Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |  |  |
| --- | --- | --- | --- |
| Институт: | ИРЭ | Кафедра: | Основ радиотехники |
| Направление подготовки: | | 11.04.01 Радиотехника | |

**ОТЧЕТ по практике**

|  |  |
| --- | --- |
| **Наименование практики:** | Учебная практика: научно-исследовательская работа (получение первичных навыков научно-исследовательской работы) |

**СТУДЕНТ**

|  |  |
| --- | --- |
|  | / Бутин А.А. / |
| *(подпись )* | (*Фамилия и инициалы*) |

|  |  |
| --- | --- |
| Группа | ЭР-11м-21 |
|  | *(номер учебной группы)* |

**ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ**

|  |
| --- |
|  |
| *(отлично, хорошо, удовлетворительно, неудовлетворительно)* |

|  |  |
| --- | --- |
|  | / Стрелков Н.О. / |
| *(подпись )* | (*Фамилия и инициалы члена комиссии*) |

|  |  |
| --- | --- |
|  | / Копылова Е.В. / |
| *(подпись )* | (*Фамилия и инициалы члена комиссии*) |

**Москва 2022**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«МЭИ»**

Кафедра Основ радиотехники

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_ / Шалимова Е.В. /

*(подпись) ФИО*

« 09 » февраля 2022 г.

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

на учебную практику: научно-исследовательскую работу (получение первичных навыков научно-исследовательской работы)

для Бутин Александр Анатольевич

*(ФИО обучающегося полностью)*

Направление подготовки/специальность 11.04.01 Радиотехника

курс 1 , группа ЭР-11м-21

Наименование предприятия (организации, учреждения):

Национальный исследовательский университет «МЭИ»

Подразделение: ИРЭ, кафедра Основ радиотехники

Сроки прохождения практики: 09.02.2022 – 07.06.2022

Цель прохождения практики: приобретение первичных навыков самостоятельной научной деятельности.

Тема НИР: Разработка компонентов и открытого кода для организации модульного решения «Умная дача» на базе систем с открытой архитектурой

Содержание практики, вопросы, подлежащие изучению:

1. Сравнительный подбор аппаратных компонентов для организации модульного решения «Умная дача» на базе систем с открытой архитектурой [1] (Arduino, STM32, и т.п.).

2. Сравнительный подбор программных компонентов для организации модульного решения «Умная дача» на базе систем с открытой архитектурой [2] (Arduino, STM32, и т.п.).

3. Выработка итоговых предложений по структуре создаваемого решения [3].

Планируемые результаты практики:

1. Аналитический обзор программных и аппаратных компонентов модульного решения «Умная дача», демонстрация итоговой структуры решения.

Рекомендуемая литература:

1. Петин В.А. Проекты с использованием контроллера Arduino. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2016. – 464 с.: ил.

2. Иго Т. Arduino, датчики и сети для связи устройств: Пер. с англ. — 2-е изд. — СПб.: БХВ-Петербург, 2015. —544 с.: ил.

3. Петин В. Создание умного дома на базе Arduino. ДМК Пресс. 2018.

Срок предоставления отчета о прохождении практики:

« 22 » декабря 2021 г.

Научный руководитель НИР:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Стрелков Н.О /

*(подпись) ФИО*

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Бутин А.А. /

*(подпись) ФИО*

ГРАФИК прохождения ПРАКТИКИ

|  |  |  |
| --- | --- | --- |
| Номер  п/п | Перечень работ в соответствии с заданием на практику | Отметка о выполнении работы  (выполнено /не выполнено) |
| 1 | Инструктаж по разъяснению индивидуального задания на выполнение научно-исследовательской работы, содержания и оформления отчета |  |
| 2 | Анализ индивидуального задания и составление плана выполнения научно-исследовательской работы |  |
| 3 | Сравнительный подбор аппаратных компонентов для организации модульного решения «Умная дача» на базе систем с открытой архитектурой (Arduino, STM32, и т.п.) |  |
| 4 | Сравнительный подбор программных компонентов для организации модульного решения «Умная дача» на базе систем с открытой архитектурой (Arduino, STM32, и т.п.) |  |
| 5 | Выработка итоговых предложений по структуре создаваемого решения. |  |
| 6 | Анализ полученных результатов. Подготовка и оформление отчета по НИР |  |
| 7 | Защита отчета |  |

|  |  |  |
| --- | --- | --- |
| Руководитель практики (от МЭИ) |  | / Стрелков Н.О. / |
|  | *(подпись )* | (*Фамилия и инициалы*) |

**Аннотация**

Научная исследовательская работа на тему: «Разработка компонентов и открытого кода для организации модульного решения «Умная дача» на базе систем с открытой архитектурой».

Целью данной работы является предварительный подбор аппаратных и программных компонентов для организации модульного решения «Умная дача» на базе систем с открытой архитектурой и подбор предложений по структуре создаваемого решения.

Во введении описывается актуальность темы, формируются цели и задачи на работу.

В первой главе рассмотрены аппаратные решения для рассматриваемой системы.

Во второй главе описываются наиболее удачные модульные решения периферийных устройств для системы.

В третьей главе рассматриваются способы коммутации модулей в одной системе.

В четвертой главе приводится аналитическое решение заданной задачи, а именно модульного решения «умная дача».

В пятой главе приводится альтернативное решение заданной задачи.

Работа содержит 60 страниц, 30 рисунков, 3 таблицы, 14 источников используемой литературы и 5 приложений.

[ВВЕДЕНИЕ 7](#_Toc105365523)

[1. Аппаратная часть 8](#_Toc105365524)

[2. Периферийные устройства 17](#_Toc105365525)

[2.1. Датчики 17](#_Toc105365526)

[2.1.1. Датчики температуры и влажности 18](#_Toc105365527)

[2.1.2. Датчик задымления 25](#_Toc105365528)

[2.1.3. Датчик движения 30](#_Toc105365529)

[2.1.4. Датчик влажности почвы 33](#_Toc105365530)

[2.2. Актуаторы 35](#_Toc105365531)

[2.2.1. Реле-модули 35](#_Toc105365532)

[3. Организация обмена данными 37](#_Toc105365533)

[3.1. ESP-MESH 37](#_Toc105365534)

[3.2. Протокол LoRaWAN 41](#_Toc105365535)

[3.3. ESP-NOW 44](#_Toc105365536)

[3.4. Протокол ZigBee 48](#_Toc105365537)

[4. Моделирование системы с использованием протокола ESP-NOW 51](#_Toc105365538)

[4.1. Схема подключения устройств и размещения на участке 51](#_Toc105365539)

[4.2. Обмен данными в системе 54](#_Toc105365540)

[5. Моделирование системы с использованием протокола ZigBee 57](#_Toc105365541)

[Заключение 59](#_Toc105365542)

[Список использованных источников 60](#_Toc105365543)

[ПРИЛОЖЕНИЕ А 62](#_Toc105365544)

[ПРИЛОЖЕНИЕ Б 66](#_Toc105365545)

[ПРИЛОЖЕНИЕ В 69](#_Toc105365546)

[ПРИЛОЖЕНИЕ Г 71](#_Toc105365547)

[ПРИЛОЖЕНИЕ Д 74](#_Toc105365548)

[ПРИЛОЖЕНИЕ Е 75](#_Toc105365549)

# ВВЕДЕНИЕ

Интернет вещей (англ. Internet of Things, IoT) – это это глобальная сеть компьютеров, датчиков (сенсоров) и исполнительных устройств (актуаторов), связывающихся между собой с использованием интернет протокола IP (Internet Protocol) [1].

На сегодняшний день понятие Интернета вещей находится у всех на слуху. И это не удивительно, ведь самое распространенное применение Интернета вещей находят в системах «умный дом». Во многих домах и квартирах установлены полноценные системы умного дома.

В последние года тенденция распространения умных домов стала еще больше, так как на рынке появились бюджетные решения систем в виде «умных колонок», например, Алиса или Алекса, к которым довольно просто можно присоединить «умные» модули дома (смарт лампы, датчики движения, «умные» замки и.т.д.).

Одной из распространенных проблем в системах Интернета вещей является коммутация отдельных блоков системы. В приведенных выше системах коммутация отдельных блоков осуществляется посредством нахождения всех модулей системы в одной локальной Wi-Fi сети. В данной работе будут рассмотрены способы коммутации систем с открытой архитектурой в единой системе «умной» дачи.

Также в работе будут рассмотрены периферийные модули для модели системы «умная дача» и организация передачи данных между ними и приложением на смартфоне.

В роли центрального процессора, обрабатывающего данные датчиков и управляющего исполнительными устройствами, будет выступать Arduino-совместимая плата WeMos D1 R1. Модель системы подобрана таким образом, что между двумя блоками системы нет возможности передачи данных по проводам. Поэтому в работе будут рассмотрены иные способы организации передачи данных в системе.

# Аппаратная часть

**WeMos D1 R1**

Любая система умного дома должна собирать, обрабатывать и передавать данные в сеть. Эти функции в «умных» системах выполняет центральный контроллер. В рассматриваемой в данной работе модели в качестве центрального контроллера будем использовать WeMos D1 R1 (Рисунок 1.1). Выбор платы объясняется удобной работой с Wi-Fi сетями, которые в дальнейшем будут рассматриваться для передачи данных между датчиками и сетью.

WeMos – это китайская компания, которая производит Arduino-совместимые микроконтроллеры с широким спектром возможностей. Главным достоинством плат WeMos является расширение функционала для Arduino-совместимых устройств, а также их ценовая доступность. Блочная схема микроконтроллера WeMos дает возможность превратить каждый кристалл на плате устройства с записанной программой в полноценный функциональный модуль с низкой производственной себестоимостью. Интеграция таких модулей позволяет создавать на плате универсальное контрольное устройство с любой требуемой схемой управления. Функциональная гибкость микроконтроллеров WeMos заключается в возможности в любой момент внести ряд изменений в программный алгоритм без изменений в архитектуре микросхемы на плате. Под новую задачу достаточно будет загрузки новой прошивки.

Все данные для описания платы были взяты из официальной документации [2].

Основу данной платы составляет ESP8266 – микроконтроллер с интерфейсом Wi-Fi, имеющий следующие технические параметры: поддерживает Wi-Fi протоколы 802.11 b/g/n с WEP, WPA, WPA2; обладает 14 портами ввода и вывода, SPI, I2C, UART, 10-бит АЦП; поддерживает внешнюю память до 16 МБ; необходимое питание от 2,2 до 3,6 В, потребляемый ток до 300 мА в зависимости от выбранного режима [2]. Важной особенностью является отсутствие пользовательской энергонезависимой памяти на кристалле. Программа выполняется от внешней SPI ПЗУ при помощи динамической загрузки необходимых элементов программы. Доступ к внутренней периферии можно получить не из документации, а из API набора библиотек. Производителем указывается приблизительное количество ОЗУ – 50 кБ. На модуле имеется Wi-Fi антенна для покрытия Wi-Fi сетью.

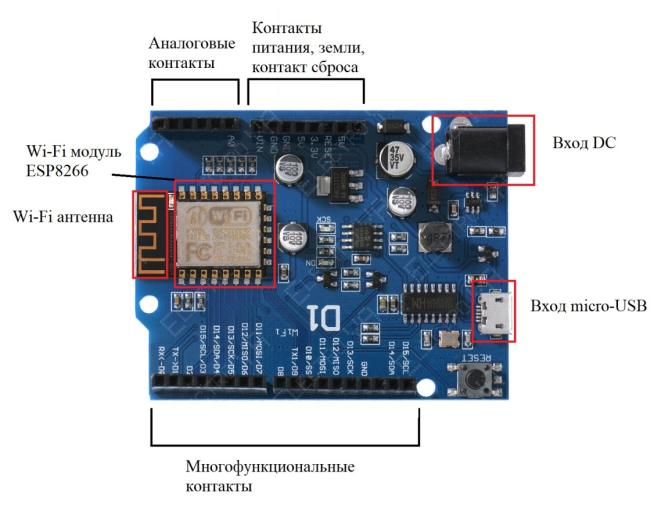


Рисунок 1.1 – плата WeMos D1 R1

Так как плата Arduino-совместимая, то нумерация контактов платы схожа с нумерацией Arduino Uno, но все же имеет ряд отличий. Самое главное отличие: увеличенное количество интерфейсов передачи данных на плате. На WeMos D1 R1 присутствует 2 группы контактов под интерфейс I2C и 2 еще 2 группы под интерфейс SPI (см. таблицу 1). Каждый цифровой контакт платы является портом общего назначения и служит для низкоуровневого обмена цифровыми сигналами с внешними устройствами (контакт GPIO – general-purpose I/O port). Так же на WeMos D1 R1 присутствует только один аналоговый контакт.

Таблица 1 – Описание контактов

|  |  |  |
| --- | --- | --- |
| Контакт WeMos D1 R1 | Контакт на плате | Специальное назначение |
| Цифровой контакт 0 | D0/GPIO3 | RX |
| Цифровой контакт 1 | D1/GPIO1 | TX |
| Цифровой контакт 2 | D2/GPIO16 |  |
| Цифровой контакт 3 | D3/GPIO5 | I2C(SCL) |
| Цифровой контакт 4 | D4/GPIO4 | I2C(SDA) |
| Цифровой контакт 5 | D5/GPIO14 | SPI (SCK) |
| Цифровой контакт 6 | D6/GPIO12 | SPI (MISO) |
| Цифровой контакт 7 | D7/GPIO13 | SPI (MOSI) |
| Цифровой контакт 8 | D8/GPIO0 |  |
| Цифровой контакт 9 | D9/GPIO2 |  |
| Цифровой контакт 10 | D10/GPIO15 | SPI (SS)/ TX |
| Цифровой контакт 11 | D11/GPIO13 | SPI (MOSI)/RX |
| Цифровой контакт 12 | D12/GPIO12 | SPI (MISO) |
| Цифровой контакт 13 | D13/GPIO14 | SPI (SCK) |
| Цифровой контакт 14 | D14/GPIO4 | I2C(SDA) |
| Цифровой контакт 15 | D15/GPIO5 | I2C(SCL) |
| Аналоговый контакт A0 | A0 |  |

Так же, как и на Arduino Uno на плате расположены контакты питания. На контакт питания 5V плата подает напряжение 5 вольт для питания внешних устройств. Соответственно на контакт 3.3V плата подает 3.3 вольта за счет встроенного стабилизатора напряжения. Три контакта GND соединены между собой и используются для подключения устройств к земле. VIN – контакт для подачи внешнего напряжения.

**Arduino UNO**

Для обработки данных с датчиков в данной работе будет использована плата Arduino Uno на базе 8-ми разрядного микроконтроллера AVR ATmega328 (см. рисунок 1.2). Преимуществом данного микроконтроллера является его дешевизна и доступность. Так же к достоинствам можно отнести низкое энергопотребление, что позволяет плате работать на автономном питании. Дальнейшее описание платы и ее характеристики взяты из официальной документации к плате [3].

Рисунок 1.2 – Плата Arduino Uno

Таблица 2 – Технические характеристики Arduino Uno

|  |  |
| --- | --- |
| Микроконтроллер | ATmega328 |
| Рабочее напряжение | 5В |
| Напряжение питания (рекомендуемое) | 7-12В |
| Напряжение питания (предельное) | 6-20В |
| Цифровые входы/выходы | 14 (из них 6 могут использоваться в качестве ШИМ-выходов) |
| Аналоговые входы | 6 |
| Максимальный ток одного вывода | 40 мА |
| Максимальный выходной ток вывода 3.3V | 50 мА |
| Flash-память | 32 КБ (ATmega328) из которых 0.5 КБ используются загрузчиком |
| SRAM | 2 КБ (ATmega328) |
| EEPROM | 1 КБ (ATmega328) |
| Тактовая частота | 16 МГц |

Описание контактов:

На плате имеются 14 цифровых входов-выходов (0-13) и 6 аналоговых (A0-A5), которые используются для подключения периферийных устройств. К каждому контакту программно может быть подключен встроенный подтягивающий резистор на 20-50 кОм. Так же для удобства некоторые контакты объединяют в себе несколько функций (например, контакты A4 и A5, которые являются аналоговыми контактами, так же используются как контакты SDA и SCL шины I2C, см. таблицу 3).

Таблица 3 – Описание контактов

|  |  |  |
| --- | --- | --- |
| Контакт Arduino | Контакт на плате | Специальное назначение |
| Цифровой контакт 0 | 0 | RX |
| Цифровой контакт 1 | 1 | TX |
| Цифровой контакт 2 | 2 | Вход для прерываний |
| Цифровой контакт 3 | 3 | Вход для прерываний |
| Цифровой контакт 4 | 4 |  |
| Цифровой контакт 5 | 5 |  |
| Цифровой контакт 6 | 6 |  |
| Цифровой контакт 7 | 7 |  |
| Цифровой контакт 8 | 8 |  |
| Цифровой контакт 9 | 9 |  |
| Цифровой контакт 10 | 10 | SPI (SS) |
| Цифровой контакт 11 | 11 | SPI (MOSI) |
| Цифровой контакт 12 | 12 | SPI (MISO) |
| Цифровой контакт 13 | 13 | SPI (SCK) |
| Аналоговый контакт A0 | A0 |  |
| Аналоговый контакт A1 | A1 |  |
| Аналоговый контакт A2 | A2 |  |
| Аналоговый контакт A3 | A3 |  |
| Аналоговый контакт A4 | A4 | I2C (SCA) |
| Аналоговый контакт A5 | A5 | I2C (SCL) |

На плате расположены контакты питания. На контакт питания 5V плата подает напряжение 5 вольт для питания внешних устройств. Соответственно на контакт 3.3V плата подает 3.3 вольта за счет встроенного стабилизатора напряжения. Три контакта GND соединены между собой и используются для подключения устройств к земле. VIN – контакт для подачи внешнего напряжения. Контакт IREF используется для информирования внешних устройств о рабочем напряжении платы.

Стоит отметить важный момент с питанием внешних устройств от платы: питание внешних устройств возможно от контактов с вольтажом в 5 и 3.3 вольта. Эти параметры будут учитываться при подборе датчиков для размещения на плате.

Питание платы:

Как видно из таблицы 2 рабочее напряжение платы – 5 вольт. Но из-за наличия встроенного стабилизатора напряжения плату можно питать от различных источников напряжением от 7 до 12 вольт. На плате размещены два разъема для подключения питания: USB-B и DC 2.1 мм. Так же для питания можно использовать контакт питания VIN. Данные способы подключения дают свободу в выборе источника питания: USB-порт компьютера, блок регулируемого напряжения, батарейка и.т.д. То есть плату можно удобно использовать как и на рабочем столе рядом с персональным компьютером, так и в местах без доступа к питанию с использованием автономного питания через вход DC.

**XBee-модуль**

Радиомодули XBee компании Digi (рисунок 1.3) относятся к классу ZigBee-модулей с уже предустановленным программным обеспечением, благодаря которому значительно сокращаются сроки разработки конечного изделия и упрощается процесс передачи данных. При этом предполагается, что модуль, в большинстве случаев, работает под управлением внешнего хост-процессора. В то же время производитель допускает загрузку в модуль собственного приложения пользователя, которое при этом должно самостоятельно взаимодействовать со стеком ZigBee, подключаемым на этапе компиляции программы.



Рисунок 1.3 – радиомодуль XBee

Основной режим работы модулей XBee — это работа под управлением внешнего микроконтроллера, управляющего модулем с помощью простых AT-команд или упорядоченных структур данных (режим API) [[14].](http://wireless-e.ru/bluetooth/xbee/)

Компания Digi разработала свой собственный фирменный (proprietary) ZigBee-профиль, позволяющий организовать прозрачную передачу данных между любыми узлами ZigBee-сети и предоставляющий доступ к цифровым и аналоговым портам ввода/вывода на удаленных узлах, работающих без хост-процессора.

Какие задачи способен выполнять XBee-модуль при использовании его без внешнего микроконтроллера? Это, прежде всего, работа с внешними датчиками, которые выдают значения параметров в виде аналогового напряжения или имеют выходы с двумя состояниями — «включено/выключено». Модуль XBee имеет мультиплексированные аналоговые (4) и цифровые (12) порты. Для управления внешними устройствами, кроме цифровых выходов, можно использовать 2 вывода ШИМ (10 бит). Также XBee-модуль напрямую сопрягается с любыми устройствами, имеющими UART-интерфейс. При самостоятельной работе XBee-модуль может передавать данные по заданному расписанию, отправляя их через определенные промежутки времени, или по изменению состояния сигнала на цифровом порту.

Мощности передатчика хватает для общения на расстоянии до 120 м на улице и до 35 м в помещении. Скорость обмена данными: до 250 кбит/с. Устройство работает на частоте 2,4 ГГц. Возможны как простые соединения «точка-точка», так и сети со сложной топологией.

Модуль работает от напряжения 2,8 - 3,4 В, потребляет 45 мА в режиме приёма, 50 мА в режиме передачи и 0,01 мА в режиме энергосбережения [14].

Подключение XBee к Arduino UNO осуществляется по последовательному интерфейсу UART. То есть для осуществления передачи данных между платами достаточно соединить контакты TX и RX платы Arduino и платы XBee. Также можно подать питание на XBee с контакта +3.3V Arduino UNO и соединить землю XBee с контактом GND.

# Периферийные устройства

К периферийным устройствам относятся любые модули системы, которые осуществляют связь физического и цифрового пространств. Периферийные устройства делятся на датчики и исполнительные устройства (актуаторы).

Датчики по своей сути являются «глазами» систем Интернета вещей. Они производят мониторинг состояния за конкретными объектами физического пространства и переводят данные об их состоянии в цифровой вид для обработки и последующих действий с ними.

Актуаторы же можно назвать «руками» систем Интернета вещей, так как они воздействуют на физические объекты посредством цифровых команд, что по сути так же является связью физического и цифрового пространств.

## Датчики

Для проектов Интернета вещей датчики являются основой, так как они устанавливают связь между физическими объектами и цифровым пространством.

Независимо от индивидуальных требований и перечня задач, которые должна решать система в целом, именно датчики обеспечивают необходимую степень автоматизации и передают другим устройствам сигнал о необходимости включения или выключения в определенный момент. [3]

На сегодняшний день на рынке радио модулей можно найти разнообразные датчики от различных производителей. Это разнообразие объясняется требованиями заказчиков. Зачастую датчики используются для пользовательских проектов, на которые наложены определенные условия в виде точности измерения и условий работы (например, мониторинг температуры в помещении, или на улице, кратковременное измерение или постоянный мониторинг).

Так как модель «умной» дачи подразумевает постоянный мониторинг различных параметров, то стоит четко обозначить область мониторинга: наблюдение за климатическими параметрами теплицы; так же в теплице необходимо следить за состоянием почвы; наблюдение за домом на участке, а именно, отслеживание задымления и присутствия посторонних на участке.

Наиболее распространенными климатическими датчиками на рынке являются: цифровой датчик температуры DS18B20, датчик влажности и температуры DHT11(DHT22), датчик атмосферного давления BMP180(BMP280). Для создания модели «умной» дачи будем использовать климатические датчики DHT22 и BMP180. Выбор обусловлен лучшими параметрами по точности и стоимости, в сравнении с другими датчиками.

Для мониторинга состояния почвы был выбран емкостной датчик влажности (Capactive soil moisture sensor v1.2). Датчик был выбран по критерию устойчивости контактов к коррозии.

Для наблюдения за домом были выбраны датчик задымления MQ-2 и pir-датчик присутствия HC-SR501.

Все характеристики датчиков, описываемых ниже, взяты с официальной документации [4][5][6].

* + 1. Датчики температуры и влажности

DHT22 (Digital Humidity Temperature) (см. рисунок 2.1) – недорогой цифровой датчик температуры и влажности. Он использует емкостной датчик влажности и терморезистор для измерения температуры окружающего воздуха, данные выдает в цифровой форме по шине типа 1-wire. В использовании он довольно прост, но требует точного определения длительности временных сигналов, чтобы декодировать данные. Единственный недостаток — это возможность получения данных не чаще 1 раза в две секунды.

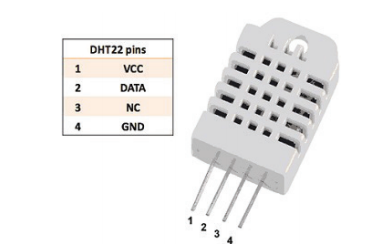


Рисунок 2.1 – Датчик влажности и температуры DHT22

Для преобразования данных внутри датчика используется 8-битный микроконтроллер. В процессе производства датчики калибруются, и калибровочная константа записывается вместе с программой в память микроконтроллера. Однопроводный последовательный интерфейс дает возможность быстрой интеграции в устройство [Петин, создание умного дома].

DHT22 определяет влажность в диапазоне от 0 до 100 % с точностью 2-5 % и температуру в диапазоне от -40 до 125 °С с точностью 0.5 °С. Частота опроса датчика не более одного раза в две секунды.

Между выводом питания и выводом данных необходимо разместить подтягивающий резистор. Рекомендуемый номинал 10 кОм, если расстояние от датчика к плате Arduino небольшое. Для расстояния больше 20 метров рекомендуется резистор номиналом 5,1 кОм.  Так же необходим сглаживающий конденсатор между выводом питания и землей. Но для удобства подключения предусмотрен модульный вариант датчика, в который уже включен резистор и конденсатор (см. рисунок 2.2).



Рисунок 2.2 – Модуль датчика DHT11

DHT22 является улучшенным аналогом DHT11 с интервалами определения влажности в диапазоне от 0 до 100 % с точностью 2-5 % и температуры в диапазоне от -40 до 125 °С с точностью 0.5 °С с частотой опроса не более раза в 2 секунды. Имеется безмодульный (см. рисунок 2.1) и модульный (см. рисунок 2.2) варианты.

Для проекта были выбраны модули датчиков, поэтому нет необходимости использовать подтягивающий резистор и сглаживающий конденсатор. Модули подключаются по однопроводным последовательным интерфейсом непосредственно к пину приема данных (в проекте используется пин 3 для DHT11 и пин 5 для DHT22). Схема подключения модуля представлена на рисунке 2.3.

Описание передачи данных по однопроводному интерфейсу:

Микроконтроллер (МК) инициирует передачу данных путем отправки сигнала «Старт». Для этой цели данный вывод МК должен быть сконфигурирован на выход. МК сначала переводит линию в состояние низкого уровня, по крайней мере на 18 мсек, а затем переводит ее в высокое состояние на 20-40 мксек, прежде чем МК освобождает линию. Далее, датчик реагирует на сигнал «Старт» от МК, отправив низкий уровень сигнала длительностью 80 мксек, а затем высокий уровень с такой же продолжительностью. После обнаружения сигнала отклика от датчика, МК должен быть готов к приему данных от датчика. Датчик посылает 40 бит (5 байт) данных, непрерывно в линию передачи данных. Следует отметить, что во время передачи байта, датчик посылает старший значащий бит первым [5] [6].

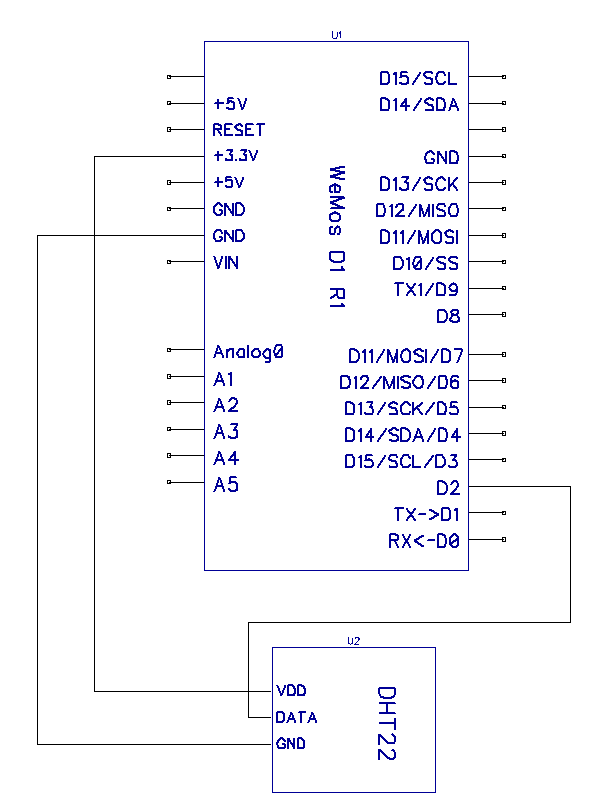


Рисунок 2.3– Схема подключения DHT22 к WeMos D1 R1

Датчик BMP180 (см. рисунок 2.4) (Digital Pressure Sensor) — является недорогим и простым в использовании сенсорным датчиком, позволяющий измерить атмосферное давления и температуру окружающей среды.

Выпускается только в модульном варианте. На модуле расположен сам сенсорный датчик BMP180 фирмы Bosch. Так как датчик BMP 180, работает от 3.3В (а почти все платы Arduino работают на 5В), на плате предусмотрен стабилизатор напряжения XC6206P332MR в корпусе SOT-23, который выдает на выходе напряжение в 3.3В, рядом установлена обвязка стабилизатора, состоящая из двух керамических конденсаторов на 1 мкФ. Подключение осуществляется по интерфейсу I2C, линии SCL и SDA выведены на группу контактов на другой стороне модуля, туда же выведено и питание. Последние два резистора на 4.7 кОм, необходимы подтяжки линии SCL и SDA к питанию.

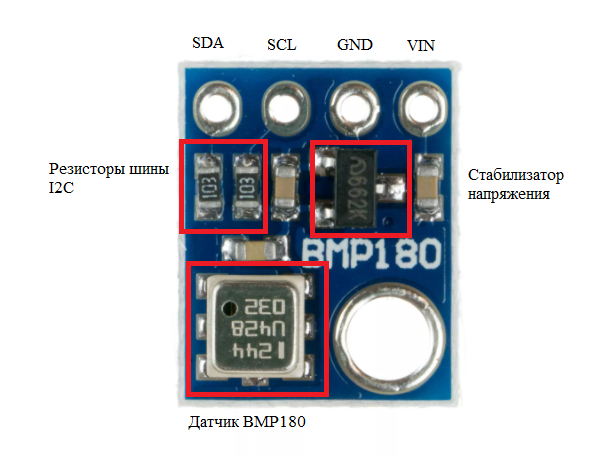


Рисунок 2.4 – Модуль датчика BMP180 с распиновкой

Характеристики датчика:

- Рабочая температура -40 … 80 °C предельно допустимые значения

- Рабочее давление 0 … 10 МПа предельно допустимые значения

- Диапазон давления: 300 … 1100 гПа разрешение 0,06 гПа точность ±0,12 гПа (на пределах ±1 гПа)

- Диапазон температуры: 0 … 65 °C разрешение 0,1°C точность ±0.5 °C (на пределах ±2 °C)

- Время преобразований 3 … 51 мс зависит от режима точности

Модуль датчика BMP180 подключается к Arduino по последовательному интерфейсу I2C. Для подключения датчика используется следующая распиновка: SDA (шина данных) подключается к пину A4 (пин данных для шины I2C), SCL (шина тактирования) подключается к пину A5 (пин тактирования шины I2C). Подключение датчика к Arduino представлено на рисунке 2.5.

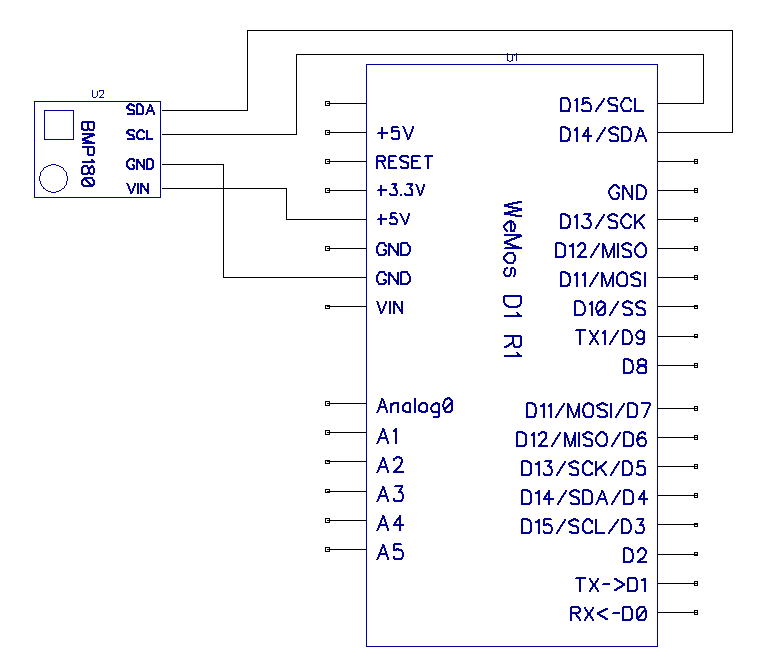


Рисунок 2.5 – Подключение BMP180 к WeMos D1 R1

Передача данных между датчиком и МК устроена более сложно, чем в DHT, так как на датчике организована EEPROM и RAM память.

Доступ к данным регистров датчика BMP180:

Каждый регистр датчика хранит 1 байт данных. Так как модуль использует интерфейс передачи данных I2C, то и доступ к данным охарактеризован им.

Запись данных в регистры:

Отправляем 1й байт (адрес датчика 0x77 и бит «R/W»=«0»); отправляем 2ой байт (адрес нужного нам регистра); отправляем 3й байт (данные для записи); после каждого отправленного байта, получаем ответ от датчика в виде одного бита «ACK». На рисунке 2.6 представлен пример записи в регистр 0xF4 значения 0xB4.

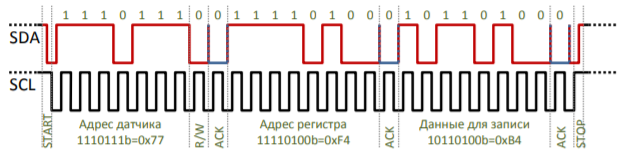


Рисунок 2.6 – Пример записи данных в регистр

Чтение данных из регистров:

Отправляем 1й байт, (адрес датчика 0x77 и бит «R/W» = «0»); отправляем 2ой байт (адрес нужного нам регистра); отправляем сигнал «RESTART»; отправляем 3й байт, (адрес датчика 0x77 и бит «R/W»=«1»); датчик ответит одним байтом данных из указанного регистра; если подать сигнал «ACK», то датчик передаст байт данных следующего регистра и т.д. пока мы не передадим сигнал «NACK». Если на шине только один ведущий, то после передачи двух первых байт (адреса датчика с битом «R/W» = «0» и адреса регистра) допустимо завершить пакет подачей сигнала «STOP» и начать новый пакет сигналом «START» передать адрес датчика с битом «R/W» после чего начать принимать или передавать данные. Такой вариант передачи данных позволяет использовать библиотеки, в которых нет сигнала «RESTART». На рисунке 2.7 представлен пример чтения байта из регистра 0xF6 (датчик ответил значением 0x5C).

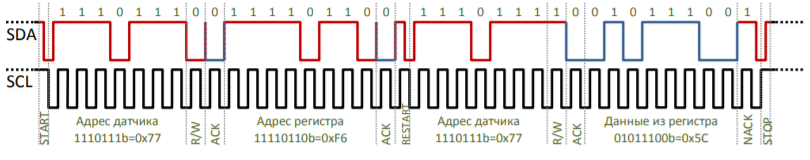


Рисунок 2.7 – Пример чтения данных из регистра

* + 1. Датчик задымления

Зачастую, на дачных участках могут быть установлены газовые плиты. К сожалению, иногда они могут выходить из строя и пропускать через конфорки газ, который заполняет помещение, или же сам человек по неосторожности может вызвать воспламенение окружающих вещей с помощью газовой плиты, например, оставив рядом с ней легковоспламеняющийся материал. Для предупреждения таких случаев в модели «умной» дачи будем использовать модульный датчик широкого спектра газов MQ-2.

Датчик MQ-2, представленный на рисунке 2.8, способен определять в воздухе концентрацию сжиженного нефтяного газа, дыма (взвешенных частиц, образуемых в результате горения), пропана, водорода, метана и угарного газа.



Рисунок 2.8 – датчик MQ-2

Принцип работы датчика построен на химическом резисторе (датчик типа металл-оксид-полупроводник, MOS). Обнаружение газов основано на изменении сопротивления чувствительного материала. Когда газ вступает в реакцию с материалом, сопротивление материала изменяется, соответственно, изменяется выходное напряжение датчика на делителе напряжения.

Для лучшего протекания реакции между газами и химическим резистором в датчике предусмотрен нагревательный элемент, поэтому во время работы датчик будет горячим. Для обеспечения более точной работы сенсора необходимо во время первого запуска сенсор необходимо прогревать в течение суток. После этой операции последующее включение и прогрев датчика будет занимать не более двух минут.

Разобравшись со строением датчика можно более точно описать принцип его работы. Когда частицы полупроводника нагреваются на воздухе до высокой температуры, на поверхности адсорбируется кислород. В чистом воздухе донорные электроны диоксида олова (полупроводника) притягиваются к кислороду, который адсорбируется на поверхности чувствительного материала. Это предотвращает протекание электрического тока. В присутствии восстановительных газов поверхностная плотность адсорбированного кислорода уменьшается, так как он реагирует с восстановительными газами. Из-за чего электроны высвобождаются в диоксид олова, что позволяет току свободно течь через датчик.

Датчик газа MQ-2 работает при постоянном напряжении 5 В и потребляет около 800 мВт. Он способен обнаруживать концентрации газов от 200 до 10000 ppm (миллионных долей). Миллионная доля (сокращенно ppm) – это соотношение одного газа к другому. Например, 1000 ppm CO означает, что, если бы вы могли сосчитать миллион молекул газа, то 1000 из них были бы моноокисью углерода, а 999 000 молекул – какими-то другими газами.

Характеристики датчика:

— Напряжение питания: 5В

— Потребляемый ток (ток нагревателя): 180мА

— Диапазон чувствительности 300-10000 ppm

— Газ, для которого нормируется датчик: изобутан, 1000ppm

— Время отклика: менее 10 с

— Рабочая температура: от -10 до +50 °C

— Рабочая влажность воздуха: не более 95% RH

— Интерфейс: аналоговый и цифровой

Сам датчик газа скрыт внутри двух слоев тонкой сетки из нержавеющей стали. Это необходимо для того, чтобы в случае выхода из строя нагревательного элемента внутри датчика не произошел взрыв, при обнаружении легковоспламеняющихся газов. Так же сквозь сетку могут проходить только взвешенные частицы, что предотвращает попадание негазообразных веществ на датчик.

Для подключения датчика к микроконтроллеру используется модульный вариант датчика (Рисунок 2.9). Такой вариант подключения датчика позволяет довольно простым способом организовать обмен данными с микроконтроллером. Также модульный датчик имеет два выхода: цифровой и аналоговый. Напряжение на аналоговом выходе изменяется пропорционально концентрации газов, попадающих на датчик. Соответственно, чем выше концентрация газов, тем выше выходное напряжение датчика.



Рисунок 2.9 – модульный вариант датчика MQ-2

Аналоговый сигнал от датчика поступает на высокоточный компаратор LM393, расположенный в нижней части модуля, для оцифровки сигнала. Рядом с компаратором расположен потенциометр, регулирующий чувствительность датчика (Рисунок 2.10). С помощью этого потенциометра производится калибровка датчика. Что бы произвести калибровку необходимо держать датчик вблизи с источником газа, который необходимо будет обнаруживать, и поворачивать ручку потенциометра, пока на модуле не начнет светиться красный светодиод, показывающий обнаружение газа.

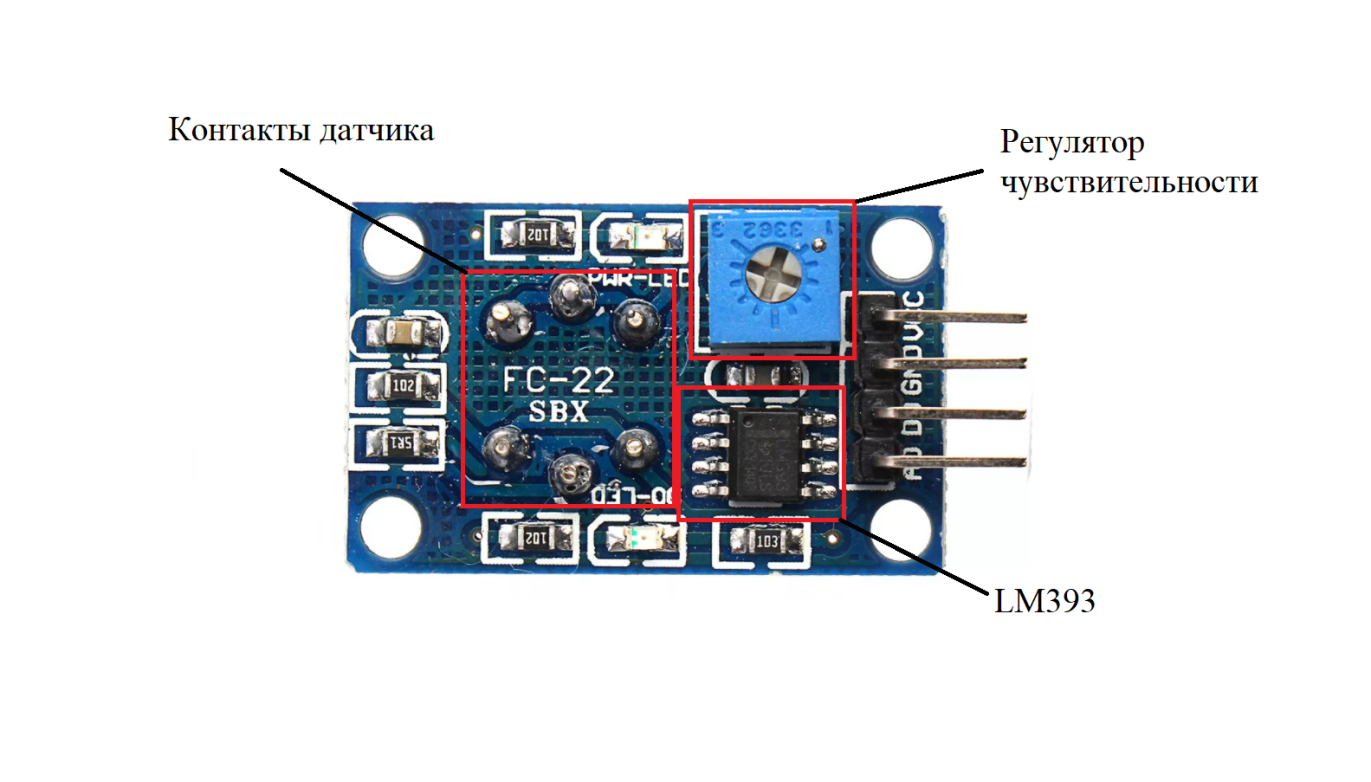


Рисунок 2.10 – модульный вариант датчика MQ-2 (вид снизу)

Рассмотрим контакты модуля, используемые для подключения к микроконтроллеру (Рисунок 2.11):

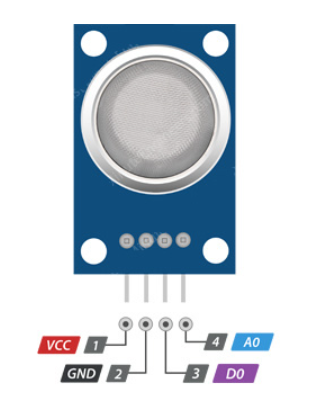


Рисунок 2.11 – контакты модуля датчика MQ-2

– VCC обеспечивает питание для модуля. Ранее было рассмотрено, что датчик питается от 5В, соответственно контакт VCC подключается к контактам питания периферии микроконтроллера 5В.

– GND – вывод земли, должен быть подключен к выводу GND на микроконтроллере.

– D0 обеспечивает цифровое представление о наличии горючих газов. Подключается к любому цифровому контакту микроконтроллера.

– A0 обеспечивает аналоговое выходное напряжение, пропорциональное концентрации газа. Подключается к любому аналоговому контакту микроконтроллера.

* + 1. Датчик движения

В продолжение темы безопасности участка, стоит задуматься о мониторинге присутствия посторонних на участке. Для заданных целей будем использовать PIR-сенсор с датчиком HC-SR501 (Рисунок 2.12).



Рисунок 2.12 – датчик HC-SR501 с линзой Френеля

Принцип действия PIR-датчика построен на сравнении инфракрасного излучения в помещении с «нормальным» инфракрасным излучением. При включении датчик настраивается на «нормальное» излучение в пределах зоны обнаружения, затем он ищет изменения излучения в этой зоне, например, если человек переместится в пределах контролируемой зоны. Для определения излучения используется пироэлектрический датчик. Пироэлектричество — это свойство генерировать определенное электрическое поле при облучении материала инфракрасными лучами. Особенностью датчика является то, что датчик не излучает сигнал, а только принимает. Это устройство генерирует электрический ток в ответ на прием инфракрасного излучения: при изменении излучения на выходе датчика формируется напряжение.

Для повышения чувствительности и эффективности датчика HC-SR501 используется метод фокусировки инфракрасного излучения на устройство, достигается, это с помощью «Линзы Френеля». Линза выполнен из пластика и 25 выполнена в виде купола и фактически состоит из нескольких небольших линз Френеля. Хоть пластик и полупрозрачен для человека, но на самом деле полностью прозрачен для инфракрасного света, поэтому он также служит в качестве фильтра.

На плате датчика присутствуют два потенциометра, предназначенные для настройки чувствительности датчика (левый потенциометр на рисунке 2.13) и времени задержки сигнала при обнаружении (правый потенциометр на рисунке 2.13). Настройка производится в пределах от 3 до 7 метров и от 3 секунд до 5 минут соответственно.

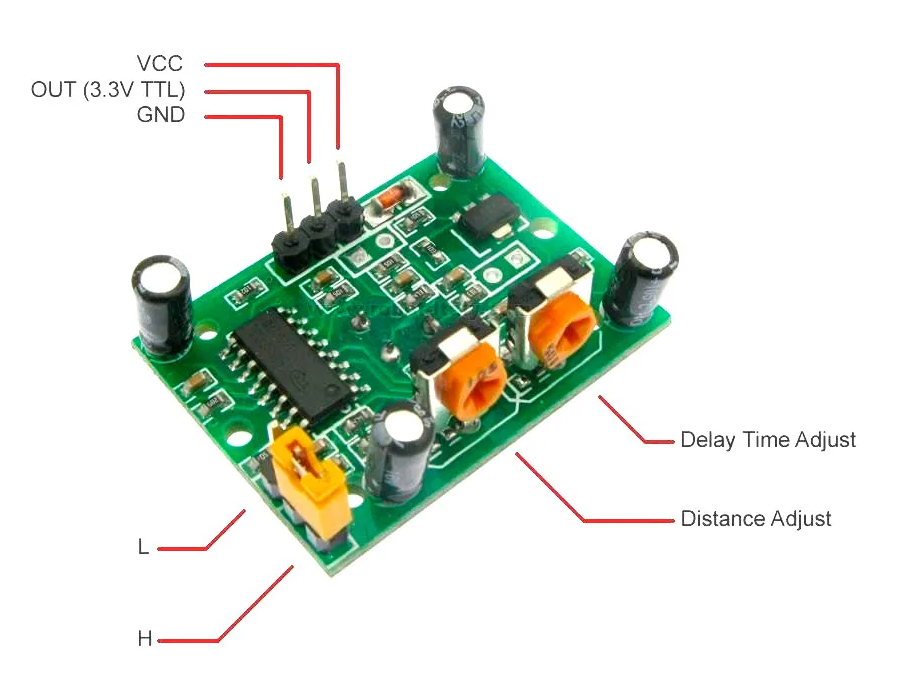


Рисунок 2.13 – обратная сторона модуля HC-SR501

На плате предусмотрена перемычка, с помощью которой выбирается режим работы датчика:

H – настройка Hold. В этом положении HC-SR501 будет продолжать выдавать высокий уровень, пока он обнаруживает движение.

L — это параметр прерывания. В этом положении выход будет оставаться в высоком уровне в течение периода, который установлен потенциометром, отвечающим за время.

Технические характеристики:

Напряжение питания: 4.8В - 20В  
Статический ток: 50 мА  
Уровня выходного сигнала: 3.3 В / 0 В  
Время задержки: 0.5 — 200с (регулируемая)  
Время блокировки: 2.5 с  
Угол работы: < 100  
Рабочая температура: -15С … + 70C  
Определение объектов: 23 мм  
Габариты: 33мм x 25мм x 24мм

Контакты модуля:

VCC — положительное напряжение постоянного тока от 4,5 до 20 В постоянного тока.

OUTPUT — логический выход на 3,3 вольта. Высокий уровень соответствует обнаружению в поле контролируемой зоны.

GND — земля.

* + 1. Датчик влажности почвы

Для мониторинга состояния почвы, а именно, влажности, лучшим выбором на рынке будет емкостной датчик влажности почвы (рисунок 2.14).

Данный датчик измеряет уровень влажности почвы посредством емкостного измерения, а не резистивного, как другие датчики. Это позволило увеличить срок службы датчика, так как он не подвержен коррозии. Так же, модуль включает в себя встроенный стабилизатор напряжения, с помощью которого обеспечивается диапазон работы от 3.3 В до 5.5 В, что позволяет подключать его как к Arduino UNO, так и к NodeMCU.

Выходное напряжение ёмкостного датчика почвы составляет от 1.2 В до 3.0 В.



Рисунок 2.14 – емкостной датчик влажности почвы

Емкостный датчик выполнен в виде штыря, которым погружается в грунт на расстояние до 80 мм. На штыре в виде дорожек расположены два электрода.

Внутри емкостного датчика находится RC-генератор на таймере 555, частота которого зависит от емкости между двумя электродами, которые выполняю роль конденсатора. Изменение влажности грунта сказывается на его диэлектрических свойствах и меняет ёмкость, что приводит к повышению или понижению выходного сигнала датчика. Итоговое напряжение пропорционально степени влажности почвы.

Емкостной датчик почвы v1.2 имеет один разъем (PH2.0-3P) для подключения:  
GND —  заземляющий вывод питания.  
VCC — вывод питания 3.3 В — 5 В.  
AUOT — аналоговый выход до 3В.

## Актуаторы

Исполнительное устройство (исполнительный элемент, актуатор)  — функциональный элемент [системы автоматического управления](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), который воздействует на объект управления, изменяя поток энергии или материалов, которые поступают на объект. Большинство исполнительных устройств имеет механический или электрический выход.

Состоит из двух функциональных блоков: исполнительного устройства и регулирующего органа, например [регулирующего клапана](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D0%BF%D0%B0%D0%BD), и может оснащаться дополнительными блоками.

* + 1. Реле-модули

Реле позволяет подключить устройства, работающие в режимах с относительно большими токами или напряжениями, которые невозможно запитать от платы напрямую, например, лампы накаливания или насосы.

Реле модули (рисунок 2.15) повсеместно встречаются в проектах умного дома, так как они очень просты в использовании и подключении, и выполняют хоть и простой, но довольно полезный функционал, а именно, управление большими нагрузками.

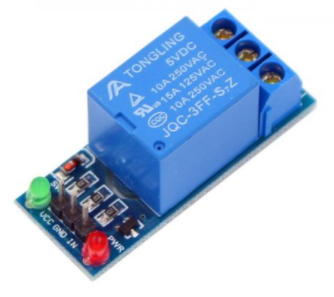


Рисунок 2.15 – электромагнитный реле-модуль

Электромагнитное реле – это электрическое устройство, которое механическим путем замыкает или размыкает цепь нагрузки при помощи магнита. состоит из электромагнита, подвижного якоря и переключателя.

Работает реле благодаря электромагнитной силе, возникающей в сердечники при подаче тока по виткам катушки. В исходном состоянии пружина удерживает якорь (рисунок 2.16, левая схема). Когда подается управляющий сигнал, магнит начинает притягивать якорь и замыкать либо размыкать цепь (рисунок 2.16, правая схема). При отключении напряжения якорь возвращается в начальное положение. Источниками управляющего напряжения могут быть датчики (давления, температуры и прочие), электрические микросхемы и прочие устройства, которые подают малый ток или малое напряжение.

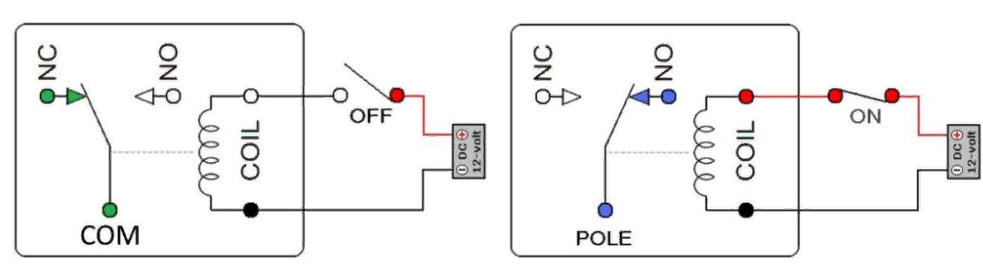


Рисунок 2.16 – принцип работы электромагнитного реле

На плате реле-модуля расположены два индикаторных светодиода:

Красный – на реле подается напряжение.

Зеленый – происходит замыкание реле.

Контакты реле, подключаемые к плате:

VCC – питание 5 В.

GND – земля.

IN – контакт управления. Низкий логический уровень – 0 В, высокий логический уровень – 5 В.

Контакты реле, подключаемые к управляемому устройству:

NO – нормально замкнутый контакт.

COM – общий контакт.

NC – нормально разомкнутый контакт.

# Организация обмена данными

## ESP-MESH

ESP-MESH – это сетевой протокол, построенный на основе протокола Wi-Fi. ESP-MESH позволяет нескольким устройствам (называемым узлами), распределенным по большой области в пространстве, соединяться в рамках одной WLAN (беспроводной локальной сети).

ESP-MESH отличается от традиционных инфраструктурных сетей Wi-Fi тем, что узлам не требуется подключаться к центральному узлу. Вместо этого, узлам разрешается соединяться с соседними узлами. Узлы несут взаимную ответственность за ретрансляцию передач друг друга. Это позволяет сети ESP-MESH иметь гораздо большую зону покрытия, поскольку узлы все еще могут обеспечивать взаимосвязь без необходимости находиться в зоне действия центрального узла (рисунок 3.1). Аналогичным образом, ESP-WIFI-MESH также менее подвержен перегрузке, поскольку количество узлов, разрешенных в сети, больше не ограничено одним центральным узлом.

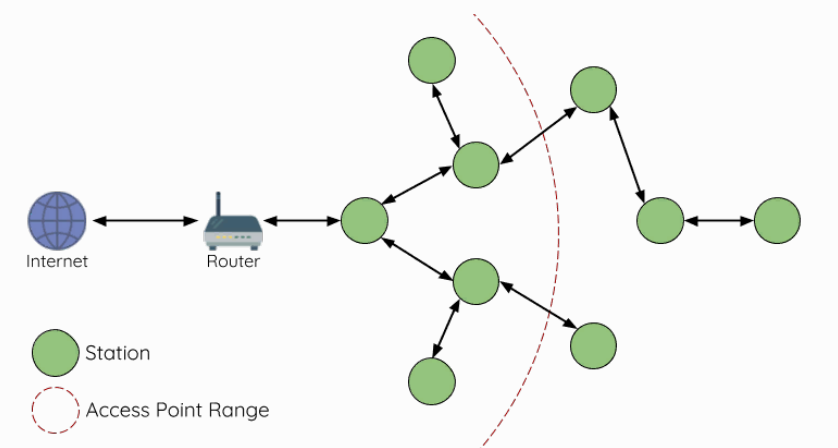


Рисунок 3.1 – строение MESH-сети

ESP-MESH построен поверх протокола Wi-Fi инфраструктуры и может рассматриваться как сетевой протокол, объединяющий множество отдельных сетей Wi-Fi в единую беспроводную сеть. В Wi-Fi станции ограничены одним соединением с точкой доступа (восходящее соединение) в любое время, в то время как точка доступа может быть одновременно подключена к нескольким станциям (нисходящие соединения). Однако ESP-WIFI-MESH позволяет узлам одновременно действовать как станция и точка доступа. Поэтому узел в ESP-WIFI-MESH может иметь несколько нисходящих соединений, используя свой интерфейс SoftAP, одновременно имея одно восходящее соединение с использованием интерфейса станции. Это, естественно, приводит к топологии древовидной сети с иерархией "родитель-потомок", состоящей из нескольких уровней.

Дальность передачи данных с помощью ESP-MESH определяется параметрами платы esp8266. Для данной платы зона покрытия составляет до 400 метров на открытой местности.

Типы узлов в сети ESP-MASH [10]:

**Корневой узел:** Корневой узел является верхним узлом в сети и служит единственным интерфейсом между сетью ESP-WIFI-MESH и внешней IP-сетью. Корневой узел подключен к обычному маршрутизатору Wi-Fi и ретранслирует пакеты во внешнюю IP-сеть/из нее на узлы в сети ESP-MESH. В сети ESP-MESH может быть только один корневой узел, и восходящее соединение корневого узла может быть только с маршрутизатором. Ссылаясь на приведенную ниже диаграмму (рисунок 3.2), узел А является корневым узлом сети.

**Конечные Узлы:** Конечный узел-это узел, которому запрещено иметь какие-либо дочерние узлы (без нижестоящих соединений). Поэтому конечный узел может передавать или принимать только свои собственные пакеты, но не может пересылать пакеты других узлов. Если узел расположен на максимально допустимом уровне сети, он будет назначен конечным узлом. Это предотвращает формирование узлом каких-либо нижестоящих соединений, тем самым гарантируя, что сеть не добавит дополнительный уровень. Ссылаясь на приведенную ниже диаграмму (рисунок 3.2), узлы L/M/N расположены на максимально допустимом уровне сетей, поэтому они были назначены конечными узлами.

**Промежуточные родительские узлы:** Подключенные узлы, которые не являются ни корневым узлом, ни конечным узлом, являются промежуточными родительскими узлами. Промежуточный родительский узел должен иметь одно восходящее соединение (один родительский узел), но может иметь от нуля до нескольких нисходящих соединений (от нуля до нескольких дочерних узлов). Поэтому промежуточный родительский узел может передавать и принимать пакеты, а также пересылать пакеты, отправленные из его вышестоящих и нижестоящих соединений. Ссылаясь на приведенную выше диаграмму, узлы от B до J являются промежуточными родительскими узлами. Промежуточные родительские узлы без нисходящих соединений, такие как узлы E/F/G/I/J (рисунок 3.2), не эквивалентны конечным узлам, поскольку им все еще разрешено формировать нисходящие соединения в будущем.

**Незанятые узлы:** Узлы, которые еще не присоединились к сети, назначаются в качестве незанятых узлов. Бездействующие узлы попытаются сформировать восходящее соединение с промежуточным родительским узлом или попытаются стать корневым узлом при правильных обстоятельствах. Ссылаясь на приведенную ниже диаграмму (рисунок 3.2), узлы K и O являются простаивающими узлами.

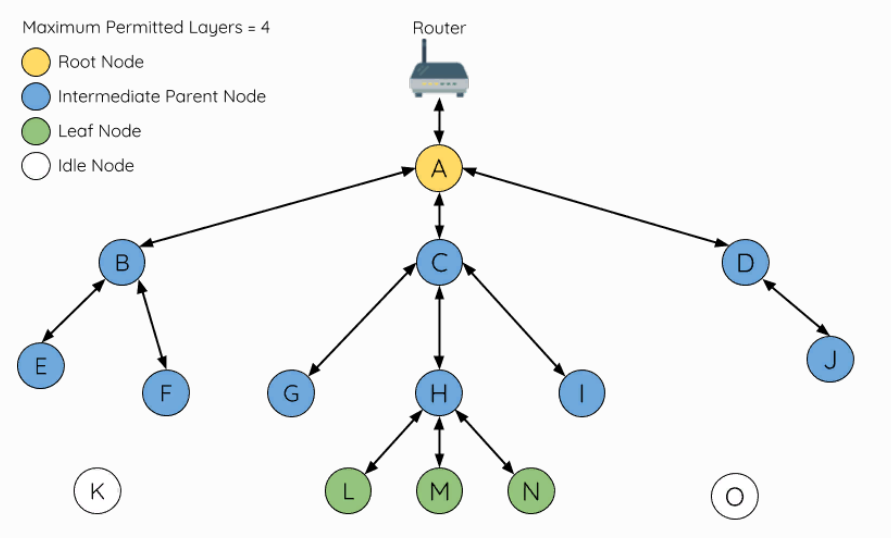


Рисунок 3.2 – типы узлов в ESP-MESH сети

## Протокол LoRaWAN

LoRaWAN – это технология беспроводной передачи данных, в которой используется метод радиомодуляции, который может осуществляться приемопередатчиками Semtech LoRa. Этот протокол модуляции обеспечивает передачу небольших данных на большие расстояния, высокую устойчивость к помехам при минимальном потреблении энергии.

LoRa использует нелицензированные частоты, доступные по всему миру. Вот самые широко используемые частоты: 868 МГц для Европы (Россия – 863 – 870MГц, 433 МГц); 915 МГц для Северной Америки; 433 МГц для Азии.

Типовая беспроводная сеть LoRaWAN представляет собой совокупность шлюзов (gateways), пересылающих сообщения между оконечными устройствами (end-devices) и центральным сервером (Network Server, NS), и характеризуется «звездной» топологией «star-of-stars» (рисунок 3.3).

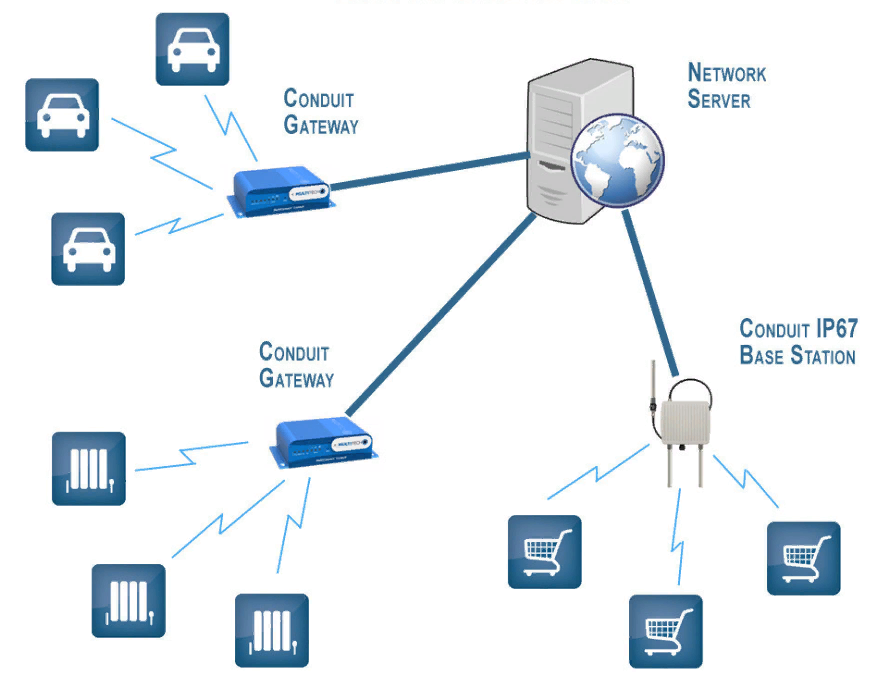


Рисунок 3.3 –топология сети LoRaWAN

Шлюзы называют также концентраторами (concentrators) и базовыми станциями (base stations). Оконечные устройства часто называют «motes».  
Связь между шлюзами и центральным сервером осуществляется через стандартные IP-соединения, а между шлюзами и оконечными устройствами — через беспроводные соединения, использующие широкополосную модуляцию LoRa или FSK. Модуляция LoRa была разработана компанией Semtech и предназначена для низкоскоростной беспроводной передачи данных на расстояния до нескольких километров в безлицензионных диапазонах частот (Европа — 433 и 868 МГц).

Связь между шлюзами и оконечными устройствами является двусторонней, но предполагается, что основной объем данных передается от оконечных устройств к шлюзам. Технология LoRa обеспечивает скорость передачи в беспроводном канале от 0.3 до 50 кбит/с. Для разделения каналов используется как набор частотных каналов, так и скоростей передачи (data rates).

Для оптимизации работы системы используется адаптивное изменение скорости передачи — ADR (adaptive data rate). Сетевой сервер оценивает качество сигнала, принимаемого от оконечного устройства, и может управлять как скоростью передачи, так и мощностью передатчика этого устройства.

Оконечное устройство может передавать данные на любом доступном канале и любой скорости передачи, учитывая следующее: каждый раз при передаче сообщения частотный канал выбирается оконечным устройством случайным образом из списка доступных каналов; перед началом передачи оконечное устройство должно убедиться в том, что канал свободен (Listen Before Talk, LBT), канал считается свободным, если измеренное мгновенное значение RSSI меньше, чем RSSI\_FREE\_TH, если канал занят, то устройство переходит на другой канал и повторяет процедуру LBT; оконечное устройство должно принимать во внимание ограничения местных регулирующих органов относительно процента времени, в течение которого устройство может занимать частотный канал [11].

Основные преимущества беспроводных сетей LoRaWAN обусловлены использованием широкополосной модуляции LoRa и безлицензионных диапазонов частот. Сети LoRaWAN: совместимы с существующими сетями/технологиями беспроводной передачи данных; обладают высокой помехоустойчивостью; способны обслуживать десятки и сотни тысяч устройств; обеспечивают большую зону охвата и малое энергопотребление оконечных устройств; дальность передачи составляет до 15 километров на открытой местности и до 5 километров в городе.

## ESP-NOW

ESP-NOW – это упрощенный протокол беспроводной связи от Espressif для обмена небольшими пакетами данных (до 250 байт). Он позволяет нескольким устройствам взаимодействовать друг с другом без подключения к Wi-Fi.

Протокол ESP-NOW образует двустороннюю передачу данных между двумя элементами сети (платами), поэтому топология сети очень простая: каждый из двух элементов сети выступает в роли приемника и передатчика одновременно. Так же возможны образования сетей «one-slave-multi-master» (один приемник, много передатчиков) или «one-master-multi-slave» (один передатчик, много приемников) (рисунок 3.4).

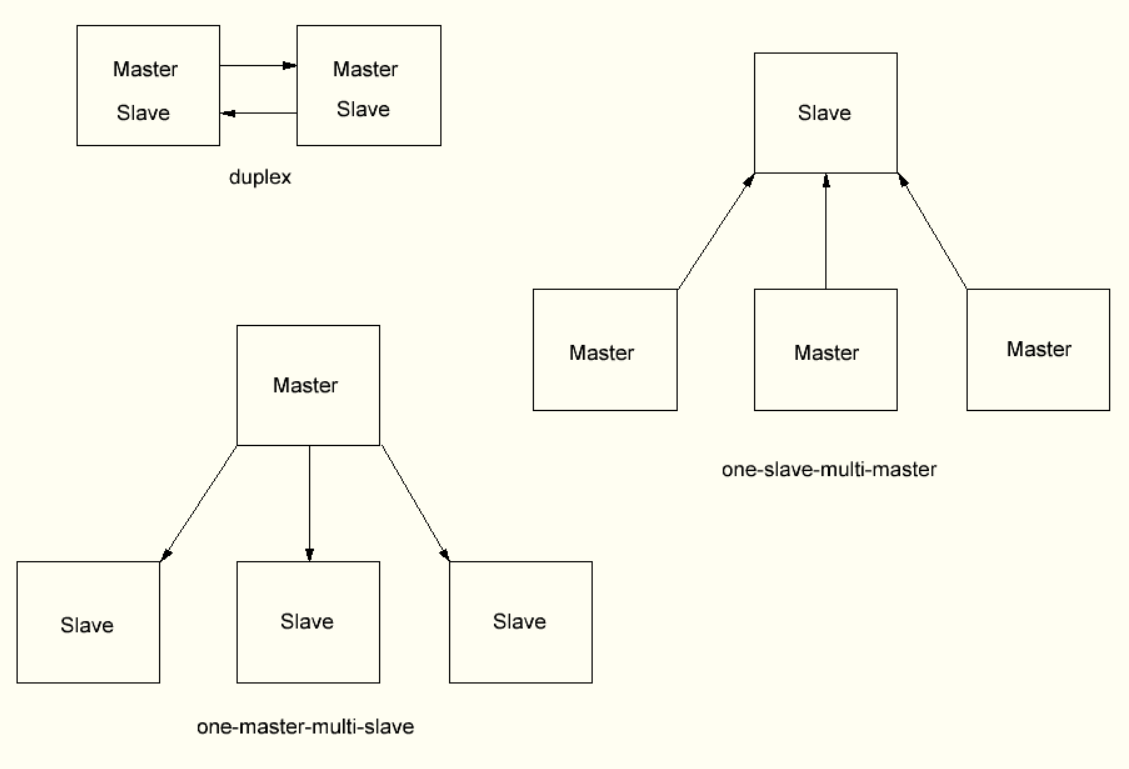


Рисунок 3.4 – виды сетей ESP-NOW

ESP-NOW поддерживает следующие функции:

1. Зашифрованная и незашифрованная связь между сопряженными парами устройств.
2. Смешанные зашифрованная и незашифрованная связь между сопряженными устройствами.
3. Передача до 250 байт полезной информации.
4. Настройка функции обратного вызова для информирования прикладного уровня, в частности, об успешности или сбое передачи.

На нижнем уровне протокола ESP-NOW поддерживается связанный список, содержащий информацию о локальном устройстве и о сопряженном устройстве, в том числе MAC-адреса и ключи. ESP-NOW также хранит часто используемые данные для прикладного уровня, чтобы избежать накладных расходов на повторную обработку связанного списка. Информация об устройствах используется для отправки и получения данных и включает в себя:

Информацию о локальном устройстве (PMK: 16 байт — основной мастер-ключ, который используется для шифрования ключа на присоединенном устройстве (KOK в API) ESP\_NOW поддерживает PMK по умолчанию, поэтому настройка не требуется. Если необходимо, можно убедиться, что значение PMK совпадает с локальным устройством. Режим: 1 байт — режим локального устройства, определяющий передающий WiFi интерфейс (SoftAP или STA) ESP-NOW. Режим сопряженного устройства не влияет на какую-либо функцию, а только сохраняет информацию о режиме для прикладного уровня. В режиме STA WiFi применим только Station и SoftAP WiFi — только SoftAP.);

Информацию о сопряженном устройстве в паре (LMK: 16 байт — локальный мастер-ключ, который используется для шифрования ключа полезной информации во время связи в данной паре; MAC-адрес: 6 байт — адрес сопряженного устройства, совпадает с адресом отправителя. Например, если пакет отправляется со Station, MAC-адрес должен совпадать с адресом Station; Режим: 1 байт — режим локального устройства определяющий передающий интерфейс (SoftAP или STA) ESP-NOW; Канал: 1 байт — канал, через который обмениваются данными устройства, соединенные в пару. Может иметь значение 0…255. Канал не влияет ни на какую функцию, а только сохраняет информацию о канале для прикладного уровня. Значение определяется прикладным уровнем. Например, 0 означает, что канал не определен; 1 ~ 14 означает действительные каналы; всем остальным значениям могут быть назначены функции, которые определены прикладным уровнем) [12].

Формат пакета ESP-NOW [12]:

1. Заголовок MAC: 24 байта.
2. Категория: 1 байт, указывающий на категорию создателя пакета. Установлено значение (127).
3. ID организации: 3 байта, содержит уникальный идентификатор, который является первыми тремя байтами MAC-адреса, примененного Espressif. Установлено значение (0x18fe34)
4. Случайное значение: 4 байта, используется для защиты данных.
5. Данные создателя пакета: 7-255байт

Данные создателя пакета содержат следующие поля [12]:

1. ID: 1 байт, Установлено значение (221).
2. Длина: 1 байт, общая длина ID организации, типа, версии и пользовательских данных.
3. ID организации: 3 байта, содержит уникальный идентификатор, который является первыми тремя байтами MAC-адреса, примененного Espressif. Установлено значение (0x18fe34)
4. Тип: 1 байт, протокол ESP-NOW. Установлено значение (4)
5. Версия: 1 байт, текущая версия ESP-NOW. Установлено (1)
6. Содержимое: 0-250 байт пользовательские данные.
7. FCS: 4 байта, контрольная сумма

## Протокол ZigBee

Zigbee — спецификация [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) верхнего уровня — уровня приложений APS ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) application support sublayer) и сетевого уровня NWK, — использующих сервисы нижних уровней — уровня управления доступом к среде [MAC](https://ru.wikipedia.org/wiki/Media_Access_Control) и физического уровня [PHY](https://ru.wikipedia.org/wiki/PHY), регламентированных стандартом [IEEE 802.15.4](https://ru.wikipedia.org/wiki/IEEE_802.15.4-2006). Zigbee и IEEE 802.15.4 описывают [беспроводные персональные вычислительные сети](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8_(WPAN)) (WPAN). Спецификация Zigbee ориентирована на приложения, требующие гарантированной безопасной передачи данных при относительно небольших скоростях и возможности длительной работы сетевых устройств от автономных источников питания (батарей).

ZigBee создан на основании стандарта IEEE 802.15.4-2006 и состоит из высокоуровневых протоколов, которые работают с маленькими цифровыми трансиверами. Передача данных осуществляется по радиоканалу. Частота зависит от региона. Функционирование на частоте 2,4 ГГц не связано с расположением. Стандарт создавался с целью быть легче в эксплуатации и ниже в цене. Характерен низкий период ответа оборудования. Период активации занимает не более 15 миллисекунд [13]. Благодаря переходу в спящий режим значительно падает потребность в электроэнергии. Система позволяет работать с приложениями для:

* телекоммуникаций;
* игрушек;
* коммерческого строительства;
* повышения энергоэффективности;
* персонального домашнего ухода за больными;
* автоматизации процессов дома.

Внутри системы все ZigBee устройства делятся на несколько групп (рисунок 3.5):

1. Координаторы (ZC). Прибор запускает сеть и задает все команды для управления ее действиями. Также контроллер обеспечивает безопасность всех процессов.
2. Маршрутизаторы (ZR). Функционируют непрерывно и обеспечивают работу устройств, находящихся в режиме сна (до 32 штук). Также переносят данные и занимаются восстановлением гаджетов в случае большой загруженности или неисправности системы. Образуют соединение с координатором, другим маршрутизаторам, а также с дочерними периферийными приборами и оборудованием для передачи информации.
3. Конечные устройства (ZED). Выполняют получение и отправку пакетов данных. Подключаются к двум гаджетам, перечисленным выше. Не подключают дочерние приборы. Работают с сенсорами, контроллерами и механизмами, выполняющими команды. Для сохранения энергии часто работают в спящем режиме.

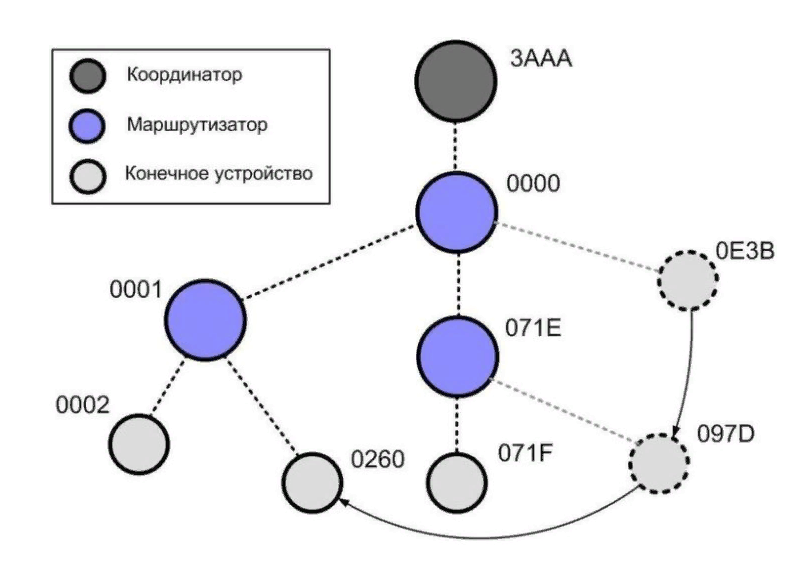


Рисунок 3.5 – топология ZigBee сети

На начальном этапе формирования сети используется PAN координатор. Он вычисляет радиоканал, в котором нет помех, и ждет запросы на подключение, которые отправляют разные устройства. Изначально подключать других участников может только координатор, а в дальнейшем и другие устройства для пересылки данных. После получения запроса происходит обмен сообщениями.

Бывает два варианта присоединения: MAC ассоциация, повторное сетевое присоединение. В первом случае устройство, желающее стать частью сети на MAC уровне, отправляет запрос  маячка. Когда маячки достигают цели, управляющее устройство сканирует сети и выбирает, какую лучше подключить в конкретной ситуации, высылает притязание о присоединении со значком «повторное присоединение». Получив ответ, управляющее устройство высылает сообщение с адресом. Такой способ присоединения не достаточно безопасен, поскольку все данные не защищены кодированием.

Повторное сетевое присоединение также часто используется при первичном обращении. В таком случае участник, желающий вступить в сеть, знает текущий ключ. Поэтому, такой способ присоединения намного безопаснее. Узнать ключ можно при настройке оборудования. При повторном подключении оборудование обменивается между собой пакетами запроса и ответа на присоединение.

Видно, то сеть ZigBee представляет собой MESH-сеть. Единственным отличием от mesh-сети является то, что сеть ZigBee является самоорганизующейся сетью, то есть узлы могут сами перенаправлять поток данных для более устойчивой работы сети. Так, например, при выходе из строя одного устройства-маршрутизатора сеть перестраивает свою структуру в обход вышедшего из строя устройства и передача данных продолжается. Поэтому ZigBee сети являются сетями с большой отказоустойчивостью.

# Моделирование системы с использованием протокола ESP-NOW

Разобравшись с работой выбранных модулей и способом обмена данных внутри сети, образованной несколькими платами ESP8266, можно рассмотреть аналитическую модель «умной дачи».

Для формирования модели будем использовать все вышеперечисленные датчики и актуаторы, две платы WeMos D1 R1, так же будем использовать помпу или насос как мощную нагрузку для реле.

## Схема подключения устройств и размещения на участке

Аналитическая модель предусматривает наличие доступной Wi-Fi сети на участке с доступом в Интернет, так как это будет являться единственным каналом передачи данных на смартфон. На рисунке 4.1 представлена схема расположения модулей на участке.

Модуль теплицы целесообразно поместить в пластиковый корпус с помещенным внутрь впитывающим материалом, так как повышенная влажность в теплице может вызвать коррозию контактов платы и преждевременный выход ее из строя. Чтобы не искажать результаты датчиков их стоит вынести за корпус, в котором будет помещена плата. Так же все элементы тепличного модуля помещены внутрь тепличного помещения для того, чтобы внешние погодные условия не вывели элементы из строя.

Модуль жилого помещения размещен непосредственно в помещении, рядом платой ESP8266 будет расположен датчик задымления MQ2. Датчик целесообразно устанавливать в комнате, где имеется большая вероятность возникновения задымления, например, на кухне. Датчик стоит расположить под потолком, так как угарный газ под действием тепла от источника задымления сначала поднимается наверх.

Датчик присутствия будет расположен так, чтобы линза Френеля охватывала большую часть двора. Так как датчик будет расположен на улице, то для него стоит предусмотреть отдельный навес, или расположить его под уже имеющимся навесом.

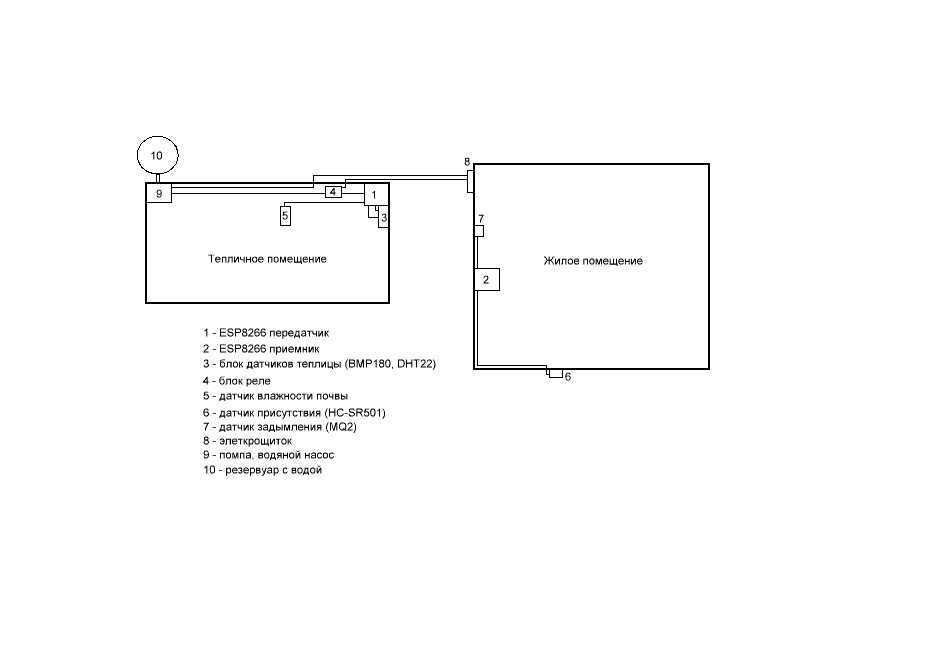


Рисунок 4.1 – план-схема расположения блоков на участке

Подключение элементов к плате осуществляется согласно описанию датчиков и актуаторов в пункте 2 данной работы (рисунки 4.2 и 4.3).

Питание насоса подается от электрощитка, расположенного на доме. Для предупреждения случаев короткого замыкания подключение реле в разрыв фазы насоса расположено в теплице, в корпусе, рассмотренном ранее.

Питание модулей осуществляется с помощью подзаряжаемых батарей (PowerBank). При дальнейшем развитии модели можно использовать солнечные батареи, расположенные на крыше жилого помещения и теплицы.

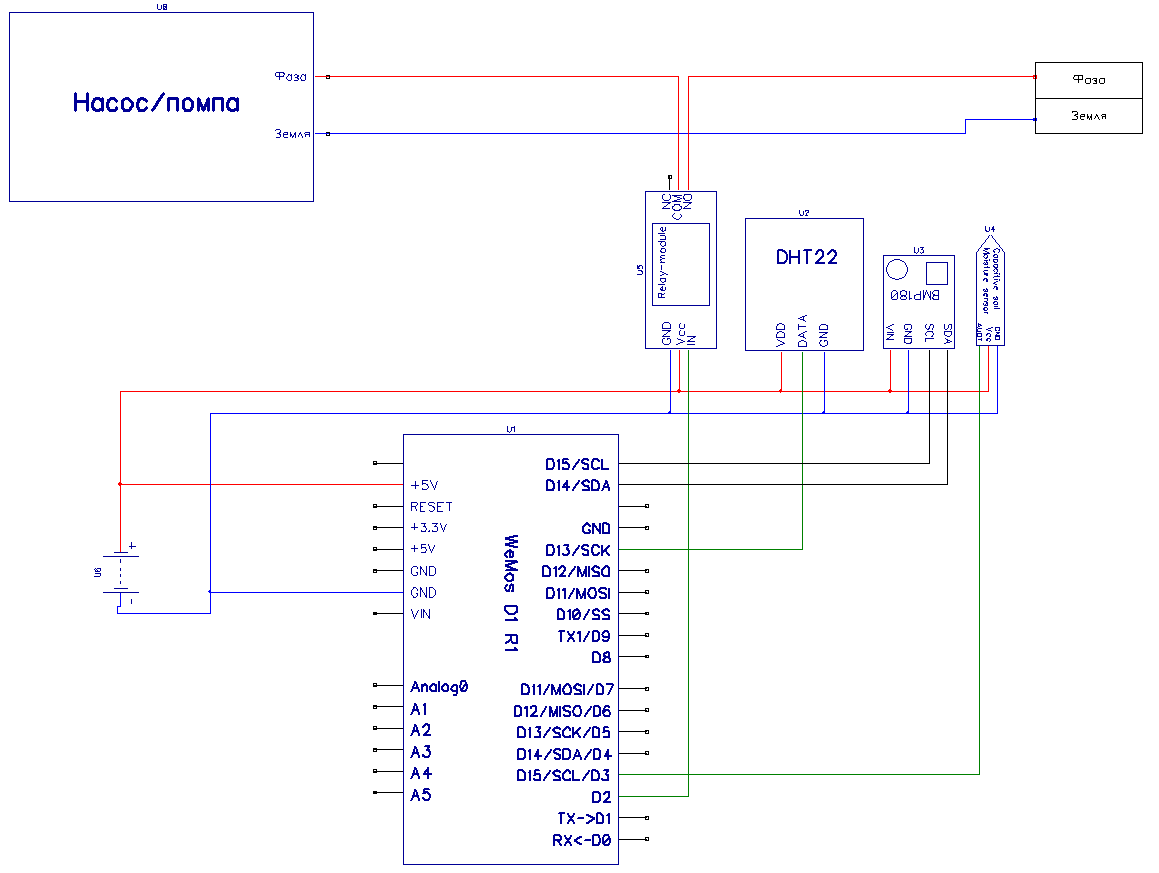


Рисунок 4.2 – схема подключения модулей в тепличном блоке

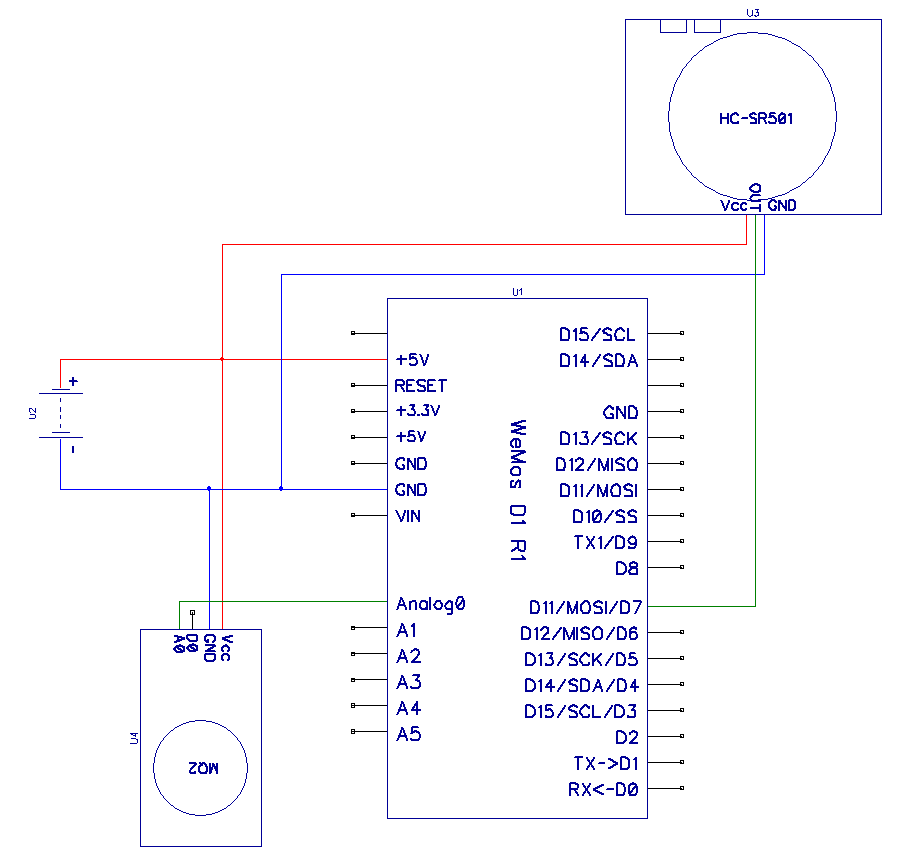


Рисунок 4.3 – схема подключения модулей в блоке жилого помещения

## Обмен данными в системе

В пункте 3 данной работы были рассмотрены три различных способа организации локальной сети на платах ESP8266. Для заданной аналитической модели наиболее подходящим способом организации данных между платами будет использование протокола ESP-NOW. Выбор обусловлен по следующим критериям: во-первых, в модели используется только две платы, что отбрасывает вариант использования ESP-MASH сети, которую используют для более сложных проектов, во-вторых, в модели считается, что весь участок попадает в зону покрытия Wi-Fi сети, поэтому использовать протокол LoRaWAN так же нецелесообразно, так как расстояния между платами составляет десятки метров.

В качестве сервера для хранения данных и отображения из на смартфоне рассмотрим сервис, направленный на проекты Интернета вещей, Blynk.

Данный сервис предоставляет пользователям удобный интерфейс и множество инструментов для создания визуального приложения для IoT устройств.

Передача данных осуществляется посредством API ключей, для идентификации устройства на сервере.

В результате схема обмена данных будет выглядеть как двухсторонняя передача данных между несколькими блоками модели (рисунок 4.4).

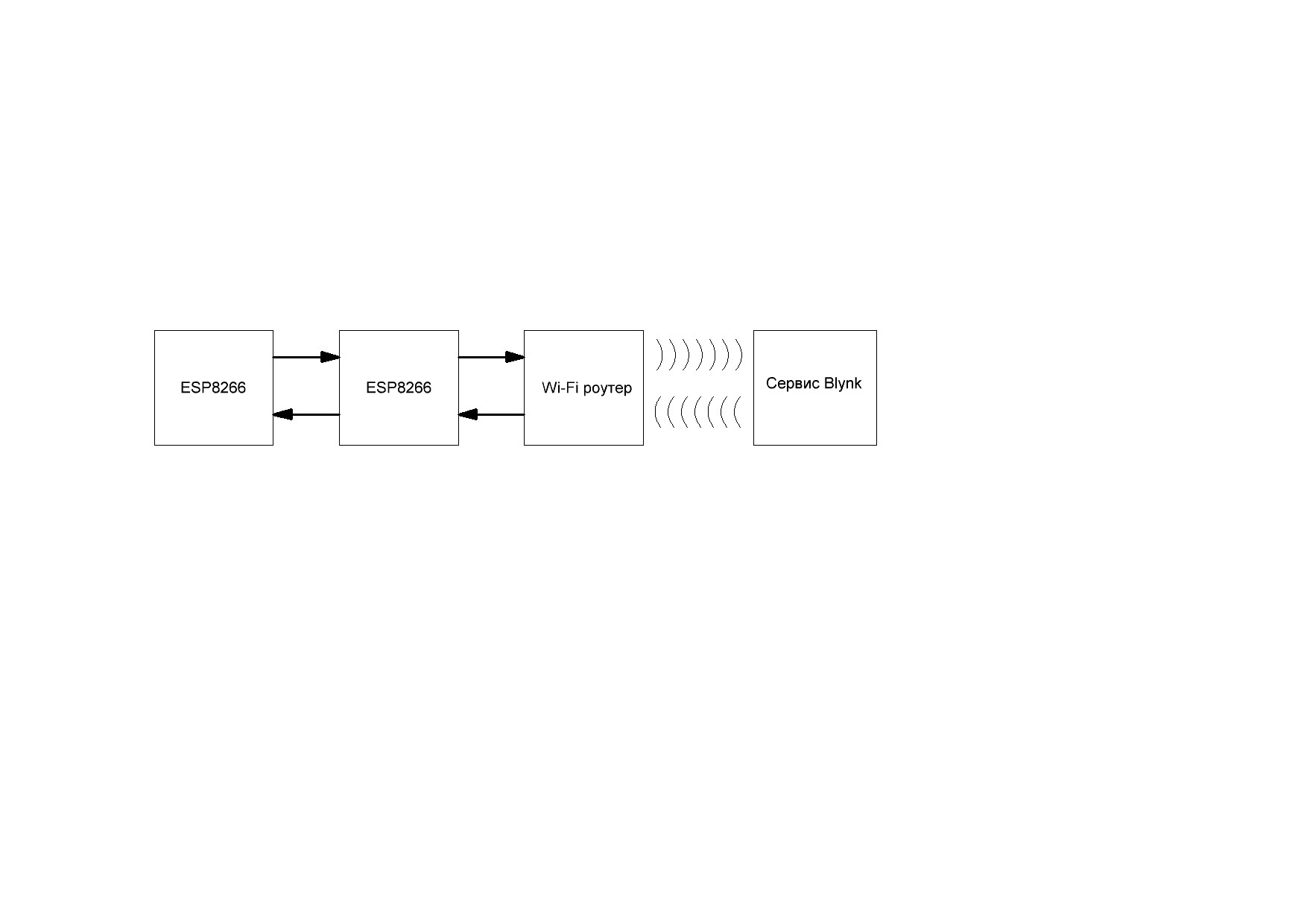


Рисунок 4.4 – схема передачи данных в модели

Сначала происходит обмен данными между платами по протоколу ESP-NOW (обмен данными представлен в приложениях А и Б в блоках 6), затем, данные с основной платы передаются на Wi-Fi роутер, откуда по протоколу TCP/IP данные отсылаются на сервис Blynk. Для реле-модуля в модели предусмотрен отдельный механизм передачи данных: после отправки данных на сервис происходит мониторинг виртуальной кнопки на сервисе, если ее значение изменилось, то Blynk отправляет на устройство текущее значение кнопки, далее это значение записывается в сообщение на плате, обмен данных которой происходит с роутером, и сообщение посылается на плату, установленную в теплице, и в зависимости от полученного значения реле замыкается или размыкается.

На сервисе Blynk с помощью заданных инструментов можно выставить нужные формы для принятия и отправления данных (рисунок 4.5). В настройках приложения выставляются виртуальные каналы с номерами потоков данных (V0, V5-V10). Для рассматриваемой модели используем каналы для выводов дробных значений (V5-V9), канал-индикатор (выводит 1 или 0 в зависимости от получаемого значения, V10), и канал-кнопку, для управления реле V0 (рисунок N).

Передача данных на каналы представлена в приложении Б в блоках 7 и 8.

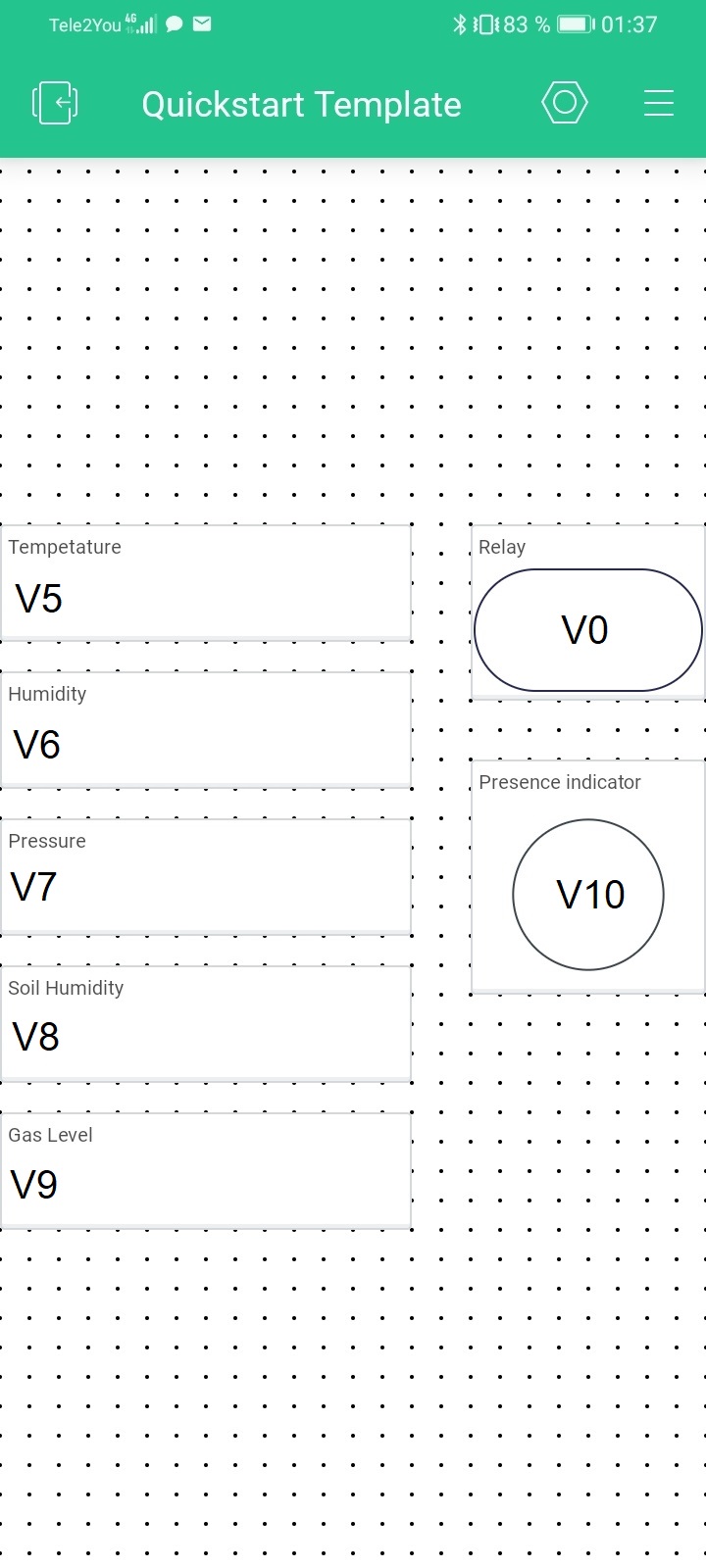


Рисунок 4.5 – интерфейс настроек приложения

После сохранения настроек приложение на сервисе Blynk будет выглядеть так, как представлено на рисунке 4.6.

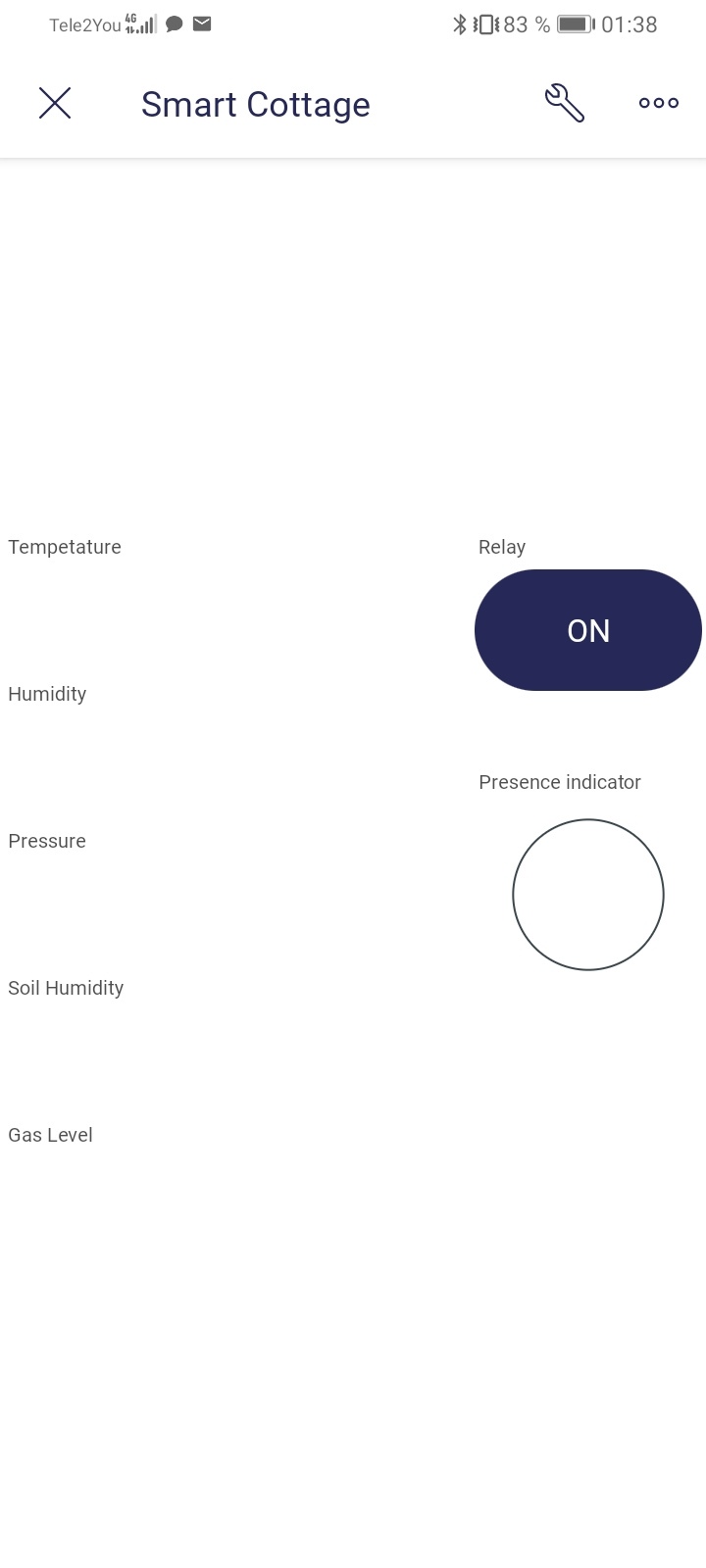


Рисунок 4.6 – вид приложения Blynk для модели устройства

# Моделирование системы с использованием протокола ZigBee

При моделировании данной системы сначала нужно разобраться с топологией разрабатываемой сети. Так как особенностью ZigBee сетей является высокая отказоустойчивость, достигающаяся при помощи перераспределения сети, то есть смысл использовать эту функцию для разработки топологии. В данном случае каждый узел сети (связка Arduino UNO и модуль XBee) будет рассматриваться как маршрутизатор, а связь с Интернетом будет осуществлять плата ESP8266, на которой будет расположен XBee-координатор. Также при использовании MESH топологии упрощается коммутация между отдельными блоками системы, поэтому имеет смысл разнести блоки 7 и 6, представленные на рисунке 4.1, в отдельные узлы. Топология такой сети представлена на рисунке 5.1.

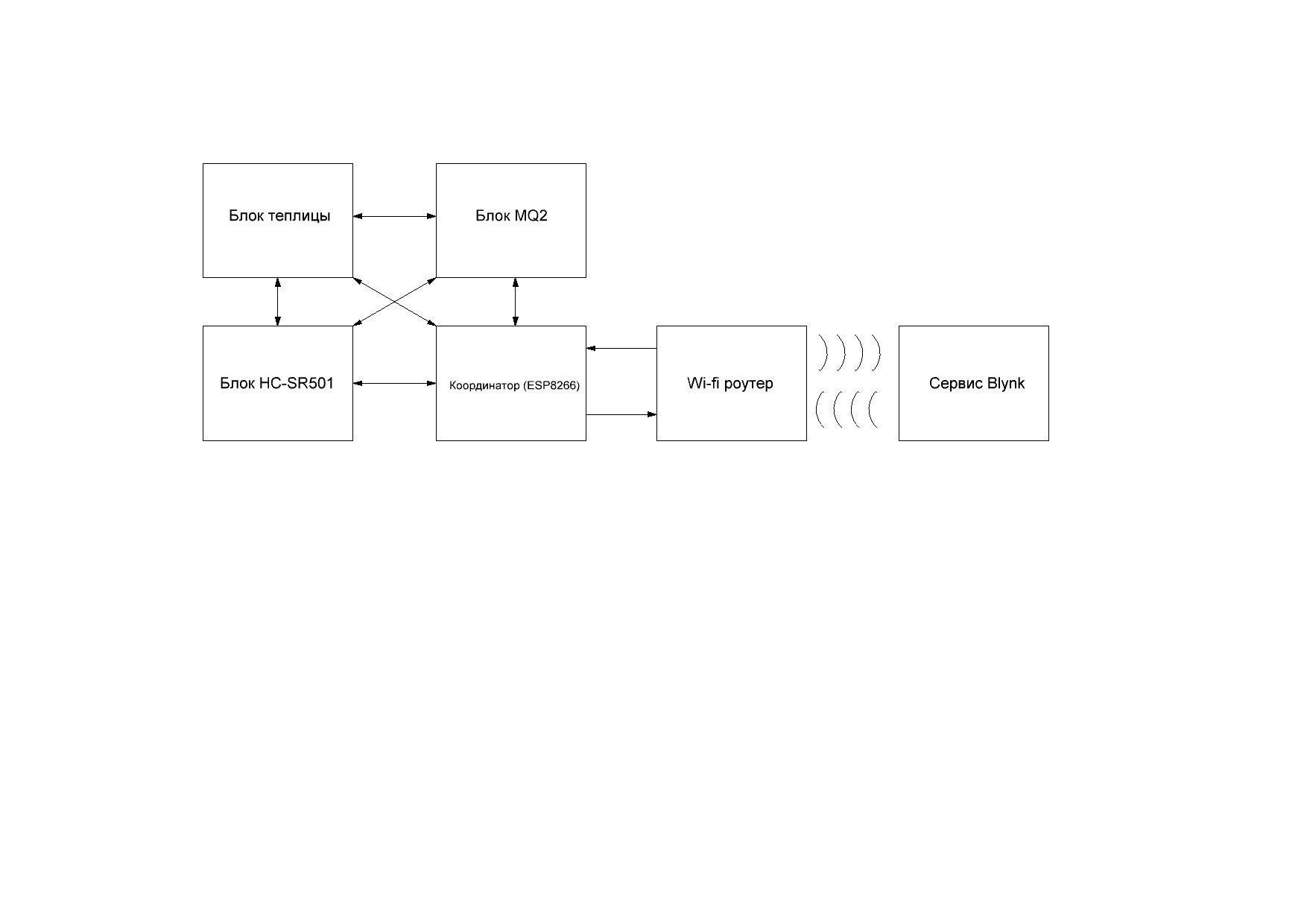


Рисунок 5.1 – схема передачи данных в модели

Расположение блоков на участке останется таким же, как и в пункте 4, за исключением разбитых на два узла блоков с датчиком MQ-2 и датчиком HC-SR501. Подключение элементов к плате Arduino UNO также ничем не отличается от показанного в пункте 4 подключения к плате WeMos D1 R1, кроме как использованием контактов RX и TX для подключения радио модуля XBee.

Важным моментом является передача данных внутри сети. Так как топология сети предусматривает передачу данных между всеми узлами сети одновременно, то выделение информации для каждого блока из общего потока данных будет происходить программно платой Arduino UNO (Блоки 4 в приложениях В, Г, Д и Е).

Передача данных на сервис Blynk и интерфейс приложения остаются такими же, как и в пункте 4.

Можно сказать, что в результате изменения топологии сети отказоустойчивость системы увеличилась. Во-первых, при непредвиденном выходе из строя одного из узлов системы, передача данных между блоками не прекратится. Во-вторых, протокол ZigBee предусматривает шифрование AES-128, что исключает случаи взлома сети и утечки данных.

# Заключение

В результате работы были рассмотрены периферийные устройства для модульного решения «умная дача». Были рассмотрены несколько вариантов коммутации модулей системы между собой.

При сравнивании всех способов коммутации между собой можно сделать следующие выводы:

1. Для организации передачи данных на небольшие расстояния выгодно использовать протоколы ESP-MASH и ESP-NOW, так как эти протоколы используют возможность платы ESP8266 передавать данные посредством встроенной в плату антенны.
2. Для обмена данными между двумя платами целесообразно использовать протокол ESP-NOW. Если передача данных будет осуществляться между несколькими модулями системы, тогда удобнее всего будет строить MASH сеть, создавая нужную топологию сети для конкретной системы.
3. Протокол LoRaWAN подходит для передачи данных на большие расстояния, поэтому в проектах «умного дома» его можно использовать только для объединения модулей близкорасположенных квартир или домов.
4. Так же можно сказать, что все рассмотренные протоколы имеют высокий уровень защищенности данных, так как в каждом протоколе перед передачей информации данные шифруются универсальным ключом, что усложняет попытки перехватить данные, или изменить передающийся пакет данных.

Отдельного внимания требует протокол ZigBee. Хоть он и имеет топологию MASH-сети, но имеет существенные отличия. Во-первых, ZigBee-сеть является самоорганизующейся. При назначении ролей всех узлов сеть сама возведет необходимые соединения для более лучшей работы сети. Во-вторых, ZigBee имеет более высокий уровень безопасности, по сравнению с обычной MESH-сетью.

Так же была рассмотрена аналитическая модель системы «умная дача». На примере данной системы была рассмотрена работа протокола ESP-NOW и ZigBee, а также был произведен обзор на специализированный сервис для проектов Интернета вещей – Blynk.

Список использованных источников

1. Росляков А.В. Р75 Интернет вещей: учебное пособие / А.В. Росляков, С.В. Ваняшин, А.Ю. Гребешков. – Самара: ПГУТИ, 2015. – 200 с. // стр. 5, 7-8, 13-14
2. WeMos D1 R1. Техническое описание – URL: [**https://arduinomaster.ru/datchiki-arduino/esp8266-wemos-d1-mini-raspinovka/**](https://arduinomaster.ru/datchiki-arduino/esp8266-wemos-d1-mini-raspinovka/)
3. Петин В.А. Проекты с использованием контроллера Arduino. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2016. – 400 с. // стр. 170-173, 178-193
4. Digital-output relative humidity & temperature sensor/module DHT22. Aosong Electronics. DataSheet – URL: [**https://cdn-shop.adafruit.com/datasheets/DHT22.pdf**](https://cdn-shop.adafruit.com/datasheets/DHT22.pdf)
5. BMP180 Digital Pressure Sensor. Bosh Sensortec. DataSheet – URL: [**https://robot-kit.ru/wa-data/public/blog/download/BST-BMP180.pdf**](https://robot-kit.ru/wa-data/public/blog/download/BST-BMP180.pdf)
6. MQ2 Smoke Sensor (WINSEN). Manual – URL: [**https://www.compel.ru/infosheet/WINSEN/MQ-2%20Smoke%20Sensor**](https://www.compel.ru/infosheet/WINSEN/MQ-2%20Smoke%20Sensor)
7. HC-SR501 PIR MOTION DETECTOR. HK Shan Hai Group Limited. Datasheet – URL: [**https://static.chipdip.ru/lib/319/DOC005319544.pdf**](https://static.chipdip.ru/lib/319/DOC005319544.pdf)
8. Capacitive soil moisture sensor v2.0 DataSheet – URL: [**https://static.chipdip.ru/lib/149/DOC007149803.pdf**](https://static.chipdip.ru/lib/149/DOC007149803.pdf)
9. SONGLE RELAY. RELAY ISO9002 DataSheet – URL: [**https://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf**](https://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf)
10. ESP-MESH. Espressif. Programming Guide – URL: [**https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/mesh.html**](https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/mesh.html)
11. LoRaWAN. LoRa Alliance Inc. Specification – URL: [**https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1\_0\_2-20161012\_1398\_1.pdf**](https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1_0_2-20161012_1398_1.pdf)
12. ESP-NOW. Espressif. User Guide – URL: [**https://www.espressif.com/sites/default/files/documentation/esp-now\_user\_guide\_en.pdf**](https://www.espressif.com/sites/default/files/documentation/esp-now_user_guide_en.pdf)
13. ZigBee. ZigBee Alliance. ZigBee Spetification – URL: <https://www3.nd.edu/~mhaenggi/ee67011/zigbee.pdf>
14. XBee. XBee/XBee PRO S2C Zigbee RF Module – URL: <https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf>

ПРИЛОЖЕНИЕ А

/\*

Прошивка для тепличного блока

\*/

// Блок 1: Подключение необходимых библиотек и инициализация подключенных модулей

#include <ESP8266WiFi.h>

#include <espnow.h>

#include <Adafruit\_Sensor.h>

#include <math.h>

#include <SFE\_BMP180.h>

#include <Wire.h>

#include <DHT.h>

#define DHTPIN1 13 // Подключение DHT22 к пину 3 GPIO0

SFE\_BMP180 pressure; // Переменная для датчика BMP180

#define ALTITUDE 151.0 // Высота над уровнем моря в метрах

DHT dht(DHTPIN, DHT22);

#define RELAY\_IN 2

#define HUM\_SOIL\_PIN 3

// МАС-АДРЕС платы, на которую отправляем данные

uint8\_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

// Вводим переменные для хранения отправляемых данных

float temperature;

float humidity;

char status;

double P;

String pressure;

float humidity\_soil;

// Вводим переменные для хранения принимаемых данных

float incomingTemp;

float incomingHum;

// Обновляем показания датчика каждые 10 секунд

const long interval = 10000;

unsigned long previousMillis = 0; // Время последнего обновления

// Переменная для хранения состояния отправки

String success;

// Структура для отправки сообщения другой плате. Должна совпадать со структурой на плате-приемнике

typedef struct struct\_message {

float temp;

float hum;

String pres;

float hum\_soil;

bool relay\_status;

} struct\_message;

// Создаем переменную для хранения отправляемого сообщения

struct\_message Readings;

// Создаем переменную для для принимаемого сообщения

struct\_message incomingReadings;

// Блок 2: Callback-функция для получения состояния отправки

void OnDataSent(uint8\_t \*mac\_addr, uint8\_t sendStatus) {

Serial.print("Last Packet Send Status: ");

if (sendStatus == 0){

Serial.println("Delivery success");

}

else {

Serial.println("Delivery fail");

}

}

// Блок 3: Callback-функция для индикации состояния приема данных

void OnDataRecv(uint8\_t \* mac, uint8\_t \*incomingData, uint8\_t len) {

memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));

incomingTemp = incomingReadings.temp;

incomingHum = incomingReadings.hum;

incomingPres = incomingReadings.pres;

incomingHumSoil = incomingReadings.hum\_soil;

incomingRelayStatus = incomingReadings.relay\_status;

}

// Блок 4: Функция снятия показаний с датчиков

void getReadings(){

temperature = dht.readTemperature();

if (isnan(temperature)){

Serial.println("Failed to read from DHT");

temperature = 0.0;

}

humidity = dht.readHumidity();

if (isnan(humidity)){

Serial.println("Failed to read from DHT");

humidity = 0.0;

}

status = pressure.startPressure(1);

delay(status);

status = pressure.getPressure(P);

pres = String(P\*0.750064);

humidity\_soil = analogRead(HUM\_SOIL\_PIN);

}

// Блок 5: Настройка устройств и подключений

void setup() {

dht.begin();

pinMode(RELAY\_IN, OUTPUT);

WiFi.mode(WIFI\_STA);

WiFi.disconnect();

// Инициализируем протокол ESP-NOW

if (esp\_now\_init() != 0) {

return;

}

// Указываем роль платы в сети

esp\_now\_set\_self\_role(ESP\_NOW\_ROLE\_SLAVE);

// Регистрируем callback-функцию для получения статуса отправки

esp\_now\_register\_send\_cb(OnDataSent);

// Регистрируем пиры

esp\_now\_add\_peer(broadcastAddress, ESP\_NOW\_ROLE\_SLAVE, 1, NULL, 0);

// Регистрируем callback-функцию для получения статуса приема

esp\_now\_register\_recv\_cb(OnDataRecv);

}

void loop() {

// Блок 6: Формирование сообщения. Прием и передача сообщений

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

// Сохраняем время последнего обновления показаний

previousMillis = currentMillis;

//Запрашиваем показания датчика

getReadings();

//Записываем их в переменные

Readings.temp = temperature;

Readings.hum = humidity;

Readings.pres = pressure;

Readings.hum\_soil = humidity\_soil;

if(relay\_ststus == true)

digitalWrite(RELAY\_IN, HIGH);

if(relay\_status == false)

digitalWrite(RELAY\_IN, LOW);

// Отправка сообщения и принятие данных от второй платы

esp\_now\_send(broadcastAddress, (uint8\_t \*) &Readings, sizeof(Readings));

}

}

ПРИЛОЖЕНИЕ Б

/\*

Прошивка для блока жилого помешения

\*/

// Блок 1: Подключение необходимых библиотек и инициализация подключенных модулей

#define BLYNK\_PRINT Serial

// Данные аунтентификации для поиска сервисом Blynk данного утройства

#define BLYNK\_TEMPLATE\_ID "\*\*\*\*\*\*\*\*\*\*\*"

#define BLYNK\_DEVICE\_NAME "\*\*\*\*\*\*\*\*\*\*\*"

#define BLYNK\_AUTH\_TOKEN "\*\*\*\*\*\*\*\*\*\*\*"

#include <ESP8266\_HardSer.h>

#include <BlynkSimpleShieldEsp8266\_HardSer.h>

// Переменные для подлкючения к сети Wi-Fi

char auth[] = " ";

char ssid[] = " ";

char pass[] = " ";

// // МАС-АДРЕС платы, на которую отправляем данные

uint8\_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

// Вводим переменные для хранения отправляемых данных

float temperature;

float humidity;

char status;

double P;

String presure;

float humidity\_soil;

#define MQ2\_PIN A0;

float MQ2value;

#define pir\_pin 7;

int pirValue;

// Структура для отправки сообщения другой плате. Должна совпадать со структурой на плате-приемнике

typedef struct struct\_message {

float temp;

float hum;

String pres;

float hum\_soil;

bool relay\_status;

} struct\_message;

// Создаем переменную для хранения отправляемого сообщения

struct\_message Readings;

// Блок 2: Функция обратного вызова, которая будет выполнена при получении данных

void OnDataRecv(uint8\_t \* mac, uint8\_t \*incomingData, uint8\_t len) {

memcpy(&Readings, incomingData, sizeof(Readings));

}

// Блок 3: Настройка устройств и подключений

void setup()

{

WiFi.mode(WIFI\_AP\_STA);

// Инициализация подключения к сервису Blynk

Blynk.begin(auth, ssid, pass);

// Инициализация протокола ESP-NOW

if (esp\_now\_init() != 0) {

return;

}

// Установка роли платы в протоколе ESP-NOW

esp\_now\_set\_self\_role(ESP\_NOW\_ROLE\_CONTROL);

esp\_now\_register\_recv\_cb(OnDataRecv);

// Попытка подключиться к серверу

resultConnection = Blynk.connect();

// Указание получателя сообщения

esp\_now\_peer\_info\_t peerInfo;

memcpy(peerInfo.peer\_addr, broadcastAddress, 6);

peerInfo.channel = 0;

peerInfo.encrypt = false;

if (esp\_now\_add\_peer(&peerInfo) != ESP\_OK){

return;

}

}

void loop()

{

// Блок 4: Считывание данных с датчиков, подключенных к плате

MQ2value = analogRead(MQ2\_PIN);

pirValue = analogRead(pir\_pin);

// Блок 5: Запуск обмена данных с сервисом Blynk при успешном подключении к сервису

if(resultConnection) Blynk.run();

// Блок 6: Указание данных для отправки на другую плату

Readings.temp = temperature;

Readings.hum = humidity;

Readings.pres = pressure;

Readings.hum\_soil = humidity\_soil;

// Отправка сообщения и принятие данных от второй платы

esp\_err\_t result = esp\_now\_send(broadcastAddress, (uint8\_t \*) &Readings, sizeof(Readings));

sendData();

}

// Блок 7: Отправка полученных данных на сервис

void sendData()

{

Blynk.virtualWrite(V5, temperature);

Blynk.virtualWrite(V6, humidity);

Blynk.virtualWrite(V7, pressure);

Blynk.virtualWrite(V8, humidity\_soil);

Blynk.virtualWrite(V9, MQ2value);

Blynk.virtualWrite(V10, pirValue);

}

// Блок 8: Вызов функции при обновлении виртуального порта V0

BLYNK\_WRITE(V0)

{

// Считываем новое значение порта

relay\_status = param.asInt();

}

ПРИЛОЖЕНИЕ В

/\*

Прошивка для блока координатора

\*/

// Блок 1: Подключение необходимых библиотек и инициализация подключенных модулей

#define BLYNK\_PRINT Serial

// Данные аунтентификации для поиска сервисом Blynk данного утройства

#define BLYNK\_TEMPLATE\_ID "\*\*\*\*\*\*\*\*\*\*\*"

#define BLYNK\_DEVICE\_NAME "\*\*\*\*\*\*\*\*\*\*\*"

#define BLYNK\_AUTH\_TOKEN "\*\*\*\*\*\*\*\*\*\*\*"

#include <ESP8266\_HardSer.h>

#include <BlynkSimpleShieldEsp8266\_HardSer.h>

#include <SoftwareSerial.h>

SoftwareSerial XBee(2,3);

// Переменные для подлкючения к сети Wi-Fi

char auth[] = " ";

char ssid[] = " ";

char pass[] = " ";

// // МАС-АДРЕС платы, на которую отправляем данные

uint8\_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

// Вводим переменные для хранения отправляемых данных

float temperature;

float humidity;

char status;

double P;

String presure;

float humidity\_soil;

String analogMQ2;

String analogPir;

// Блок 3: Настройка устройств и подключений

void setup()

{

WiFi.mode(WIFI\_AP\_STA);

Serial.begin(9600);

XBee.begin(9600);

// Инициализация подключения к сервису Blynk

Blynk.begin(auth, ssid, pass);

// Попытка подключиться к серверу

resultConnection = Blynk.connect();

}

void loop()

{

// Блок 4: Запуск обмена данных с сервисом Blynk при успешном подключении к сервису

if(resultConnection) Blynk.run();

// Блок 5: Указание данных для отправки на другие блоки

Readings.temp = temperature;

Readings.hum = humidity;

Readings.pres = pressure;

Readings.hum\_soil = humidity\_soil;

// Отправка сообщения и принятие данных от других блоков

if(XBee.read() > 0)

{

sendData();

}

XBee.write(relay\_status);

}

// Блок 6: Отправка полученных данных на сервис

void sendData()

{

Blynk.virtualWrite(V5, temperature);

Blynk.virtualWrite(V6, humidity);

Blynk.virtualWrite(V7, pressure);

Blynk.virtualWrite(V8, humidity\_soil);

Blynk.virtualWrite(V9, analogMQ2);

Blynk.virtualWrite(V10, analogPir);

}

// Блок 7: Вызов функции при обновлении виртуального порта V0

BLYNK\_WRITE(V0)

{

// Считываем новое значение порта

relay\_status = param.asInt();

}

ПРИЛОЖЕНИЕ Г

/\*

Прошивка для блока теплицы

\*/

// Блок 1: Подключение библиотек и инициализация подключенных модулей

#include <math.h>

#include <SFE\_BMP180.h>

#include <Wire.h>

#include <DHT.h>

#include <SoftwareSerial.h>

SoftwareSerial XBee(2,3);

#define DHTPIN 5 // Подключение DHT22 к пину 5

#define DHTPIN1 3 // Подключение DHT11 к пину 3

#define relay\_pin 6 // Подключение реле к пину 6

SFE\_BMP180 pressure; // Переменная для датчика BMP180

#define ALTITUDE 151.0 // Высота над уровнем моря в метрах

DHT dht(DHTPIN, DHT22); //Инициация датчиков

DHT dht1(DHTPIN1, DHT11);

float h1;

float t1;

float h;

float t;

char status;

double T,P;

int relay\_status;

// Блок 2: Настройка устройств и подключений

void setup()

{

// Запуск последовательного порта

Serial.begin(9600);

XBee.begin(9600);

dht.begin(); // Инициализируем DHT22

dht1.begin(); // Инициализируем DHT11

if (pressure.begin())

{

// Проверка на подключение BMP180

Serial.print("BMP180 ok");

}

else

{

Serial.print("BMP180 fail");

delay(2000);

while(1); // Бесконечный цикл. Если есть сообщение об ошибке, нужно проверить подключение BMP180 и перезапустить плату (выключить и включить питание)

}

}

void loop()

{

// Блок 3: Снятие показаний с датчиков

String analogValue0;

String analogValue1;

h1 = dht1.readHumidity(); //Измеряем влажность

t1 = dht1.readTemperature(); //Измеряем температуру

if (isnan(h1) || isnan(t1)) { // Проверка. Если не удается считать показания, выводится «Ошибка считывания», и программа завершает работу

Serial.println("Ошибка считывания");

return;

}

status = pressure.startTemperature(); // Старт измерения температуры

delay(status);

status = pressure.getTemperature(T);

analogValue0 = String(T,DEC);

// Определяем атм. давление:

status = pressure.startPressure(1);

delay(status);

status = pressure.getPressure(P,T);

analogValue1 = String(P\*0.750064);

h = dht.readHumidity(); //Измеряем влажность

t = dht.readTemperature(); //Измеряем температуру

if (isnan(h) || isnan(t)) { // Проверка. Если не удается считать показания, выводится «Ошибка считывания», и программа завершает работу

Serial.println("Ошибка считывания DHT22");

return;

}

// Запись данных в переменные String

String analogValue2 = String(t1,DEC);

String analogValue3 = String(h1);

String analogValue4 = String(t, DEC);

String analogValue5 = String(h);

// Блок 4: Отправка данных

if(XBee.connected())

{

XBee.write("&field4="+analogValue3+"&field5="+analogValue4+"&field6="+analogValue5+"&field1="+analogValue0+"&field2="+analogValue1+"&field3="+analogValue2);

Serial.println(analogValue0);

Serial.println(analogValue1);

Serial.println(analogValue2);

Serial.println(analogValue3);

Serial.println(analogValue4);

Serial.println(analogValue5);

}

// Блок 5: Управление реле

if(relay\_status == 1)

{

digitalWrite(relay\_pin, HIGH);

} else {

digitalWrite(relay\_pin, LOW);

}

}

ПРИЛОЖЕНИЕ Д

/\*

Прошивка для блока MQ2

\*/

// Блок 1: Подключение необходимых библиотек и объявление входов датчика

#include <SoftwareSerial.h>

#define MQ2\_PIN A0;

float MQ2value;

SoftwareSerial XBee(2,3);

// Блок 2: Настройка последовательной передачи данных и настройка входа датчика

void setup()

{

Serial.begin(9600);

XBee.begin(9600);

pinMode(MQ2\_PIN, OUTPUT);

}

void loop()

{

// Блок 3: Считывание данных с датчика

MQ2value = analogRead(MQ2\_PIN);

String analogValue = String(MQ2value, DEC);

// Блок 4: Передача данных в общий поток сети

if(XBee.connected())

{

XBee.write(analogMQ2);

}

}

ПРИЛОЖЕНИЕ Е

/\*

Прошивка для блока HC-SR501

\*/

// Блок 1: Подключение необходимых библиотек и объявление входов датчика

#include <SoftwareSerial.h>

#define PIR\_PIN A0;

float pir\_value;

SoftwareSerial XBee(2,3);

// Блок 2: Настройка последовательной передачи данных и настройка входа датчика

void setup()

{

Serial.begin(9600);

XBee.begin(9600);

pinMode(PIR\_PIN, OUTPUT);

}

void loop()

{

// Блок 3: Считывание данных с датчика

pir\_value = analogRead(PIR\_PIN);

String analogPir = String(pir\_value, DEC);

// Блок 4: Передача данных в общий поток сети

if(XBee.connected())

{

XBee.write(analogPir);

}

}