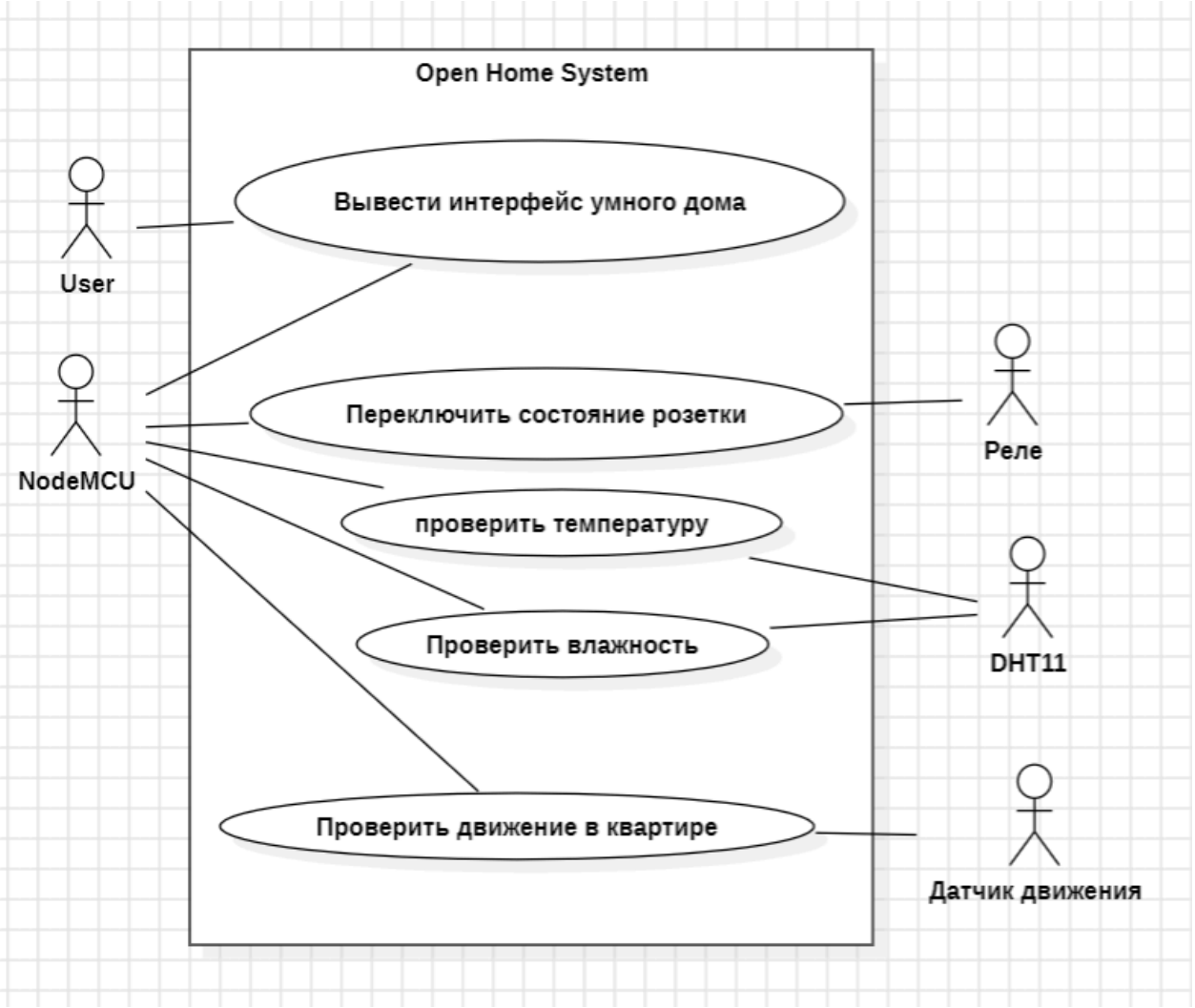

Рисунок 4.5 - UML Use Case диаграмма (Диаграмма Прецедентов)

ЗАКЛЮЧЕНИЕ

В данной дипломной работе была спроектирована и разработана система управления умным домом, был разработан веб-интерфейс для единого управления всеми модулями, подключенными к сети.

Был спроектирован и разработан модуль умного дома, управляемый удаленно по сети, для тестирования.

Разработаны веб-страницы для каждого модуля по отдельности, объединенные затем в единый интерфейс.

Устройства, собранные в рамках данной дипломной работы, не имеют дорогостоящих комплектующих и являются экономически выгодными по сравнению с коммерческими аналогами.

Результаты, полученные при дипломном проектировании, дают возможность в дальнейшем масштабировать систему управления и беспрепятственно добавлять в нее новые модули. Интерфейс прост в использовании и в управлении.

Заявленные задачи, в ходе работы были выполнены.

Таким образом, можно утверждать, что поставленная цель дипломной работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бирюкова С., Макаренцева А., Третьякова Е. Кто в доме хозяин / Бирюкова С., Макаренцева А., Третьякова Е. [Электронный ресурс] // IQMedia : [сайт]. — URL: https://iq-media.ru/archive/206810985.html (дата обращения: 15.02.2025).
2. Internet of things / [Электронный ресурс] // Википедия : [сайт]. — URL: https://en.wikipedia.org/wiki/Internet_of_things (дата обращения: 17.02.2025).
3. Перри Л. Архитектура Интернета Вещей [Текст] / Перри Л. — 1-е изд.. — Москва: ДМК Пресс, 2019 — 456 с.
4. Rob Van Kranenburg The Internet of Things: A Critique of Ambient Technology and the All-seeing Network of RFID [Текст] / Rob Van Kranenburg — . — : Institute of Network Cultures, 2007 — 60 с.
5. Интернет вещей на ESP8266 / [Электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/companies/masterkit/articles/256931/ (дата обращения: 05.03.2025).
6. / [Электронный ресурс] // ESP8266 Community Forum : [сайт]. — URL: https://www.esp8266.com/ (дата обращения: 12.03.2025).
7. Документация / [Электронный ресурс] // JQuery Documentation : [сайт]. — URL: https://jquery-docs.ru/ (дата обращения: 25.03.2025).
8. Документация / [Электронный ресурс] // JQuery User Interface : [сайт]. — URL: https://jqueryui.com/ (дата обращения: 14.04.2025).
9. Руководство по PHP / [Электронный ресурс] // php.net : [сайт]. — URL: https://www.php.net/manual/ru/index.ph (дата обращения: 17.04.2025).
10. OpenWeather One Call API Documentation / [Электронный ресурс] // OpenWeather : [сайт]. — URL: https://openweathermap.org/api/one-call-api (дата обращения: 02.05.2025).

ПРИЛОЖЕНИЕ А

Техническое задание на разработку

Цель дипломной работы: разработать веб-интерфейс для управления модулями умного дома через единую централизованную систему.

В соответствии с поставленной целью, были определены следующие задачи:

Спроектировать и создать модуль умного дома для тестирования систем

Разработать дизайн интерфейса управления

Спроектировать и запрограммировать функционал системы управления.

Проверить работоспособность системы на практике с помощью таких модулей как: датчики, платы nodemcu и модули умного дома.

В ходе проектной деятельности выполняются следующие работы:

1. Спроектировать модуль умного дома – дистанционно-управляемая розетка на безе платы NodeMcu, 1-канального реле и понижающего ac/dc преобразователя.
2. Подключить датчики температуры, влажности и движения к плате nodemcu. Создать html-страницу для вывода информации, получаемой с датчиков DHT11 и HS-SR501.
3. Разработать веб-интерфейс с использованием средств стандартизированного языка разметки документов HTML и формального языка описания внешнего вида документа CSS.
4. Спроектировать реляционную базу данных системы, хранящую информацию на локальном сервере, с использованием средств MySQL и языка программирования SQL.
5. Протестировать работоспособность системы на локальном сервере с использованием программной платформы OpenServer.

Требования к языкам программирования и среде разработки программы.

Исходный код программы для модулей умного дома должен быть написан на языке программирования C++. В качестве среды разработки должна быть использована среда Arduino IDE.

Веб-интерфейс будет написан с использованием HTML, CSS и JavaScript, как наиболее удобных и распространенных средств создания веб-страниц.

База данных будет спроектирована в реляционной системе MySQL, с использованием веб-интерфейса для администрирования СУБД – phpMyAdmin.

Локальный сервер будет запущен и протестирован на программной платформе OpenServer. Open Server Panel – это портативная программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций. OpenServer был выбран благодаря удобному многофункциональному интерфейсу, широкому спектру возможностей по администрированию и настройке компонентов, большому сообществу разработчиков и накопленной за долгие годы информационной базе проекта. Он прост в управлении и его использовании не потребует от пользователя специализированных знаний.

Связь веб-интерфейса с базой данных будет осуществляться путем отправки SQL-запросов через php-файлы. PHP - скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Требования к дизайну сайта

При разработке сайта должны быть использованы преимущественно светлые тона. Должна быть предусмотрена возможность изменять визуальный стиль в процессе использования.

Основной функционал интерфейса должен быть доступен с первой страницы.

Блоки контента (фреймы, комнаты) должны быть выполнены в светлых тонах.

Общая стилистика сайта должна использовать средства material design.

Блоки должны быть независимы друг от друга.

Функциональные требования

Требования к представлению главной страницы сайта

Главная страница сайта должна содержать графическую часть, меню управления, а также контентную область для того, чтобы пользовать с первой страницы мог начать работу с системой.

Контентная область главной страницы должна содержать предустановленные блоки (фреймы) с информацией о:

- времени
- погоде
- времени суток

В правом верхнем углу страницы отображаются кнопки управления системой:

- создание новой комнаты
- настройки
- создание нового фрейма

Блоки внутри контента страницы, должны быть масштабируемыми. Изменение размеров должно происходить путем растягивания элемента с помощью мышки. При перезагрузке страницы они должны сохранять полученный размер.

Все элементы, содержащиеся в пределах блока контент должны поддерживать функцию drag-and-drop. **Drag-and-drop** - способ оперирования элементами интерфейса при помощи манипулятора «мышь» или сенсорного экрана. При переносе элемента на другую позицию, его положение должно сохраняться и после перезагрузки страницы.

Требования к структуре сайта

Сайт должен иметь следующую структуру:

a. Главная страница

a.1 Виджет времени

a.2 Виджет погоды

a.3 Виджет времени суток

a.4 Комнаты (создаются пользователем)

a.5 Фреймы (создаются пользователем)

b. Комната

b.1 Фреймы (создаются пользователем)

Требования к системе управления

Для управления интерфейсом должны быть предусмотрены следующие функции:

- Создание новых фреймов на главной странице
- Создание новых комнат на главной странице
- Создание новых фреймов в комнатах
- Удаление фреймов
- Удаление комнат
- Изменение фреймов (обновление информации)
- Изменение масштабов фреймов и их положения на странице
- Изменение масштабов комнат и их положения на странице
- Отображение информации о фреймах
- Изменение стиля страниц (выбор темы)

Управление наполнением интерфейса

Для управления наполнением веб-интерфейса должны быть предусмотрены следующие блоки:

А. поле ввода информации для создания (обновления) фрейма:

А.1 поле ввода frameID (VARCHAR) – имя фрейма

А.2 поле ввода Device IP (VARCHAR) – ip-адрес

Б. поле ввода информации для создания комнаты

Б.1 поле ввода Room Name (VARCHAR) – имя комнаты

Б.2 radio-button с выбором иконки комнаты

Поля фрейм-элемента должны редактироваться в появляющемся pop-up меню.

Требования к хранению данных

Все данные сайта должны храниться в структурированном виде под управлением реляционной СУБД. Исключения составляют файлы данных, динамически созданные при добавлении новой комнаты на главную страницу, а также svg-иконки. Файлы сохраняются в файловой системе, а в базе размещаются их имена, по которым, при загрузке странице, комнатам присваиваются ссылки. SVG сохраняются в бд в виде идентификаторов – уникальных для каждого нового изображения, по которым позже будут отображаться html-элементы на веб-странице.

Руководство пользователя

Наименование: Open Home

Описание: Данная система представляет из себя веб-интерфейс, получающий данные с управляющей платы, которая в свою очередь, получает информацию от подключенных к сети модулей. Каждый из модулей выполняет определенную функцию: сбор данных с датчиков, управления реле для подачи тока в розетку и так далее.

Система управления создана для более простого взаимодействия пользователя с имеющимися у него «умными» устройствами, будь то розетка на дистанционном управлении или освещение, управляемое через wi-fi.

Возможности системы: В данной сборке реализован и протестирован веб-интерфейс для отображения показаний с различных датчиков (температуры и влажности, движения и т.п.), удаленное управления реле по средствам переключения состояния кнопки в веб-интерфейсе. Система способна добавлять любое количество новых модулей для управления различными устройствами. Интерфейс позволяет объединить все имеющиеся приборы и датчики с модулем wi-fi в единую централизованную сеть с топологией «звезда» - все устройства присоединены к центральному узлу (ноутбуку, одноплатному компьютеру Rasberry Pi и т.п.).

Уровень подготовки пользователя: Данная система предполагает наличие у пользователя базовых навыков программирования микроконтроллеров, и владение компьютером на уровне уверенного пользователя.

Назначения и условия применения

Система управления умным домом предназначена для:

- Контроля за подключенными модулями
- Сбора информации с датчиков
- Вывод этой информации в веб-интерфейсе

Интерфейс представляет из себя набор параметров, собираемых с модулей, объединенных в единую сеть и управляемых посредством веб-страницы с изолированными друг от друга блоками данных (фреймами).

Собранная информация визуально оформлена для лучшего понимания пользователем.

Для работы всей системы необходимо наличие у пользователя плат с микроконтроллером esp8266 для подключения к wi-fi или создания точки доступа. Примером такого контроллера является – NodeMcu любой версии.

Подготовка к работе

Необходимые устройства и программы для начала работы:

- Программируемая плата с микроконтроллером esp для передачи данных по wi-fi
- Любой браузер (Google Chrome, Yandex Browser, Opera, Firefox и т.п.)
- Питание для плат (включая возможность питания от ноутбука)
- Датчики, с которых планируется считывание информации, модули реле
- Среда разработки Arduino IDE для загрузки скетчей на плату
- Провод micro-usb type b для первоначальной загрузки кода на платы. (В дальнейшем, при желании, можно настроить загрузку кода по воздуху с помощью протокола OTA)
- Open Server – локальный сервер для развертывания веб-интерфейса

Для начала работы с системой управления умным домом, необходимо:

- Скачать с официального сайта и установить на свой компьютер Open Server
- Скачать архив данного проекта Open Home и разархивировать его в папке OSPanel/domains
- Запустить Open Server

На данном этапе, если все шаги выполнены, в браузере пользователя станет доступен адрес – http://openhome.esp/

Перейти по адресу http://openhome.esp/. Должен открыться стартовый интерфейс веб-приложения.

Для настройки работы виджетов необходимо:

- Зарегистрироваться на сайте openweathermap.org
- На сайте, во вкладке API, выбрать пункт One Call API (нажать кнопку Subscribe)
- Выбрать ценовой вариант Free (бесплатный)
  (После этого на email, указанный при регистрации придет письмо с уникальным ключом api. Его нужно сохранить).
- Открыть файл weather.js в корневом каталоге проекта
- В строчку с «appid», после двоеточия, в двойных кавычках, вставить ключ api, пришедший на почту после шага 3.

Чтобы выбрать город проживания:

- Перейти в папку assets/json/
- Открыть json файл в архиве - **city.list.json.gz**
- Воспользоваться поиском по файлу (Ctrl + F). Ввести название города в английской транскрипции
- Найдя нужный город, скопировать значение из строки «id»
- Вставить полученное значение в **weather.js** в строчку «id» аналогичным appid способом.
- Проверить работу API. При перезагрузке страницы с веб-интерфейсом, в предустановленных фреймах должны отобразиться: подробный отчет по погоде в городе, время восхода и заката.

Для создания нового модуля умного дома:

- Скачать и загрузить скетч для управляющей платы, на имеющийся микроконтроллер, предварительно внеся изменения в код, поменяв номера пинов для получения и отправки данных в соответствии с используемой схемой подключения. А также

сменив ssid и пароль wi-fi, задав данные домашней сети (при использовании платы в роли станции).

- Загрузить скетчи для каждого модуля в соответствии с выполняемой им задачей (считывания показаний датчиков, переключение реле)
- Проверить работоспособность системы, подключив шлюз к компьютеру через usb и проверив данные в COM-порту.
- Получив ip-адрес из COM-порта, ввести его в поисковую строку любого используемого браузера. Данный адрес является ссылкой на локальный сервер модуля. При успешном отображении html-страницы с данными, можно подключать модуль к интерфейсу.

Добавление нового модуля в интерфейс:

На данном этапе, должен быть установлен и запущен Open Server с разархивированной в нем папкой проекта. А также создан и проверен новый модуль, который планируется добавить в интерфейс управления.

- Открыть веб-интерфейс в браузере, перейдя по адресу – http://openhome.esp
- В открывшейся странице, нажать в правом верхнем углу по кнопке «+».
- Ввести в открывшейся popup-форме frameID – придумать имя устройства для отображения в интерфейсе, и Device IP – ip-адрес устройства, полученный при создании нового модуля (рисунок 44).
- Нажать кнопку Add или клавишу Enter, после чего страница обновится и новый фрейм автоматически добавиться на доску content.

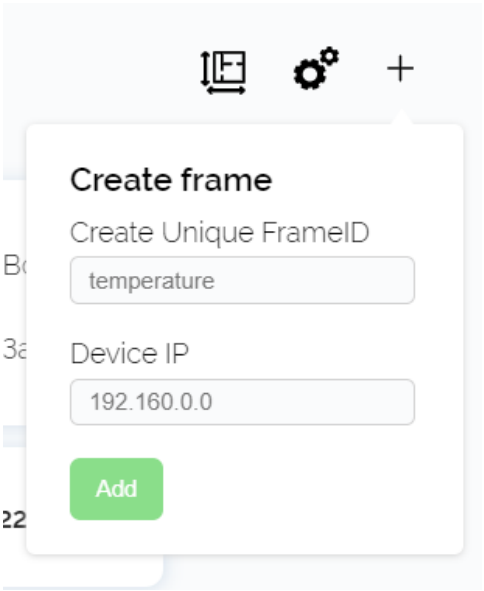

Рисунок 44 – Создание нового фрейма (руководство пользователя)

Добавление новой комнаты в интерфейс:

- На главной странице веб-интерфейса нажать на крайнюю левую кнопку в правом верхнем углу экрана (рисунок 45).
- В открывшейся форме ввести Room Name – придумать название комнаты (например: гостиная).
- Выбрать иконку, которая будет отображаться для этой комнаты.
- При неудовлетворительном результате – нажать кнопку Reset. Она сбросит все введенные данные.
- Когда данные введены, нажать кнопку Add или Enter на клавиатуре. Страница обновится, и новая комната автоматически добавится на доску.

При нажатии на комнату, откроется новая страница, где можно создать фреймы, принадлежащие модулям, находящимся в едином физическом пространстве.

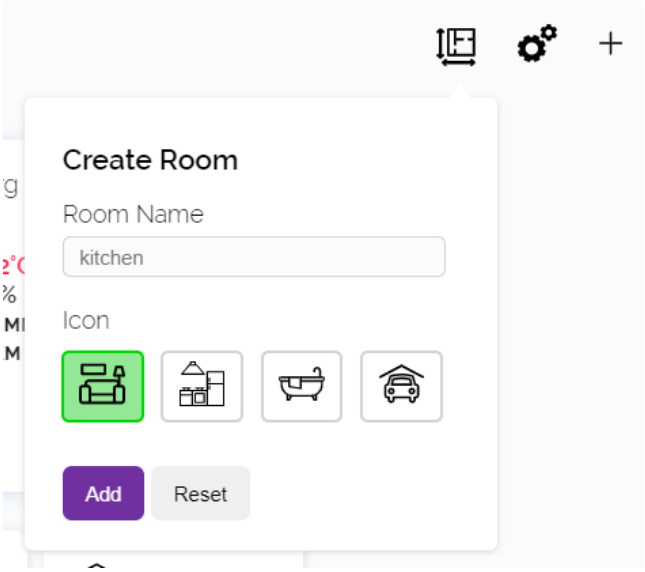

Рисунок 45 – Создание новой комнаты (руководство пользователя)

Удаление блоков

Чтобы удалить ранее созданный блок (фрейм, комнату), достаточно кликнуть правой кнопкой мыши у краев элемента. Это откроет контекстное меню (рисунок 46), в котором необходимо выбрать и нажать пункт Delete, после чего страница обновится и блок будет удален.

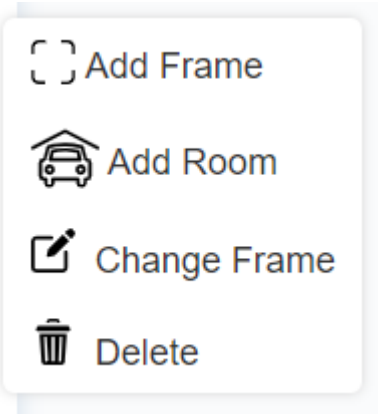

Рисунок 46 – Контекстное меню (руководство пользователя)

Изменение фрейма

Чтобы изменить фрейм, необходимо нажать правой кнопкой мыши у краев элемента и выбрать из выпадающего контекстного меню, пункт – **Change Frame**. После чего, в popup-форме **Update Frame** (рисунок 47),

ввести новые значения (frameID и Device IP) и нажать кнопку **Add** (или **Enter**). Страница обновится и фрейм будет иметь обновленные значения.

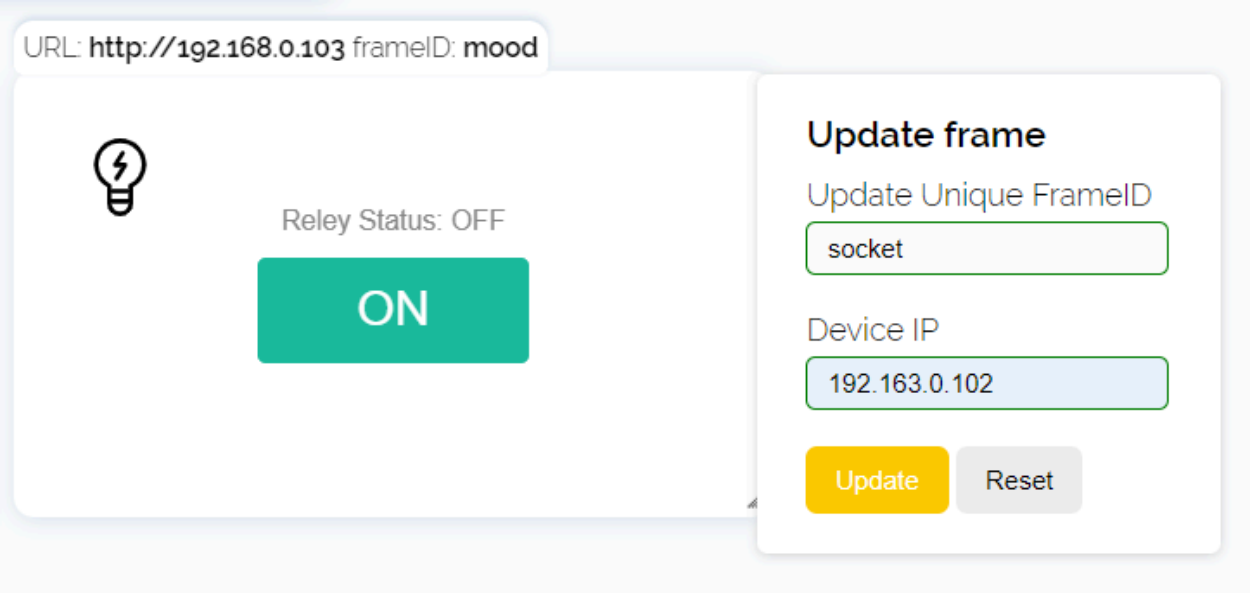

Рисунок 47 – Обновление фрейма (руководство пользователя)

Изменения темы (внешнего вида веб-страницы)

- В правом верхнем углу нажать на кнопку – «Настройки»
- В выпадающей форме выбрать один из трех вариантов цветовой палитры интерфейса: classic, dark, light
- Нажать кнопку **Change** (или **Enter**)

Страница обновится, после чего общий визуальный стиль измениться.

Замечание: Стилям нужно время, чтобы загрузиться. Иногда необходимо обновить страницу еще раз, чтобы они применились.

Нештатные ситуации при работе с системой

Список возможных ситуаций при работе с системой управления:

- Один из модулей в сети произвольно отключился
- Модуль системы перестал отсылать данные с подключенных датчиков или с текущим состоянием реле

Варианты решения:

- Отключение одного из модулей никак не повлияет на работу системы в целом. Чтобы восстановить соединение, следует перезагрузить плату с модулем esp8266 и, удостоверившись в ее функционировании, перезагрузить страницу веб-интерфейса. (Некоторым модулям требуется время, чтобы загрузиться на сайте. Если фрейм не передает ошибку превышения времени ожидания (рисунок 23), скорее всего страница еще не загрузилась).
- При остановке передачи данных, необходимо проверить соединения датчиков с платой, подключенной к системе. Нужно удостовериться, что все проводы подсоединены к нужным пинам. Если проблем не обнаружено, необходимо перезагрузить плату, не передающую данные. После перезагрузки, обновить страницу, дождаться повторного подключения платы к интерфейсу и проверить работу.

Рекомендации по освоению

Рекомендуемые темы для изучения:

- Программирование микроконтроллеров ESP8266
- Основы веб-программирования (HTML/CSS)
- Основы программирования в среде Arduino IDE

Рекомендуемая литература:

- Техническая документация платы NodeMcu ESP8266

ПРИЛОЖЕНИЕ Б

Исходные коды программных модулей

Б.1 Отправка данных с датчиков на локальный сервер

```
void handle_OnConnect()
{
  Temperature = dht.readTemperature(); // получить значение
температуры
  Humidity = dht.readHumidity();       // получить значение
влажности

  Move = digitalRead(D7); //движение 1 - есть, 0 - нет
  MoveStat = "Движения нет";
  if (Move) {
    MoveStat = "В квартире есть движение";
  }
  else {
    MoveStat = "Движения нет";
  }

  server.send(200, "text/html",
SendHTML(Temperature,Humidity, MoveStat));

//срабатывание светодиода при движении (проверка датчика)
  if (digitalRead(D7)) {
    digitalWrite(D2, HIGH);
    Serial.println("Есть движение!");
  }
  else {
    digitalWrite(D2, LOW);
  }
```

```
  delay(1000);
}

void handle_NotFound()
{
  server.send(404, "text/plain", "Not found");
}
```

Б.2 Листинг Index.php (главная страница)

```
<!-- load current theme -->
<?php include_once "load_themes.php" ?>
<!--  -->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>OpenHome Control Panel</title>
<!-- jquery ui css file -->
<link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.9.2/themes/base/jquery-ui.c
ss"/>
<!--  -->
<link rel="stylesheet" href="./css/style.css">

<!-- theme style test -->
<?php $themes = get_theme();?>
<?php foreach($themes as $theme): ?>
<link rel="stylesheet"
href="css/themes/<?=$theme['style']?>.css" id="theme">
<?php endforeach; ?>
```

```
<!--  -->
</head>

<!-- get data from db -->
<?php
include_once "iframe_url.php";
?>
<?php include_once "load_rooms.php" ?>
<!--  -->
<body>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.mi
n.js"></script>
<!-- DRAG & DROP LIBRARY -->
<script
src="libs/jquery-ui.1.12.1/jquery-ui.min.js"></script>

<div class="topline">
<div class="main_logo">
<img class="logo" src="img/mainpage_img/orange2.svg"
alt="logo">
<h1 class="main_title">Open Home</h1>
</div>

<div class="main_mnu">
<div class="add_room_btn__wrapper"></div>
<div class="add_room">
<div class="add_room_btn">
<svg>
</svg>
</div>
<div class="add_room__open popup_block">
<form action="create_room.js" id="createRoomForm">
```

```
    <h3 class="add_room__header">Create Room</h3>
    <label for="roomName">Room Name</label><br>
    <input type="text"
        class="create_room_input class_input"
        placeholder="kitchen"
        id="roomName"
        required
        maxlength="20">

    <br><br>

    <label class="create_room_label">Icon</label>
    <br>
    <!-- 1 -->
    <input class="room_type_radio" type="radio"
name="room_type" id="livingroom" checked
value="livingroom_1">
    <label class="room_type_label" for="livingroom">
        <svg>
        </svg>
    </label>

    <!-- 2 -->
    <input class="room_type_radio" type="radio"
name="room_type" id="kitchen" value="kitchen_1">
    <label class="room_type_label" for="kitchen">
        <svg>
        </svg>
    </label>

    <!-- 3 -->
    <input class="room_type_radio" type="radio"
name="room_type" id="bathroom" value="bathroom_1">
```

```
    <label class="room_type_label" for="bathroom">
        <svg>
        </svg>
    </label>

    <!-- 4 -->
    <input class="room_type_radio" type="radio"
name="room_type" id="garage" value="garage_1">
    <label class="room_type_label" for="garage">
        <svg>
        </svg>
    </label>

    <br><br>

    <input type="submit" name="createRoomButton" value="Add"
class="btn create_room_btn">
    <input type="reset" name="cancelCreateRoom"
value="Reset" class="btn cancel_room_btn">
</form>
</div>
</div>

<div class="settings">
<div class="settings_btn__wrapper"
id="settings_wrapper"></div>
<div class="settings_btn">
<svg>
</svg>
</div>
<div class="settings__open popup_block">
<form action="css/style_change.js" id="settingsForm">
    <h3 class="settings_header">Settings</h3>
```

```
    <!--  -->
    <label class="settings_theme_label">Theme (Reload page
after)</label>
    <br><br>
    <!-- 1 -->
    <input
        class="theme_type_radio"
        type="radio"
        name="theme_type"
        id="classic_theme"
        value="classic"
        <?php $themes = get_theme();?>
        <?php foreach($themes as $theme): ?>
            <?php if ( $theme['style'] == 'classic' ) {
                echo "checked";
            }?>
        <?php endforeach; ?> >
    <label class="thememe_type_label"
for="classic_theme">classic</label>
    <br><br>
    <!-- 2 -->
    <input
        class="theme_type_radio"
        type="radio"
        name="theme_type"
        id="dark_theme"
        value="dark"
        <?php $themes = get_theme();?>
        <?php foreach($themes as $theme): ?>
            <?php if ( $theme['style'] == 'dark' ) {
                echo "checked";
            }?>
        <?php endforeach; ?> >
```

```
     <label class="thememem_type_label"
for="dark_theme">dark</label>
     <br><br>
     <!-- 3 -->
     <input
          class="theme_type_radio"
          type="radio"
          name="theme_type"
          id="light_theme"
          value="light"
          <?php $themes = get_theme();?>
          <?php foreach($themes as $theme): ?>
               <?php if ( $theme['style'] == 'light' ) {
                    echo "checked";
               }?>
          <?php endforeach; ?> >
     <label class="thememem_type_label"
for="light_theme">light</label>

     <br><br>
     <input type="submit" name="settingsButton"
value="Change" class="btn settings_submit">
</form>
</div>
</div>

<div class="add_block__btn" id="svg_wrapper"></div>
<div class="add_block">
<svg>
</svg>
<div class="add_block__open popup_block">
<form action="create_frame.js" id="createFrameForm">
     <h3 class="add_block__header">Create frame</h3>
```

```
    <label for="frameName">Create Unique FrameID</label><br>
    <input type="text"
        class="create_frame_input class_input"
        placeholder="temperature"
        id="frameName"
        required
        maxlength="20">

    <br><br>

    <label for="frameIP">Device IP</label><br>
    <input type="text"
        class="create_frame_input ip_input"
        placeholder="192.160.0.0"
        id="frameIP"
        required

pattern="\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2
[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]
[0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b">

    <br><br>
    <!-- input with page_name value -->
    <input type="text" value="mainpage" style="display:
none;" id="pageNameVal">
    <!--  -->
    <input type="submit" name="createFrameButton"
value="Add" class="btn create_frame_btn">
</form>
</div>
</div>
</div>
</div>
```

```
<!-- блоки на главной странице -->
<div class="bg">
<div class="content">
<iframe src="http://192.168.0.102" frameborder="0"
id="home_weather" class="weather_frame mainframe preset_frame
drag drop ui-widget-content"></iframe>
<iframe src="http://192.168.0.103" frameborder="0" id="releу"
class="releу_frame mainframe preset_frame drag drop
ui-widget-content"></iframe>

<div class="city_weather_frame mainframe preset_frame drag
drop ui-widget-content" id="city_weather">
<div id="weather"></div>
<img src="img/sun.svg" alt="weath" class="cityWeatherImg">
</div>

<div class="sun_position_frame mainframe preset_frame drag
drop ui-widget-content" id="sun_position">
<div class="sun__icons">
<img src="img/daytime.svg" alt="sunrise" width="40px">
<img src="img/sunset.svg" alt="sunset" width="40px">
</div>
<div id="sun" class="sunstate"></div>
</div>

<div class="clock_frame mainframe preset_frame drag drop
ui-widget-content" id="clock">
<svg>
</svg>
<div class="clock_frame_text" id="clock_content"></div>
</div>
```

```
<div class="Db__content_wrapper">
<!-- DB loaded frames -->
<div class="frameDb__content">
<?php $frames = get_frames('mainpage');?>
<!-- $i - incremented element for unique IDs of resizable
elements -->
<?php $i=0;?>
<!--  -->
<?php foreach($frames as $frame): ?>
     <div class="mainframe resizable_content frame_initial
drag" id="resize_<?php echo $i;?>">
          <div class="frame_stat">URL:
<b><?=$frame["url"]?></b>  frameID:
<b><?=$frame["frameID"]?></b> </div>
          <iframe
               src="<?=$frame["url"]?>"
               id="<?=$frame["frameID"]?>"
               class="<?=$frame["frameID"]?>_frame
mainframe_content ui-widget-content drop"
               scrolling="no"
               frameborder="0">Ваш браузер не поддерживает
Фреймы</iframe>
     </div>
     <?php $i++;?>
<?php endforeach; ?>
</div>
<!--end DB load -->

<!-- ROOMS (loaded from db) -->
<div class="roomDb__content">
<?php $rooms = get_rooms();?>
<!-- $j - incremented element for unique IDs of resizable
elements -->
```

```
<?php $j=0;?>
<!--  -->
<?php foreach($rooms as $room): ?>
     <div class="mainroom resizable_content room_initial
drag" filename="<?=$room['room_file']?>"
id="resize_room_<?php echo $j;?>" style="overflow: hidden;">
          <a href="/data/rooms/<?=$room['room_file']?>"
class="mainroom_link"></a>
          <div
               class="mainroom__content drop
ui-widget-content"
               id="<?=$room['room_id']?>"
               style="height:100%; width:100%;">
               <div class="mainroom__icon">
               <?php if ($room["icon_id"] == "kitchen_1"): ?>
                         <svg>
                         </svg>
          <?php elseif ($room["icon_id"] == "livingroom_1"):
?>
                         <svg>
                         </svg>
          <?php elseif ($room["icon_id"] == "bathroom_1"): ?>
                         <svg version="1.1" id="Capa_1">
                         </svg>
               <?php elseif ($room["icon_id"] == "garage_1"):
?>
                         <svg version="1.1" id="Capa_1">
                         </svg>
                    <?php endif;?>
               </div>
          <div
class="mainroom__name"><?=$room["name"]?></div>
          </div>
```

```
     </div>
     <?php $j++;?>
<?php endforeach;?>
</div>
<!--end ROOMS -->
</div>

<!-- Change Frame Popup-Menu -->
<div class="update_block__open popup_block">
<form action="custom_mnu.js" id="updateFrameForm">
<h3 class="update_block__header">Update frame</h3>
<label for="updateframeName">Update Unique
FrameID</label><br>
<input type="text"
     class="create_frame_input class_input"
     placeholder="temperature"
     id="updateframeName"
     required
     maxlength="20">

<br><br>

<label for="updateframeIP">Device IP</label><br>
<input type="text"
     class="create_frame_input ip_input"
     placeholder="192.160.0.0"
     id="updateframeIP"
     required

pattern="\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2
[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]
[0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b">
```

```
<br><br>

<input type="submit" name="updateFrameButton" value="Update"
class="btn update_frame_btn">
<input type="reset" name="cancelUpdateFrame" value="Reset"
class="btn cancel_frame_btn">
</form>
</div>
<!-- /end change frame popup -->

<!-- invis input to get last clicked block type, id and
filename -->
<input type="text" value="" id="lastFrameValue"
style="display: none">
<input type="text" value="" id="lastBlockType"
style="display: none">
<input type="text" value="" id="lastFileName" style="display:
none;">
<!-- invis unput to get page style -->
<input type="text" value="classic" id="pageStyle"
style="display: none;">
</div>
</div>
<!-- /блоков -->

<!-- Custom context menu -->
<ul class="custom-menu" id="frameContextMnu">
<li data-action="add" class="custom-menu__item"
id="addFrameItem">
<svg>
</svg>
<div class="custom-menu__text">Add Frame</div>
</li>
```

```
<li data-action="addRoom" class="custom-menu__item"
id="addRoomItem">
<svg>
</svg>
<div class="custom-menu__text">Add Room</div>
</li>
<li data-action="change" class="custom-menu__item
iframe_custom" id="changeFrameItem">
<svg>
</svg>
<div class="custom-menu__text">Change Frame</div>
</li>
<li data-action="changeRoom" class="custom-menu__item
room_custom" id="changeRoomItem">
<svg>
</svg>
<div class="custom-menu__text">Change Room</div>
</li>
<li data-action="delete" class="custom-menu__item
content_block_custom" id="deleteFrameItem">
<svg>
</svg>
<div class="custom-menu__text">Delete</div>
</li>
</ul>

<script>
$("document").ready(function() {
var size = JSON.parse(localStorage.size || "{}");

$.each(size, function (id, siz) {
$("#" + id).css(siz)
})
```

```
$('.resizable_content').resizable({
containment: ".content",
scroll: true,
stop: function (event, ui) {
size[this.id] = ui.size
localStorage.size = JSON.stringify(size)
}
});

//сохранять позицию элемента в LocalStorage
var sPositions = localStorage.positions || "{}",
positions = JSON.parse(sPositions);
$.each(positions, function (id, pos) {
$("#" + id).css(pos)
})
$(".drag").draggable({
containment: ".content",
scroll: false,
stop: function (event, ui) {
positions[this.id] = ui.position
localStorage.positions = JSON.stringify(positions)
}
});
});
</script>

<script src="weather.js"></script>
<script src="create_frame.js"></script>
<!-- custom menu js -->
<script src="custom_mnu.js"></script>
<!-- clock frame -->
<script src="clock.js"></script>
```

```
<!-- create new room -->
<script src="create_room.js"></script>
<!-- change page theme -->
<script src="css/style_change.js"></script>
</body>
</html>
```

Б.3 Листинг create_room.js (добавление комнаты)

```
$(document).ready(function() {
     var roomForm = $('#createRoomForm');

     //Открытие формы-----------------------------
     var roomBtn = $('.add_room_btn__wrapper');
     var addRoom = $('.add_room__open');

     roomBtn.on('click', function() {
          if ( addRoom.hasClass('visible') ) {
               addRoom.removeClass('visible');
          } else {
               addRoom.addClass('visible');
          }
     });

     $(document).click(function (e) {
          if ( !roomBtn.is(e.target) && !addRoom.is(e.target)
&& addRoom.has(e.target).length === 0
               && !$('#addRoomItem').is(e.target) &&
$('#addRoomItem').has(e.target).length === 0) {
               addRoom.removeClass('visible');
          };
     });
```

```
document.querySelector("#createRoomForm").addEventListener("s
ubmit", function(e){
          e.preventDefault();
     });
     //-----Создание комнаты----------------------
     $(roomForm).submit(createRoom);
     function createRoom() {
          let roomName = $('#roomName').val();
let selectedIcon =
$("input[name='room_type']:checked").val();
var now = new Date();
let timestamp = now.getFullYear().toString();
timestamp += (now.getMonth < 9 ? '0' :
'')+now.getMonth().toString();
timestamp += ((now.getDate < 10) ? '0' :
'')+now.getDate().toString();
timestamp += ((now.getHours < 10) ? '0' :
'')+now.getHours().toString();
timestamp += ((now.getMinutes < 10) ? '0' :
'')+now.getMinutes().toString();
timestamp += ((now.getSeconds < 10) ? '0' :
'')+now.getSeconds().toString();
timestamp+=((now.getMilliseconds<10)?'0':'')+now.getMilliseco
nds().toString()
// --------------------
save_room(roomName, selectedIcon, timestamp);
roomForm[0].reset();
addRoom.removeClass('visible');
setTimeout(location.reload(), 2000);
};

//-------------------------------------------------
```

```
//Функция сохранения комнаты в БД (ajax)
     function save_room(roomName, selectedIcon, timestamp){
$.post("create_room.php", {roomName: roomName, selectedIcon:
selectedIcon, timestamp: timestamp}).done(function(data){
               console.log(data);
          });
     }
});
```

Б.4 Листинг create_room.php (сохранение комнаты в базе данных)

```
<?php
     include_once 'assets/filters/translit.php';
     //connection settings
     $servername = "openhome.esp";
     $username = "username";
     $password = "password";
     $database = "openhome";

     //connection--------------------------
     $connect = mysqli_connect($servername, $username,
$password, $database);
     // Проверяем соединение
     if (!$connect) {
          die("Не удалось подключиться к MySQL: " .
mysqli_connect_error());
     }
     echo "Соединение установленно. ";

     // Fields---------------------------------
     $roomName = filter_var(trim($_POST['roomName']),
FILTER_SANITIZE_STRING);
```

```
    $selectedIcon = filter_var(trim($_POST['selectedIcon']),
FILTER_SANITIZE_STRING);
    $timestamp = filter_var(trim($_POST['timestamp']),
FILTER_SANITIZE_STRING);
    $room_id = "room_" . $timestamp;
    //---------------------------------------------------
    // add new room in data/rooms
    //---------------------------------------------------
$newcontent =
file_get_contents(dirname(__FILE__)."\assets\\templates\\room
.php");
    $roomNameLatin = translit($roomName); //translate
kirilitca to latin
    $room_file = $roomNameLatin . '_' . $timestamp . '.php';

    if (!file_exists($room_file)) {
        $handle =
fopen(dirname(__FILE__)."\data\\rooms\\".$room_file, 'w+' );
        fwrite($handle,$newcontent);
        fclose($handle);
    }
    //Query-------------------------------------------------
    $sql = "INSERT INTO `rooms` (`name`, `room_id`,
`icon_id`, `room_file`) VALUES ('$roomName', '$room_id',
'$selectedIcon', '$room_file')";

    if (mysqli_query($connect, $sql)) {
        echo "Новая комната добавлена. ";
    } else {
        echo "Error: " . $sql . "<br>" .
mysqli_error($connect);
    }
```

```
     //Closing connection
     mysqli_close($connect);
?>
```

Б.5 Листинг load_rooms.php (получение списка комнат для страницы из бд)

```
<?php
     function get_rooms()
     {
          //connection settings
          $servername = "openhome.esp";
          $username = "username";
          $password = "password";
          $database = "openhome";
//connection-------------------------mysqli_report(MYSQLI_RE
PORT_ERROR | MYSQLI_REPORT_STRICT);
$connect = mysqli_connect($servername, $username, $password,
$database);
     // Проверяем соединение
          if (!$connect) {
               die("Не удалось подключючиться к MySQL: " .
mysqli_connect_error());
          }
          echo "Соединение установленно. ";

//Query---------------------------------------------
          $query = "SELECT * from `rooms`";
          $result = mysqli_query($connect, $query);

          if ($result) {
               $rooms = [];
```

```
            while ($rum = $result->fetch_assoc()) {
                $rooms[] = $rum;
            }
        }
        else {
        printf("Сообщение ошибки: %s\n", mysqli_error(
$link ) );
        }
        return $rooms;

        //Closing connection
        mysqli_close($connect);
    }
?>
```

Б.6 Листинг weather.js (подключение API openweathermap)

```
$(document).ready(function() {
    $.get("https://api.openweathermap.org/data/2.5/weather",
        {
            "id": "498817",
            "appid": "place for api key"
        },
        function(data){
            //получение времени восхода и заката
            let unixrise = data.sys.sunrise;
            let unixset = data.sys.sunset;
            let sunrise = new Date(unixrise*1000);
            let sunriseMin = sunrise.getMinutes();
    sunriseMin = (sunriseMin < 10 ) ? "0"+sunriseMin :
sunriseMin;
            let sunriseHour = sunrise.getHours();
```

```
                let sunriseTime = sunriseHour + ':' +
sunriseMin;

                let sunset = new Date(unixset*1000);
                let sunsetMin = sunset.getMinutes();
          sunsetMin = (sunsetMin < 10 ) ? "0"+sunsetMin :
sunsetMin;
                let sunsetHour = sunset.getHours();
                let sunsetTime = sunsetHour + ':' + sunsetMin;
                //---------------------------------
                let sun='';
                sun +='Восход: <b>' + sunriseTime + '</b><br>';
                sun +='Закат: <b>' + sunsetTime + '</b><br>';
                $('#sun').html(sun);
                //вывод информации о погоде в регионе
                let out='';
                out += (data.name)+'<br>';
                out +='Погода:
<b>'+data.weather[0].main+'</b><br>';
                out +='<p style="text-align:center"<img
src="https://openweathermap.org/img/w/'+data.weather[0].icon+
'.png"></p>';
                out +='Температура: <span
style="color:#FF1F44"><b>'+Math.round(data.main.temp-273)+'&#
176;C</b></span><br>';
                out +='Влажность:
<b>'+data.main.humidity+'%</b><br>';
                out +='Давление:
<b>'+Math.round(data.main.pressure*0.00750063755419211*100)+'
мм.рт.ст.</b><br>';
                out += 'Видимость:
<b>'+(data.visibility/1000)+' км</b><br>';
          $('#weather').html(out);
```

```
});
});
```