«Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	11.03.01 – Радиотехника
Профиль	Радиоэлектронные системы
Факультет	РТ
Кафедра	РС

К защите допустить

Зав. кафедрой _____

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: Организация ввода данных цифро-аналоговых сенсоров в среде Arduino

Студент		_____ *подпись*	Шатеров А.А.
Руководитель	к.т.н., доц. *(Уч. степень, уч. звание)*	_____ *подпись*	Смирнов Б.И.
Консультанты	ассистент *(Уч. степень, уч. звание)*	_____ *подпись*	Проценко И.М.

Санкт-Петербург

2023

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. кафедрой РС

_____

«___»_____2023 г.

Студент Шатеров А.А. Группа 9101

Тема работы: Организация ввода данных цифро-аналоговых сенсоров в среде Arduino

Место выполнения ВКР: СПбГЭТУ “ЛЭТИ”

Исходные данные (технические требования): разработка кода программы взаимодействия устройств, составляющих лабораторный макет, для получения данных с цифровых и аналоговых датчиков и последующей передачи их на ПК, а так же буферизацию на сервере.

Содержание ВКР: введение, протоколы передачи информации согласно модели OSI, код программы, результат разработки, дополнительный раздел, заключение.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал, презентация в среде Power-Point, программный код.

Дополнительные разделы: безопасность жизнедеятельности

Дата выдачи задания «___» _____ 2023 г.

Дата представления ВКР к защите «____» _____2023 г.

Студент _____ Шатеров А.А.

Руководитель к.т.н., доц. _____ Смирнов Б.И.
(Уч. степень, уч. звание)

Консультант ассистент _____ Проценко И.М.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой РС

_____

«___»_____2023 г.

Студент Шатеров А.А. Группа 9101

Тема работы: Организация ввода данных цифро-аналоговых сенсоров в среде Arduino

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	10.03-21.03
2	Разработка кода программного модуля	21.04-05.04
3	Описание устройств лабораторного макета	06.04-15.04
4	Протоколы передачи данных микроконтроллера ESP8266	15.04-25.04
5	Исследование применимости протоколов различных уровней к устройствам на базе лабораторного макета	26.04-09.05
6	Безопасность жизнедеятельности	10.05-13.05
7	Оформление пояснительной записки	13.05-17.05
8	Оформление иллюстративного материала	18.05-25.05

Студент _____ Шатеров А.А.

Руководитель к.т.н., доц. _____ Смирнов Б.И.
(Уч. степень, уч. звание)

Консультант ассистент _____ Проценко И.М.
(Уч. степень, уч. звание)

Оглавление

РЕФЕРАТ

Пояснительная записка содержит: 78 страниц, 47 рисунков, 10 таблиц, 1 приложений, 43 источника литературы.

Ключевые слова: ПЕРЕДАЧА ДАННЫХ, ПРИНЦИП ДЕЙСТВИЯ, ESP-8266, ПРОТОКОЛЫ МОДЕЛИ OSI.

Цель работы - разработка кода программы взаимодействия устройств, составляющих лабораторный макет, для получения данных с цифровых и аналоговых датчиков и последующей передачи их на ПК, а так же буферизацию на сервере.

В работе представлено описание протоколов передачи цифровой информации модели OSI, а также разработка кода программы взаимодействия устройств, составляющих лабораторный макет.

Основной задачей работы является изучение технологии получения данных цифро-аналоговых сенсоров в среде Arduino на плате ESP-8266 для последующей передачи на ПК, а также рассмотрение целесообразности использования различных протоколов передачи данных на различных уровнях модели OSI.

ABSTRACT

The main objective of the work is to study the technology of obtaining data from digital-analog sensors in an Arduino environment on an ESP-8266 board for subsequent transmission to a PC, as well as to consider the feasibility of using various data transfer protocols at different levels of the OSI model.

As a result, optimal protocols were identified to ensure the most efficient operation for the laboratory set. These protocols are Wi-Fi 802.11n on the physical, IPv6 on the network, TCP on the transport, MQTT on the application layer of the OSI model.

ВВЕДЕНИЕ

С самых древних времен у человека возникла необходимость передавать различную информацию на расстояния. Задолго до открытия радио, да и вообще электричества, люди уже были способны обмениваться информацией через достаточно большие промежутки пространства.

Первобытный человек полагался на сигналы огня и дыма, а также на барабанные сообщения для кодирования информации в ограниченной географической области, когда они пытались связаться с соседними племенами. Эти сигналы также должны были иметь очень простые, заранее определенные значения, такие как «безопасный» или «опасность» или «победа», или могли использоваться в качестве системы сигнализации для оповещения доисторических племен, к примеру, о вторжении чужаков.

Информация в современном мире по-прежнему играет одну из важнейших ролей. Сегодня у людей имеется широчайший выбор различных средств связи, однако и объемы передаваемых данных возросли до невообразимых размеров.

Скорость и целостность переданной информации являются высшим приоритетом при разработке новейших средств связи. Чем точнее и быстрее информация дойдет до одного получателя, чем до второго, тем большее преимущество будет иметь первый. Либо же наоборот, чем более искаженными окажутся данные, тем хуже будет их получателю.

Подводя итог вышесказанному, можно заключить, что всё это подкрепляет актуальность проводимого в данной работе исследования, которое посвящено выявлению протоколов передачи, обеспечивающих наиболее эффективный обмен информацией на примере лабораторной установки.

1. Описание устройств лабораторного макета

1.1. Микроконтроллер ESP-8266

1.1.1. Модуль ESP-12E

Отладочная плата ESP-8266 оснащена модулем ESP-12E, содержащим микросхему ESP8266 с микропроцессором RISC 32-bit LX106, который работает на тактовой частоте, регулируемой в пределах от 80 до 160 МГц а имеет поддержку RTOS.

Характеристики ESP-12E [1]:

- 32-разрядный LX106 от Tensilica Xtensa
- Тактовая частота от 80 до 160 МГц
- 128 КБ встроенной оперативной памяти
- 4 МБ внешней внешней флеш-памяти
- Приемопередатчик Wi-Fi 802.11b/g/n

Данный модуль содержит 128 КБ ОЗУ и 4 МБ флеш-памяти (для хранения различных данных и программ), что дает возможность справляться с большими объемами входных данных, состоящих из веб-страниц, JSON/XML-запросов и т.п.

ESP8266 содержит встроенный передатчик Wi-Fi 802.11b/g/n HT40, он способен не только взаимодействовать с сетью через подключения посредством Wi-Fi, но также формировать собственную локальную сеть и предоставлять возможность подключаться напрямую к ESP-8266 другим устройствам. Это делает ESP8266 NodeMCU крайне универсальным.

Рисунок 1. ESP-12E и встроенная антенна

1.1.2. Требования к питанию

Диапазон рабочего напряжения ESP8266 составляет от 3 В до 3,6 В [2], данный микроконтроллер для поддержания постоянного напряжения на уровне 3,3 В имеет LDO стабилизатор напряжения. Он способен обеспечивать ток до 600 мА. Этого достаточно, ведь ESP8266 потребляет ток около 80 мА в течении выполнения операций взаимодействия с данными. Выход стабилизатора, обозначенный как 3V3, выведен на контакты по сторонам платы. Данные выводы используются для подведения питания на подключенные внешние компоненты.

- Рабочее напряжение: от 2,5 до 3,6 В
- Встроенный стабилизатор: 3,3 В, 600 мА
- Рабочий ток: 80 мА
- Потребление в спящем режиме: 20 мкА

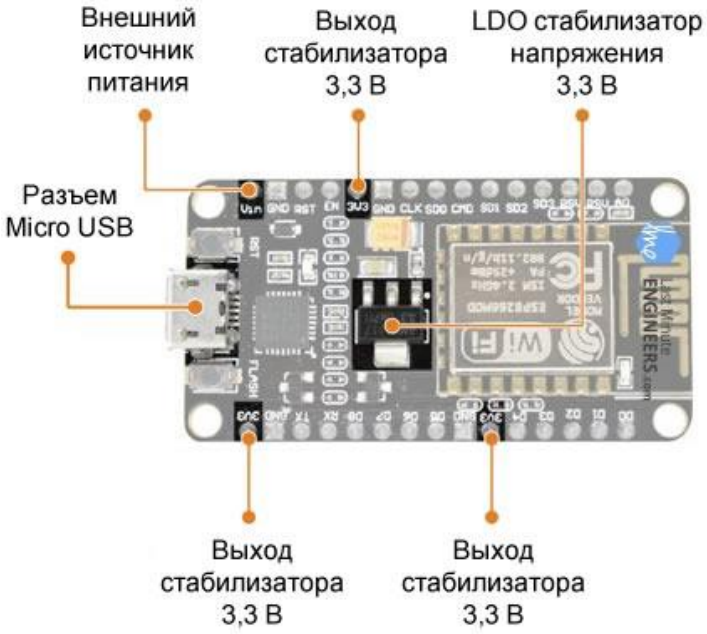

Рисунок 2. Элементы питания ESP8266 NodeMCU

Питание ESP8266 NodeMCU производится через встроенный USB-разъем.

В качестве альтернативы можно использовать вывод VIN для непосредственного питания ESP8266 и его периферии, если имеется стабилизированный источник напряжения 5 В [3].

1.1.3. Периферия и ввод/вывод

В общем ESP8266 имеет 17 выводов GPIO, находящихся на разъемах с обеих сторон отладочной платы. Среди них:

- Канал 10-разрядного АЦП;
- Интерфейс UART по последовательной связи используется для загрузки кода;
- Выходы ШИМ используются для управления подключенными двигателями, сервоприводами или для регулировки яркости подключенных светодиодов
- Интерфейсы SPI, I2C используются для получения данных с различных датчиков и дополнительных устройств;
- Интерфейс I2S используется для звуковой передачи.

Мультиплексируемые выводы ввода/вывода

- 1 канал АЦП
- 2 интерфейса UART
- 4 выхода ШИМ
- Интерфейсы SPI, I2C и I2S

Рисунок 3. Мультиплексируемые выводы GPIO платы ESP8266 NodeMCU

ESP8266 оснащена функцией мультиплексирования выводов (т.е. на один контакт вывода GPIO выводят несколько устройств). Это позволяет, например, одному выводу GPIO выполнять функции PWM/UART/SPI.

1.1.4. Кнопки и светодиодный индикатор на плате

На плате микроконтроллера ESP8266 находятся 2 кнопки. Одна расположена в верхнем левом углу и помечена как RST, нужна для асинхронного сброса микросхемы. Вторая кнопка - FLASH, в левом нижнем углу – кнопка загрузки, которая используется при обновлении прошивки.

Кнопки и индикаторы

- RST – сброс чипа ESP8266
- FLASH – загрузка новой программы
- Синий светодиод - программируется пользователем

На микросхеме также установлен программируемый светодиодный индикатор, подключенный к выводу D0 платы, который может быть использован для определенной функции.

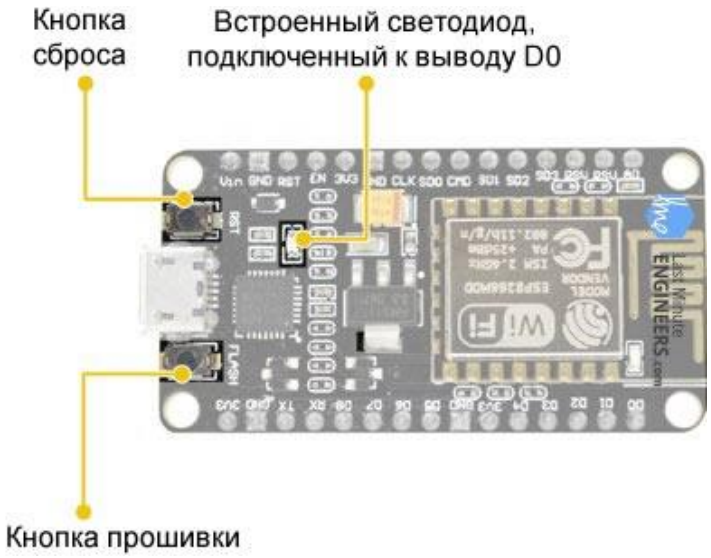

Рисунок 4. Кнопки и светодиоды на плате ESP8266 NodeMCU

1.1.5. Последовательная связь

На ESP-8266 расположен контроллер USB-UART CP2102. Он преобразует USB сигнал в сигнал последовательного порта и дает возможность компьютеру передавать информацию и взаимодействовать с микросхемой ESP8266.

Последовательная связь

- USB-UART преобразователь CP2102
- Скорость связи 4,5 Мбит/с
- Поддержка управления потоком

Рисунок 5. Преобразователь USB↔TTL CP2102

1.1.6. Контакты ESP8266

На плате ESP8266 находятся в общем 30 выводов. На рисунке 6 приведена схема контактов микроконтроллера.

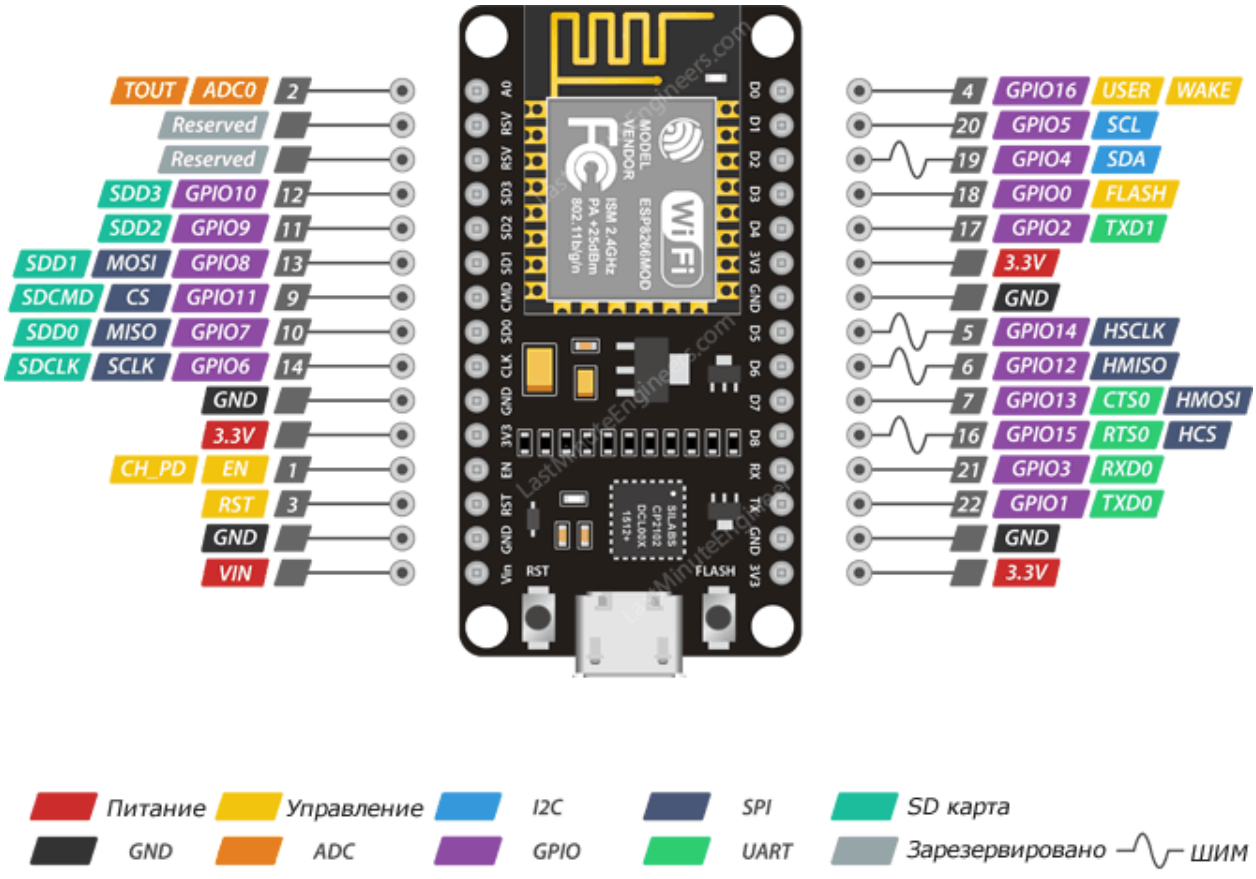

Рисунок 6. Распиновка ESP8266 NodeMCU

Сгруппируем выводы по функциям.

Выводы питания – на плате расположено 4 вывода питания: 1 вывод VIN и 3 вывода 3.3V. Вывод VIN можно используется для непосредственного питания ESP8266 и его периферии, при наличии стабилизированного источника напряжения 5 В. Выводы 3.3V – выходы встроенного стабилизатора напряжения. Они используются для подачи питания на подключаемые компоненты.

GND –вывод на землю платы ESP8266 NodeMCU.

Выводы I2C используются для подключения различных датчиков и внешних устройств по шине I2C. Поддерживаются протоколы I2C Master, и I2C Slave. Работа интерфейса I2C реализована программным методом, максимальная тактовая частота составляет 100 кГц. Необходимо учесть, что тактовая частота I2C должна быть выше наименьшей тактовой частоты ведомых устройств.

Выводы GPIO - на ESP8266 NodeMCU находятся 17 выводов GPIO. Каждый подключенный вывод может быть настроен либо на внутренний вывод к земле или к шине питания, либо установлен на z-состояние. Он может быть запрограммирован на срабатывание по фронту или по спаду для генерации сигналов прерывания процессора при конфигурировании на вход.

Вывод ADC подает сигнал на имеющийся на микроконтроллере, встроенный 10-разрядный прецизионный АЦП последовательного приближения (SAR ADC). С помощью него можно реализовать 2 функции: проверка напряжения питания на выводе VDD3P3 или проверка входного напряжения на выводе TOUT.

Выводы UART ESP8266 располагают двумя интерфейса UART: UART0 и UART1. Они обеспечивают асинхронную связь (RS232 и RS485) и могут обмениваться данными со скоростью вплоть до 4,5 Мбит/с. Для связи используют поддерживающий управление потоком интерфейс UART0 (выводы TXD0, RXD0, RST0 и CTS0). Интерфейс UART1 (вывод TXD1) используют для генерации журнала событий, так как он поддерживает только лишь сигнал передачи.

Выводы SPI имеют 2 интерфейса SPI (SPI и HSPI). Интерфейсы имеют ведомый (slave) и ведущий (master) режимы, а также поддерживают следующие SPI функции:

- до 64 байт FIFO
- до 80 МГц и тактовые частоты, полученные делением 80 МГц
- 4 режима синхронизации передачи SPI.

Выводы SDIO обладают защищенным цифровым интерфейсом ввода/вывода (SDIO, Secure Digital Input/Output Interface), использующегося

для прямого подключения SD-карт. Интерфейс поддерживает 4-битный 25 МГц SDIO v1.1 и 4-битный 50 МГц SDIO v2.0.

Выводы PWM - на плате предусмотрено 4 канала широтно-импульсной модуляции (PWM), частотный диапазон которых устанавливается в пределах от 1000 мкс до 10000 мкс, (100 Гц - 1 кГц). Выход ШИМ может быть реализован программным методом и использован для осуществления управления подключенным двигателями и светодиодами.

Выводы управления необходимы непосредственно для осуществления управления контроллером ESP8266. Состоят их выводов включения микросхемы EN, сброса RST и пробуждения WAKE.

- Вывод WAKE используется для вывода чипа из режима сна.
- Вывод EN – когда на вывод EN подается высокий логический уровень, микросхема ESP8266 включена. При низком логическом уровне микросхема работает в режиме минимальной мощности.
- Вывод RST используется для асинхронного сброса микросхемы ESP8266.

1.2. Датчик света и жестов APDS-9960

1.2.1. Общие сведения:

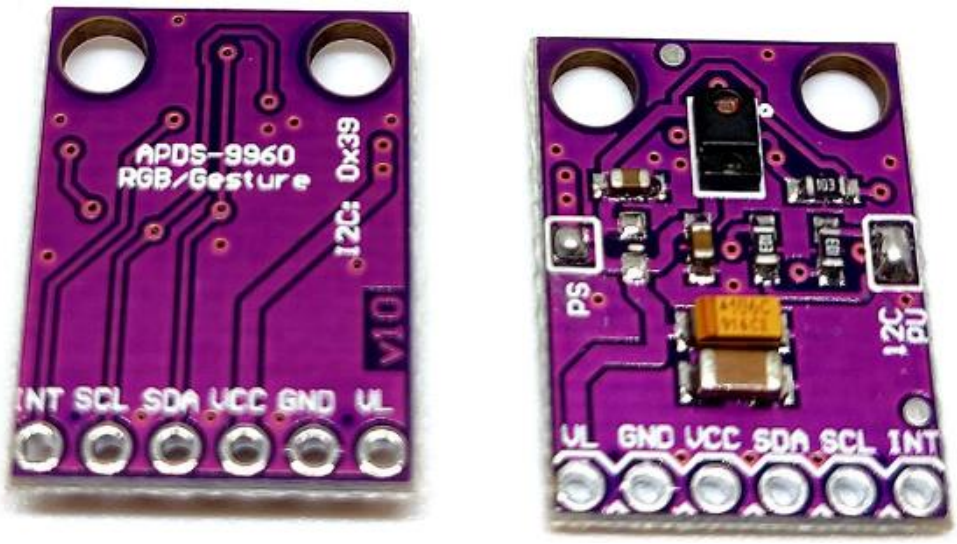

Рисунок 7. APDS-9960

Датчик жестов APDS-9960 — мультифункциональный ИК-датчик, способный обнаруживать и различать жесты, а также RGB цвет и освещенность. Обмен данными с микроконтроллерами осуществляется по шипе I2C. Рабочий уровень напряжения датчика составляет 3,3 В, но логическая часть может работать и на напряжении в 5 В.

Конструктивно датчик состоит из ИК-светодиода (излучателя) и 4 фотодиодов (приемников), обслуживающих ИК-излучение, а также располагает встроенным датчиком освещенности (ALS), ультрафиолетовым и инфракрасным фильстрами. Когда необходимо различить жест, датчик получает отраженное ИК излучение фотодиодами и выполняется расчет направления и скорости. Расстояние различения находится в пределах от 10 до 20 см.

Назначение контактов [6]:

- VL — дополнительное питание ИК-светодиода. Должно находится в пределах 3 – 4,5 В;
- VCC и GND — питание модуля, 3.3 В;
- SCL — шина I2C;
- SDA— шина I2C;
- INT — вывод внешнего прерывания.

На передней части датчика имеются две перемычки.

PS —соединяет источник питания датчика и ИК-светодиода. Когда она установлена, необходимо подключать только контакт VCC. Если же перемычки нет, нужно подавать питание на контакты VCC (2,4 — 3,6 В) и на VL (3,0 — 4,5 В) отдельно. Изначально перемычка установлена.

I2C PU — это 3-контактная перемычка, которую используют для подключения/отключения подтягивающих резисторов шины I2C. По умолчанию она установлена.

1.2.2. Принцип работы

Для получения необходимой информации о движении и направлении движения в APDS-9960 используются инфракрасный светодиод и 4 вспомогательных фотодиода, которые, как проиллюстрировано на рисунке 9, регистрируют сигналы в диапазоне ближнего инфракрасного излучения (NIR).

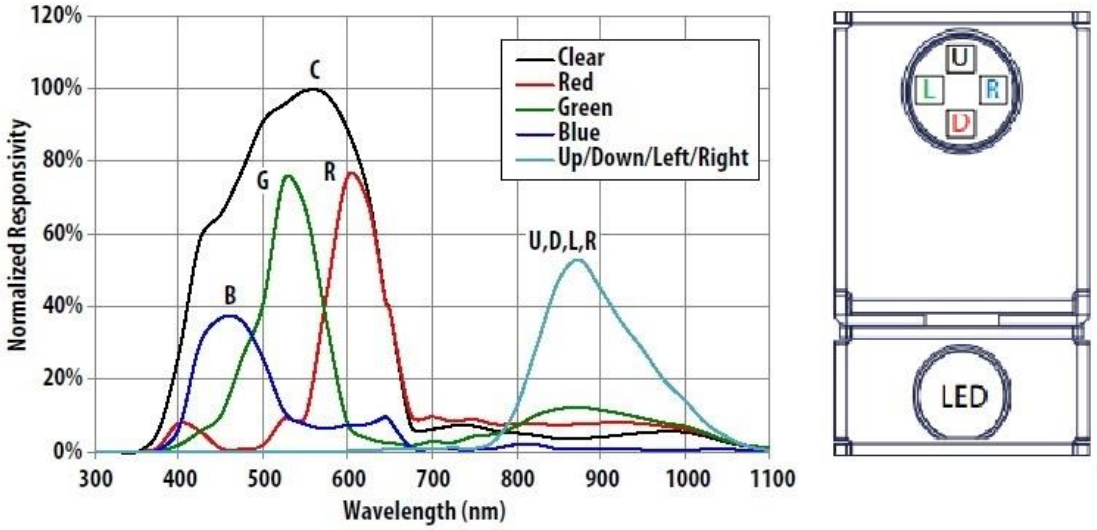

Рисунок 8. Действие сенсора в спектральной области

ИК-светодиод (LED) осуществляет функцию подсветки, а фотодиоды (UDLR) регистрируют отраженный от движущегося объекта свет.

Фотодиоды расположены на сенсоре таким образом, что соответствующий фотодиод получит наибольшую часть отраженного ИК-сигнала на входе и меньшую часть на выходе в зависимости от направления движения регистрируемого объекта. Интерпретировать же направление движения можно измеряя и сравнивая разность фаз и амплитуд сигналов, полученных на фотодиодах UDLR. Пример полученного на фотодиоде сигнала приведен на рисунке 10.

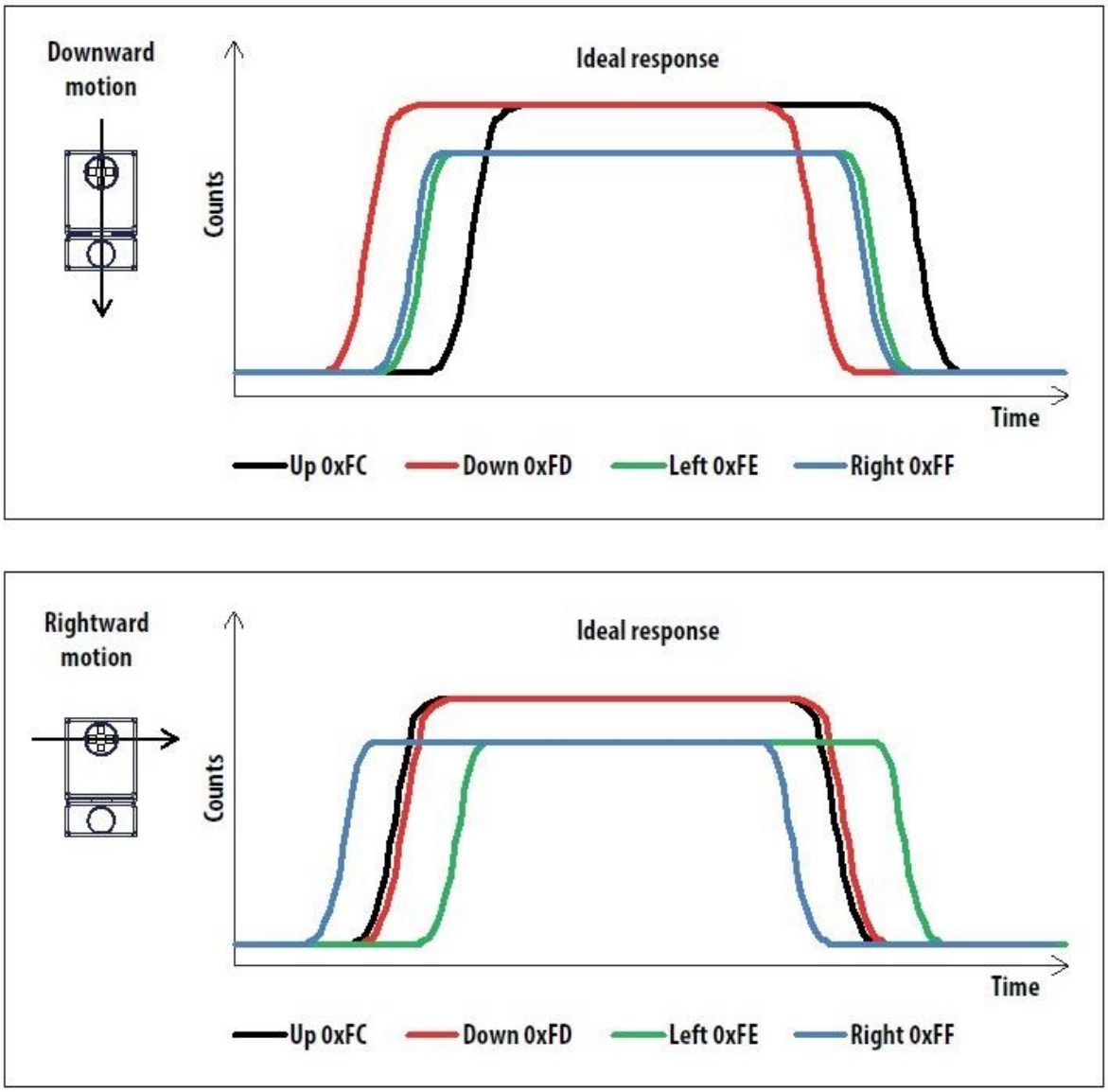

Рисунок 9. Реакция сенсора на направление сигнала

1.3. Датчик давления BMP-180

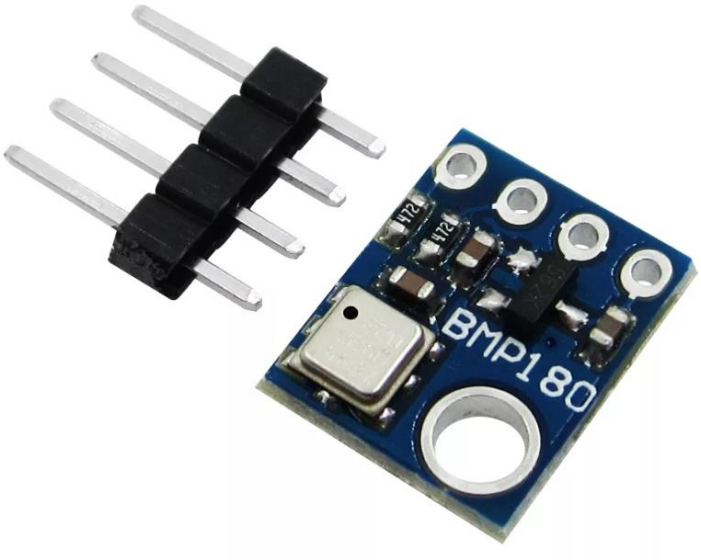

Рисунок 10. BMP-180

1.3.1. Устройство датчика BMP-180:

В левой части платы расположен датчик BMP180. Он работает от напряжения 3.3В. Но, так как почти все устройства Arduino работают на напряжении 5В, то на плате предусмотрен стабилизатор напряжения XC6206P332MR, который и выдает необходимое напряжение. Рядом с ним установлена обвязка стабилизатора, которая состоит из 2 керамических конденсаторов по 1 мкФ.

Подключение датчика осуществляется по шине I2C, линии SCL и SDA выведены на группу контактов на другой стороне модуля, куда также выведено питание. 2 резистора на 4.7 кОм, необходимы подтяжки линии SCL и SDA к питанию, которые можно удалить при использовании нескольких устройств на шине I2C.

На рисунке 12 приведено расположение контактов платы датчика BMP-180.

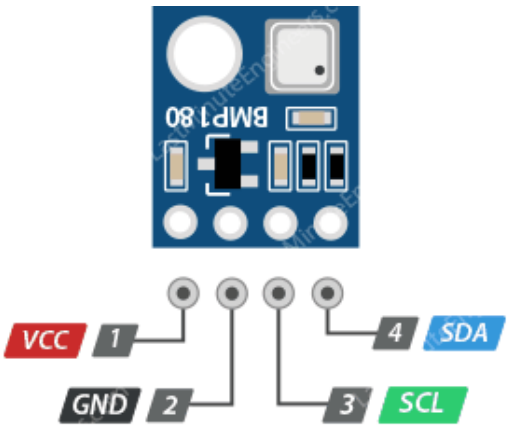

Рисунок 11. Контакты BMP-180

Назначение контактов [10]:

- SDA выполняет функцию вывода данных для интерфейса I2C.
- SCL является выводом синхронизации для интерфейса I2C.
- VIN используется для подключения источника питания для модуля, напряжение питания может находиться в диапазоне от 3,3 до 5 В.
- GND подключается к выводу земли на плате Arduino.

Принципиальная схема датчика приведена на рисунке 13.

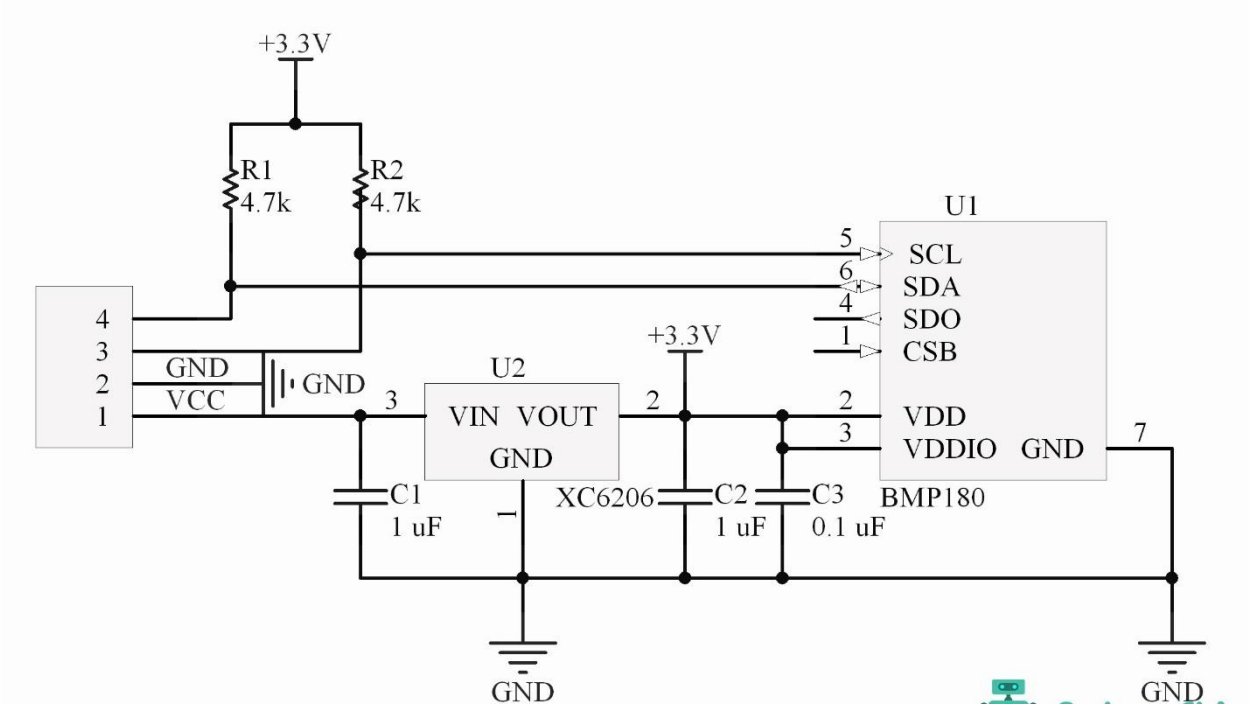

Рисунок 12. Принципиальная схема BMP-180

Основные характеристики BMP-180:

- Рабочий диапазон измеряемых значений: 300 гПа-1100 гПа (от -500м от +9000м над уровнем моря);
- напряжение питания: от 3.3 до 5 Вольт; сила тока: 5 мкА при скорости опроса — 1 Герц;
- уровень шума: 0.06 гПа (0.5м) в грубом режиме (ultra low power mode) и 0.02 гПа (0.17м) а режиме максимального разрешения (advanced resolution mode);
- Интерфейс: I2C;
- Время срабатывания: 4.5 мс.

1.3.2. Принцип действия датчика BMP-180:

В датчике расположена герметичная камера, одна из стенок которой является гибкой мембраной с установленными на ней тензодатчиками. При изменении давления между камерой и внешней средой, мембрана гнется, что изменяет сопротивления тензодатчиков электрическому току. Так же имеется термодатчик, сопротивление которого изменяется пропорционально температуре. Результат измерения датчиков АЦП (аналого-цифровой преобразователь) преобразует в цифровой формат - «некомпенсированные результаты». Для компенсации указанных результатов (компенсации смещения, температурной зависимости, погрешностей при изготовлении, неоднородностей материалов и т.д.) каждый датчик калибруется на заводе, и в

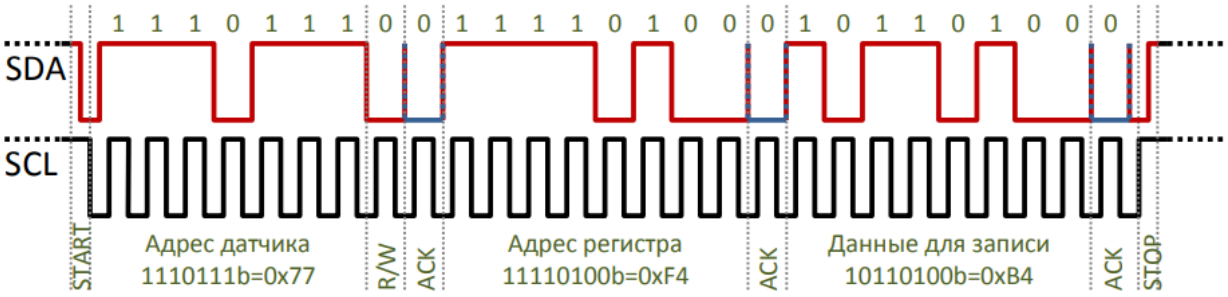

Рисунок 13. Пример записи в регистр 0xF4 значения 0xB4

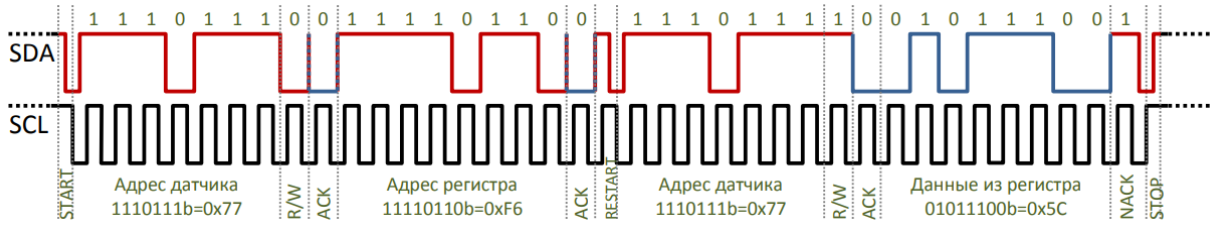

Рисунок 14. Пример чтения байта из регистра 0xF6 (датчик ответил значением 0x5C)

1.4. Акселерометр ADX-354
1.4.1. Общие сведения

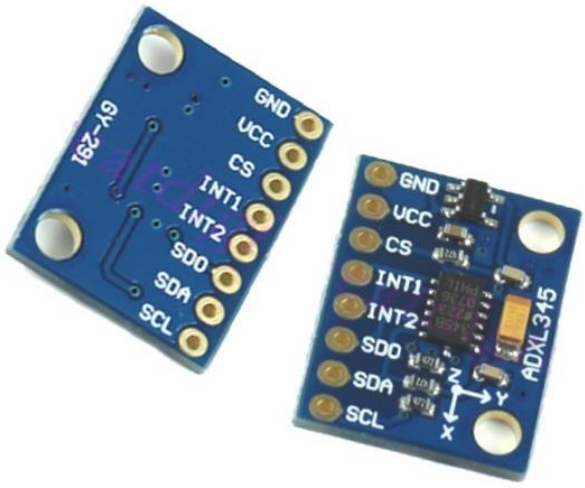

Рисунок 15. ADX-354

Акселерометр ADX-354 (рисунок 16) представляет собой датчик, использующийся для измерения проекции ускорения на пространственные оси. Определить ориентацию датчика-акселерометра в пространстве можно по данным измерения по трем осям и учитывая величину ускорения свободного падения (g).

Основные характеристики ADX-354 [13]:

- разрешение: 10 - 13 13bit (4 мг / LSB).
- тип шины: SPI (3- или 4-х проводный) и I2C.
- напряжение питания: 2-5V.
- диапазон температур: -40 до +85.
- диапазон измерений: + / - 16 г.
- потребление тока в рабочем режиме: 40 - 150 мкА.

Акселерометр ADX354 способен измерять ускорение в пределах до ±16 g и максимальным разрешением в 13 бит при частоте измерения вплоть до 3.2 кГц. Датчик имеет низкое энергопотребление (максимум 140 мкА), а напряжение питания может находиться в пределах 2-3,6 В.

Акселерометр имеет поддержку двух самых широко используемых интерфейсов - SPI и I2C, а также обладает 2 выходами прерываний и

встроенным буфером хранения измеренных данных. Схема модуля приведена на рисунке 17.

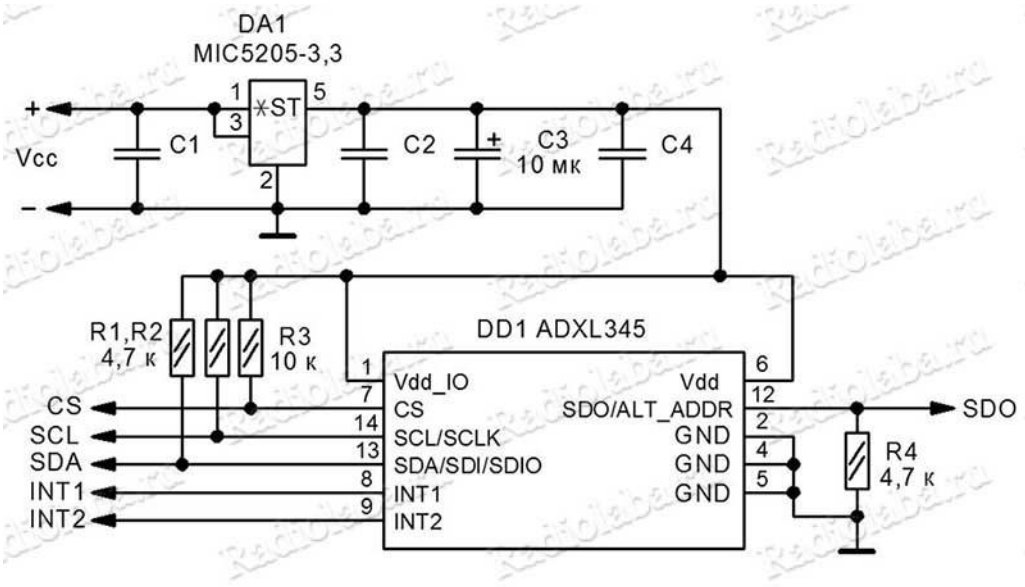

Рисунок 16. Принципиальная схема акселерометра ADX-354

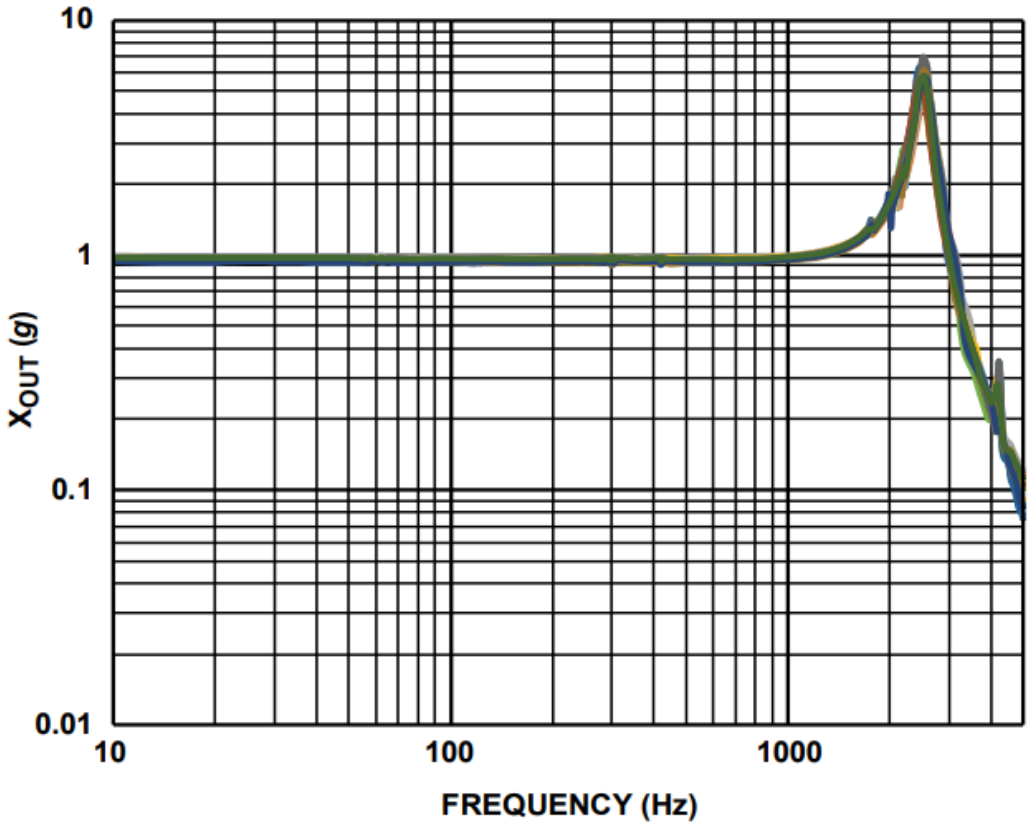

Рисунок 17. Пример реакции акселерометра ADX-354 на движение вдоль оси X

2. Протоколы передачи данных микроконтроллера ESP8266

Конструкция, которую большинство людей называют "Интернетом" - не просто один протокол: это целое множество уровней протоколов, часто называемый стеком TCP/IP. Мы рассмотрим эти различные уровни, чтобы понять, как ESP8266 взаимодействует с другими устройствами в сети.

Уровень OSI	Протокол
Прикладной	HTTP, MQTT, ESP-NOW, I2C
Транспортный	TCP, UDP
Сетевой	IP
Физический	Ethernet, Wi-Fi

Таблица 1. Протоколы обмена данных уровней OSI

2.1. Протокол Wi-Fi

2.1.1. Общие сведения

Wi-Fi – это технология беспроводной передачи данных, беспроводная локальная сеть (WLAN). В основу функционирования Wi-Fi заложена передача кодированных сигналов методом СВЧ-волн (сверхвысокочастотные волны) на сравнительно малые (несколько десятков метров) расстояния. Сеть построена минимум из 2 составляющих: точка доступа и клиент.

Точка доступа передает в эфир идентификатор (SSID, имя сети) передавая специальные пакеты данных около 10 раз в секунду со скоростью 100 Кбит/с. В принципе, это самая малая теоретическая пропускная способность информационного канала.

Устройство-клиент делает вывод о допустимости присоединения к беспроводной сети при попадании в её зону действия и обнаружении самого сигнала. Если же передатчик не будет трансировать идентификатор сети, то тогда сеть не будет обнаруживаться устройствами-клиентами. Осуществить подключение к ней возможно только введя SSID и пароль, если сеть защищена.

Точкой доступа в домашней беспроводной сети Wi-Fi обычно является беспроводной маршрутизатор – так называемый роутер. Роутеру и клиентам необходимо работать в одинаковом режиме (т.е. с идентичной частотой, модуляцией сигнала).

Маршрутизатор получает поток данных посредством сетевого кабеля, далее преобразовывает его и транслирует в виде сверхвысокочастотного

радиосигнала со своими параметрами. Приёмник улавливает эти волны и дешифрует их. Идентичным образом производится и передача информации.

Алгоритмы кодирования данных различаются в зависимости от версии стандарта протокола сети.

На сегодняшнее время беспроводные сети Wi-Fi функционируют преимущественно в 2 частотных диапазонах: 2,4 ГГц и 5 ГГц.

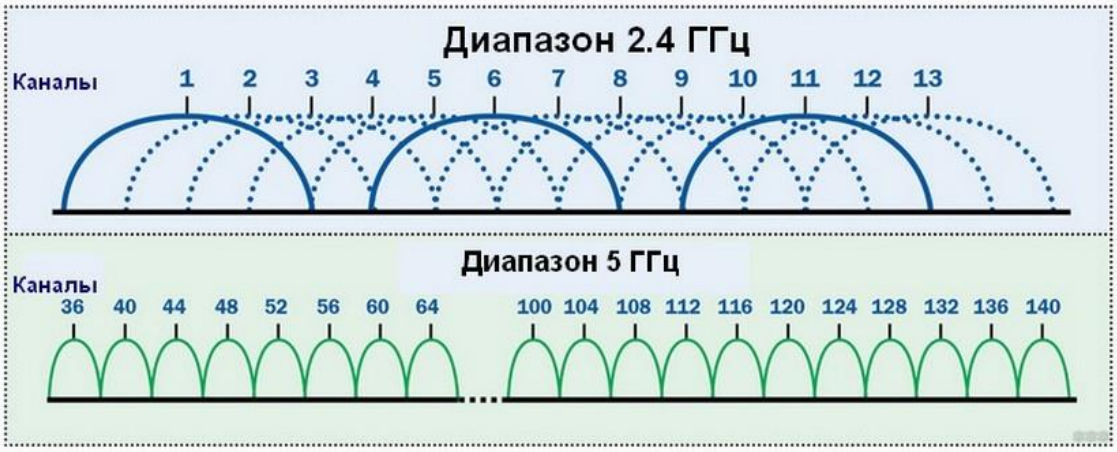

Рисунок 18. Частотный диапазон Wi-Fi

2.1.2. Стандарты протокола Wi-Fi

Стандарт	Диапазон	Максимальная скорость передачи, МБит/сек		Особенности
		канальная	реальная	
802.11a	5 ГГц	108	40	чувствителен к препятствиям
802.11b	2,4 ГГц	11	5	работают старые устройства
802.11g	2,4 ГГц	54	24	
802.11n	2,4 ГГц, 5 ГГц	150	55	при 1 антенне
		300	110	при 2-х антеннах
		450	165	при 3-х антеннах
		600	220	при 4-х антеннах
802.11ac	5 ГГц	433-6770	более 200	зависит от количества антенн.

Рисунок 19. Описание стандартов протокола Wi-Fi

802.11a – данный протокол является изначальной версии беспроводной передачи данных. Метод функционирования построен на базовой версии Wi-Fi, использовавшей стандартные методы кодирования стандарта. Отличием данной версии от изначальной является возможность функционирования на частоте 5 ГГц, что дало возможность передавать данные со скорость до 54 Мбит/с. Со стандартной частотой 2,4 ГГц данный протокол был несовместим.

802.11b – в данной версии было принято решение использовать изначальную частоту 2,4 ГГц из-за более широкой пропускающей способности. Скорость передачи информации находилась в пределах 5,5 – 11 Мбит/с.

802.11g – является доработанным стандартом 802.11b с возросшей скорость передачи (до 54 Мбит/с).

802.11n – один из наиболее совершенных протоколов. Радиосигнал способен проникать через толстые препятствия, например через бетонные преграды. Предоставляет возможность одновременной стабильной работы нескольких устройств-клиентов. Функционирует на обеих частотах 2.4 и 5 ГГц, обладает встроенной технологией MIMO, увеличившей скорость передачи до 150 Мбит/с..

802.11ac – самый современный и совершенный протокол передачи. Поддерживает только частоту 5 ГГц, из-за чего уменьшилась пропускная способность сети, что компенсируется встроенной технологией MIMO [15].

2.2. Протокол Ethernet

2.2.1. Общие сведения

Ethernet – протокол, отвечающий за передачу данных посредствам подключенного кабеля, использующая в основном в компьютерных и промышленных сетей. В модели OSI протокол Ethernet находится на канальном (LLC и MAC) и физическом уровнях.

Типы Ethernet [17]:

Название	Скорость	Кабель	Стандарт
Ethernet	10Mb/s	Толстый, тонкий коаксиал, Витая пара, оптика	802.3
Fast Ethernet	100Mb/s	Витая пара, оптика	802.3u
Gigabit Ethernet	1Gb/s	Витая пара, оптика	802.3z, 802.3ab
10G Ethernet	10Gb/s	Витая пара, оптика	802.3ae, 802.3an

Рисунок 20. Разновидности технологий Ethernet

Существуют 2 технологии Ethernet: классический и коммутируемый.

2.2.2. Классический Ethernet

Физический уровень включает в себя три варианта работы Ethernet, различающихся средой передачи данных:

- коаксиальный кабель
- витая пара
- оптоволокно

Канальный уровень, отвечает за методы доступа, а также протоколы, которые принципиально не имеют отличий для различных сред передачи информации. В классической технологии Ethernet одновременно присутствуют и подуровень LLC, и подуровень MAC.

MAC-адреса позволяют различать и определять устройства, присутствующие в сети Ethernet. Устройства не могут иметь одинаковый MAC-адрес, так как это приведет к некорректной работе сети и определяться будет лишь одно из этих устройств.

По типам MAC-адреса можно различить:

- Индивидуальные (для отдельных устройств).
- Групповые (для нескольких устройств).
- Широковещательные (для всех устройств, находящихся в сети).

Адреса могут быть локально распределены администратором сети, либо же центрально производителем сетевого оборудования. Форма кадра при использовании классического протокола Ethernet представлен на рисунке 21.

Рисунок 21. Формат кадра классического протокола Ethernet

Если принятый сигнал имеет отличия от переданного сигнала, то это может значить, что произошла так называемая коллизия (искажение данных). Для борьбы и контроля коллизий разработана технология CSMA.

Модель CSMA/CD приведена на рисунке 22:

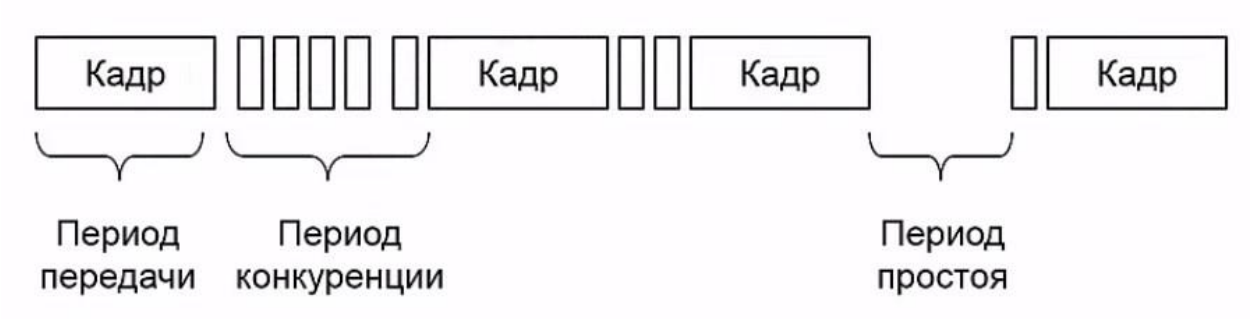

Рисунок 22. Формат кадра классического протокола Ethernet с учетом коллизии

2.2.3. Коммутируемый Ethernet

Коммутируемый Ethernet является самой оптимальной альтернативой, абсолютно исключающей возможность появления коллизий и связанных с ними проблем.

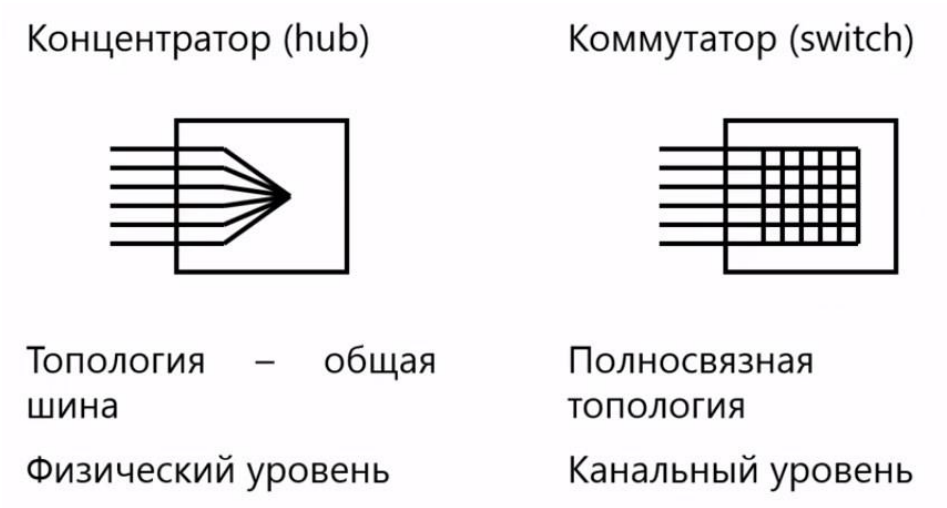

Рисунок 23. Концентратор и коммутатор протокола Ethernet

Принцип работы технологии коммутируемого Ethernet состоит в том, что вместо hub используется switch (коммутатор) – устройство, работающее на канальном уровне, которое обладает полносвязной топологией, позволяющей соединить все порты между собой напрямую. В каждом таком устройстве присутствуют таблицы коммутации, описывающие какое устройство должно быть подключено к данному порту свича.

Пример таблицы коммутации приведен на рисунке 24.

Порт коммутатора	MAC-адрес
1	1C-75-08-D2-49-45
2	00-02-B3-A7-49-D1
3	00-04-AC-85-E7-03

Рисунок 24. Таблица коммутации протокола Ethernet

Для получения MAC-адреса устройства используют алгоритм обратного обучения.

Данный алгоритм функционирует следующим образом: коммутатор принимает кадры и производит анализ заголовока, далее извлекает из него адрес устройства-отправителя кадра.

Для передачи данных используют алгоритм прозрачного моста.

Прозрачный мост не нуждается в настройки и получил свое название из-за своей незаметности для устройств в сети, так как он не имеет своего MAC-адреса. Коммутатор получает кадр и анализирует его заголовок, далее извлекает адрес устройства-получателя, которые соотносит с таблицей коммутации, определяя соответствующий порт подключенного устройства. Так кадр передается на определенный порт, а не на все порты, в отличии от технологии, использовавшей концентраторы. Если же адрес устройства не обнаружен в таблице коммутации, то коммутатор выполняет роль хаба.

2.3. Протокол IP

2.3.1. Общие сведения

Протокол IP обеспечивает передачу данных без гарантии доставки, а также порядка правильного следования сообщений, как в протоколе Ethernet. В протоколе IP используется технология передачи данных без прямой установки соединения. Проще говоря, пакет данных просто отправляется в сеть надеясь, что он дойдет до нужного устройства-получателя, а в случает, если данные не дошли до нужного адреса, то не происходит какого либо оповещения отправителя пакета или же попыток запросить данный пакет ещё раз. Принято, что протоколами на вышестоящих уровнях данная ошибка доставки должна быть разрешена.

Функцией технологии IP является присоединения в одну крупную объединенную сеть более мелких сетей, разработанных на основе различных методов и технологий канального уровня, значительно отличающихся между собой, в которой устройства способны свободно взаимодействовать между собой не взирая на различия их сетевых технологий.

Второй задачей является маршрутизация – нахождение оптимального маршрута посредством промежуточных узлов между отправителем и получателем в объединенной составной сети, а также обеспечение должного качества обслуживания.

2.3.2. Структура заголовка протокола IP

Рисунок 25. Структура заголовка протокола IP

Формат заголовка данных при использовании протокола IP приведен на рисунке 25.

Номер версии – на данный момент существует 2 версии протокола IP 4 и IP 6. Большинство устройств используют версию IP 4. Длина адреса в данной версии равняется 4 байтам. Главным недостатком является то, что адресов IP 4 уже крайне большое количество (более 4 миллиардов), из-за чего новых адресов не всегда хватает для подключенных устройств. На замену IP 4 разработана новая его версия – IP 6. В данной версии протокола длина IP адреса уже равняется 16 байт.

Длина заголовка - заголовок IP включает в себя некоторые обязательные поля, а также дополнительные поля, называемые опциями.

Тип сервиса – данное поле необходимо для обеспечения надлежащего качества обслуживания сети, однако на данный момент используется крайне нечасто.

Общая длина – она содержит длину всего пакета данных, то есть включает заголовок и данные. Максимальная длина данного пакета составляет 65 535 байт, однако на практике максимальный размер ограничен размером самого кадра канального уровня. В ином случае для передачи одного пакета данных необходимо будет несколько кадров канального уровня, что усложняет процедуру передачи.

Идентификатор пакета - для реализации фрагментации используются поля идентификатора пакета, флагов и смещения фрагмента.

Время жизни (Time To Live или TTL) - это максимальное время, в течении которого пакет данных способен перемещаться внутри сети. Данное понятие введено с целью предотвращения бесконтрольного перемещения пакетов по сети при возникновении в конфигурации сети ошибок.

Тип протокола - необходим для осуществления мультиплексирования и демультиплексирования, передачи посредством протокола IP данных от других различных протоколов следующего уровня. В данном поле указывается код протокола, находящегося на следующем уровне.

Контрольная сумма - необходима для проверки корректности доставки пакета. В ситуации, если во время проверки контрольной суммы обнаруживаются какие-либо ошибки, то данный пакет просто пропускается сетью и содержащаяся в нем информация отправителю пакета не доставляется. Контрольная сумма пересчитывается на каждом маршрутизаторе из-за того что данные в заголовке меняются и рассчитывается только по заголовку IP пакета.

IP адрес получателя и отправителя - в IPv4 длина IP адреса 4 байта (32 бита)

Далее в заголовке находятся необязательные его части (опции). Опций в заголовке IP может быть несколько, они могут обладать различным размером. Однако общая длина заголовка в протоколе IP обязана быть кратна 32, в этом случае, при необходимости, конец заголовка просто заполняют нулями до момента кратности 32 битам.

При установки опции, названной временные метки, каждый маршрутизатор записывает время прохождения пакета. Для диагностики работы сети используется опция под названием записать маршрут, при которой в IP пакет записывается адрес каждого маршрутизатора через которую он проходит.

Также есть опции, позволяющие отказаться от автоматической маршрутизации или задать маршрут отправитель:

Существует жесткая маршрутизация, где в пакете перечислены маршрутизаторы через которые пакету данных необходимо пройти. Также существуют свободные маршрутизации, при которых указываются лишь некоторые маршрутизаторы, через которые необходимо пройти, но в случае необходимости пакет способен пройти через любые иные маршрутизаторы.

2.4. Протокол TCP

2.4.1. Общие сведения

TCP (Transmission Control Protocol) является протоколом управления передач. TCP способен предоставить надежную доставку данных, в отличии от протокола UDP. TCP поддерживает надежную передачу потока байт (reliable byte stream), а также гарантию сохранения порядка следования сообщений и доставки данных.

Прокол TCP получает поток байт от приложения, которые далее одним большим массивом байт переходят на транспортный уровень.

Поток байт делится на отдельные части в протоколе TCP, называемые сегментами, каждый из которых отдельно отправляется получателю. Получатель принимает сегменты и пересобирает их в один массив байт, пример которого приведен на рисунке , который отправляет приложению.

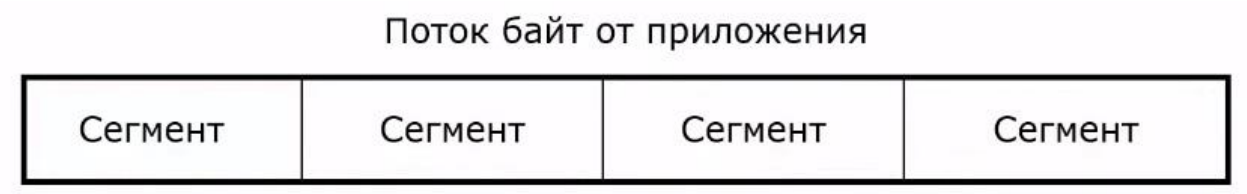

Рисунок 26. Поток байт в протоколе TCP

2.4.2. Протокол TCP: скользящее окно

Механизм работы по принципу скользящего окна подразумевает, что происходит подтверждение нескольких последовательных сегментов, а не каждого отдельного сегмента.

Варианты подтверждения приведены на рисунке 27 [20]:

- Остановка и ожидание (Wi-Fi, канальный уровень)
- Скользящее окно (TCP, транспортный уровень)

Рисунок 27. Варианты подтверждения доставки в протоколе TCP.

Приведем пример варианта остановки и ожидания. Отправитель останавливается ожидая подтверждение после передачи информации. Получатель отправляет подтверждение, после чего происходит передача следующей далее инфомации и цикл повторяется.

Рассмотрим второй вариант под названием скользящее окно (рисунок). При нем отправитель без ожидания подтверждения посылает одновременно несколько сегментов данных, а получатель отсылает лишь одно кумулятивное подтверждение. Оно означает, что были получены все сегменты данных.

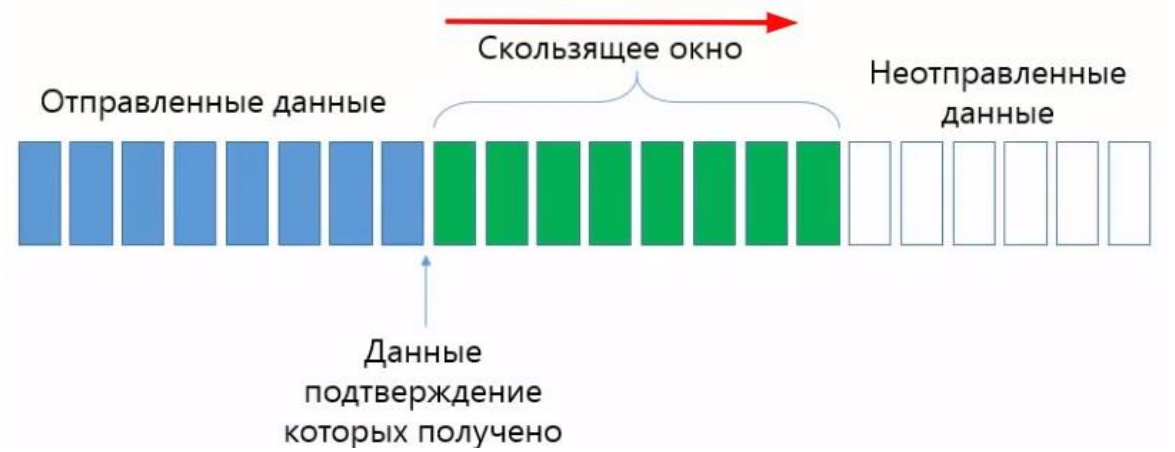

Рисунок 28. Пример подтверждения методом скользящего окна в протоколе TCP

В примере на рисунке 28 размер скользящего окна составляет 8 сегментов. При получении каждого подтверждения можно передвинуть окно далее по потоку данных, в результаты чего в него попадет следующая порция данных. Далее происходит отправка этих данных получателю, после чего отправитель останавливается и дожидается подтверждения.

2.5. Протокол UDP

2.5.1. Общие сведения

UDP – стандартный протокол передачи данных без установки предварительного соединения клиентов. Данный протокол специализирован для передачи так называемых датаграмм – это отдельные пакеты данных малого объема, которые передаются без проверки готовности приемного устройства, а также без подтверждения доставки и лишь в одном направлении. Клиенту необходимо лишь передавать данные, так как остальная техническая часть осуществляется автоматически с помощью библиотек реализации протокола.

Порядок передачи данных в протоколе TCP:

- происходит соединение;
- передаются данные;
- данные проверяются с помощью контрольных кодов;
- если необходимо, данные автоматически повторяются;
- соединение разрывается.

2.5.2. Формат заголовка протокола UDP

Формат заголовка в протоколе UDP приведен на рисунке 29 и состоит из 4-х полей [19]:

- Порт отправителя;
- Длина UDP, содержащая длину всей дейтаграммы UDP;
- Порт получателя;
- Контрольная сумма UDP, необходимая для осуществления корректной доставки дейтаграммы.

16 бит Порт отправителя	16 бит Порт получателя
16 бит Длина UDP	16 бит Контрольная сумма UDP

Рисунок 29. Формат заголовка протокола UDP

Длина заголовка протокола UDP составляет от 8 байт (только заголовок) до 65 515 байт (максимальная длина данных IP-пакета)

В отличии от протокола TCP, UDP не дает гарантий, что данные будут доставлены, пакеты будут доставляться в той же последовательности, как при передаче и что придут правильные данные. Только заголовок UDP-пакета защищен контрольным кодом. Протокол лишь дает гарантию, что данные не попадут по неверному адресу. Клиенту необходимо самостоятельно заботиться о целостности информации в пакетах.

Не смотря на недостатки, главное преимущество передачи UDP-данных состоит в том, что отправка и получение происходит значительно быстрее, чем через TCP. По этой причине протокол UDP распространен шире.

2.6. Протокол HTTP

2.6.1. Общие сведения

HTTP – является высокоуровневым протоколом уровня приложений, использующийся для передачи гипертекстовых документов.

Информация в нем разбита на мелкие фрагменты, между которыми присутствуют видимые ссылки, позволяющие переходить между фрагментами. При организации информации подобным образов, протокол HTTP может обеспечить возможность просмотра текста. Однако на данный момент, по этому протоколу уже передается практически любая информация, в том числе мультимедиа (видео, фото). Можно заключить, что данный протокол уже используется для передачи гипермедиа данных.

В настоящее время протокол HTTP является основным для использования сервисом World Wide Web (Всемирная Сеть) для получения различных данных WEB-сайтов.

Особенности протокола HTTP:

- HTTP – протокол уровня приложений, в нем приложение на устройстве осуществляет обмен данными с приложением на удаленном сервере.
- Основной объект для HTTP протокола – это ресурс (URI – Uniform Resource Identifier), который задается в запросе клиента. Обычно ресурсом являются файлы, расположенные на сервере, но может быть что угодно, даже нечто абстрактное. Для идентификации ресурсов используются глобальные идентификаторы URI, т.е. адреса Интернета.
- HTTP функционирует по технологии клиент-сервер. То есть, клиент посылает запрос и получает в ответ информацию с сервера.
- Сама текстовая информация обычно передается на языке гипертекстовой разметки HTML.
- Обмен данными происходит через TCP-соединение.
- Запрос будет формировать браузер, отвечать – удаленный сервер.

2.6.2. Структура протокола HTTP

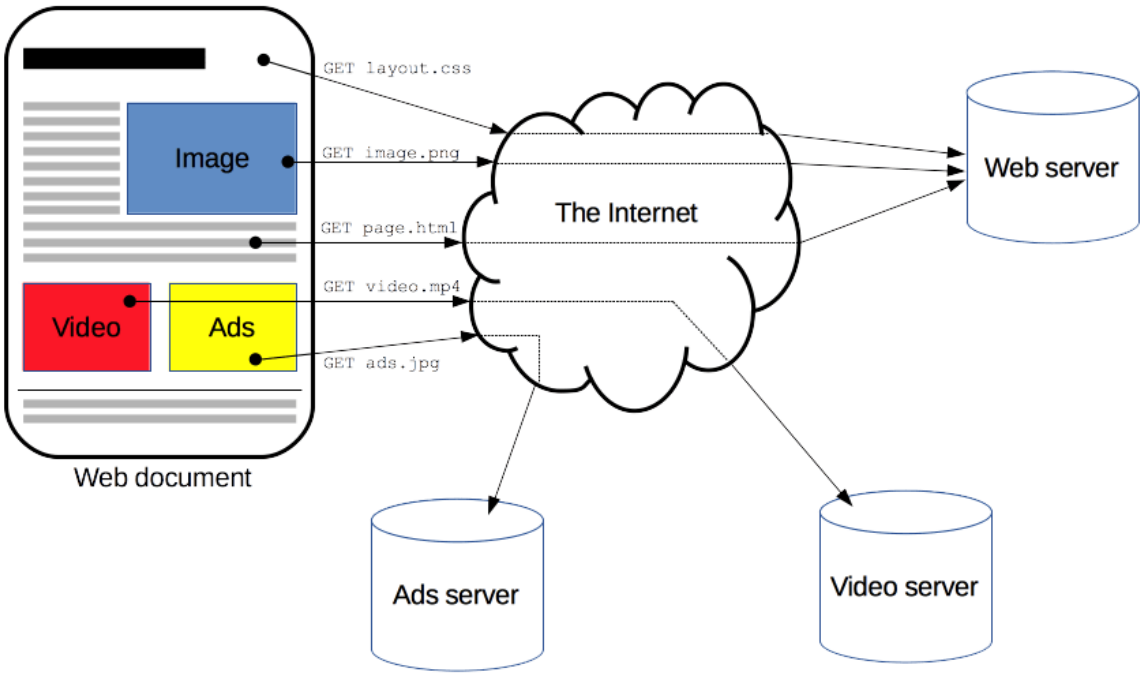

Рисунок 30. Структура протокола HTTP

Протокол HTTP состоит из 2 элементов: клиента и сервера. Клиент отправляет запросы и ожидает в ответ данные от сервера. Сервер в свою очередь ждёт, очередного запроса от клиент, который обрабатывает его и возвращает ответ обратно.

Для отправки HTTP-запроса, необходимо воспользоваться URL-адресом (Uniform Resource Locator) - унифицированный указатель ресурса. Он указывает приложению (веб-браузеру), что для работы необходиво воспользоваться HTTP-протоколом и далее получить в качестве ответа веб-файл с данного адреса обратно.

На данный момент URL-адреса стандартно начинаются с префикса http:// или https://.

Веб-браузер отправляет введённые пользователем в поисковую строку слова (например Google, Yandex) в DNS-преобразователь (Domain Name System - «доменная система имён») URL-адресов, являющийся по сути таблицей, содержащей все множество зарегистрированных имен и IP-адресов сайтов. DNS в ответ возвращает веб-браузеру IP-адрес, далее браузер приступает к составлению HTTP-запроса уже с вложенным непосредственно в него IP-адресом.

После процесса получения и обработки HTTP-запроса сервер формиует ответ, которые отправляется клиенту. Данный ответ содержит запрашиваемые данные, а также дополнительную информацию (метаданные).

В момент получения веб-браузером ответа, он производит отображения веб-сайта посредством использования внутреннего ПО (так называемого движка). В этот момент процедура обмена HTTP-запросами клиент-сервер заканчивается и пользователь получает необходимые ему данные.

2.7. Различия протоколов TCP и HTTP модели OSI

2.7.1. Выводы по протоколу TCP

Из данных, представленных в предыдущих пунктах, может создаться впечатления о идентичности проколов TCP и HTTP. Во избежание данного заблуждения следует прояснить некоторые детали.

TCP - аббревиатура от протокола управления передачей. Это стандартный протокол связи, который позволяет различным сетевым устройствам и прикладным программам взаимодействовать друг с другом по сети. Это

протокол, ориентированный на подключение, который требует установления соединения между клиентом и сервером перед передачей пакетов данных.

Более того, TCP является протоколом с отслеживанием состояния, и Целевая группа по разработке Интернета (IETF) включила TCP в свои стандарты. Этот протокол делит все сообщение на множество меньших пакетов данных и передает эти пакеты от источника к получателю. Все эти пакеты повторно собираются в пункте назначения, чтобы сформировать полное сообщение.

TCP хорошо работает с Интернет-протоколом (IP), и вместе они определяют правила Интернета. Кроме того, это безопасный и надежный протокол, который обеспечивает целостность данных, передаваемых по сети.

В результате многие протоколы высокого уровня, требующие передачи данных, используют TCP. Кроме того, для отправки и получения электронной почты по протоколу доступа к интернет-сообщениям (IMAP), протоколу простой передачи почты (SMTP) и протоколу почтового отделения (POP) также требуется протокол TCP. Другое основное применение TCP можно увидеть в веб-доступе через HTTP.

Особенности TCP [30]:

- Протокол транспортного уровня: TCP - это протокол транспортного уровня, который обеспечивает передачу данных между источником и пунктом назначения.
- Ориентированный на подключение: TCP - это протокол, ориентированный на подключение, что означает, что он требует, чтобы клиент и сервер установили соединение между собой перед передачей данных.
- Полнодуплексный: TCP поддерживает полнодуплексную связь, т.е. поддерживает передачу данных в обоих направлениях одновременно.
- Потоково-ориентированный: TCP позволяет отправителю передавать данные в виде потока байтов, а также получателю принимать данные в той же форме. Следовательно, это потоково-ориентированный протокол.
- Контроль ошибок: TCP предоставляет механизм контроля ошибок. Это помогает идентифицировать поврежденные сегменты данных, потерянные сегменты, дублированные сегменты и сегменты, вышедшие из строя, и обеспечивает надежную доставку данных. Кроме того, он использует три простых метода для выполнения контроля ошибок, а именно контрольную сумму, подтверждение и повторную передачу.

- Управление потоком: Механизм управления потоком управляет объемом данных, которые отправитель передает получателю, чтобы сторона получателя не была перегружена.

Преимущества TCP:

- TCP является надежным протоколом, поскольку он обеспечивает механизмы контроля ошибок и управления потоком.
- TCP гарантирует, что пакеты данных прибудут к месту назначения в том же порядке, в каком их передает отправитель.
- TCP присваивает уникальный IP-адрес каждому устройству в сети. Это позволяет идентифицировать все устройства в сети.
- Получатель отправляет подтверждение отправителю независимо от того, получил он пакет данных или нет. В случает неполучения, отправитель повторно передает этот пакет данных.
- Это открытый протокол, который никому не принадлежит.

Недостатки TCP:

- Протокол не идеален для малых сетей с небольшим количеством ресурсов, например, локальные сети или персональные вычислительные сети.
- С момента его создания в этот протокол не вносились никакие изменения.
- TCP работает медленнее, чем протокол пользовательских дейтаграмм (UDP).

2.7.2. Выводы по протоколу HTTP

HTTP - аббревиатура от протокола передачи гипертекста. Это протокол типа запрос-ответ, который позволяет получать доступ к данным во Всемирной паутине (WWW). Другими словами, это основной протокол передачи данных для Всемирной паутины (WWW).

Всемирная паутина (WWW) относится к связи между веб-клиентами и веб-серверами. Веб-клиенты и серверы взаимодействуют друг с другом, отправляя HTTP-запросы и получая HTTP-ответы. Здесь веб-клиентами в основном являются браузеры, такие как Google Chrome, Safari, Edge и т.д. Однако ими также может быть любое устройство или прикладная программа. С другой стороны, веб-серверы - это компьютерные системы в облаке.

В качестве альтернативы мы можем определить HTTP как протокол передачи данных, который пользователи используют для загрузки веб-страниц с использованием гипертекстовых ссылок. Он предоставляет пользователям возможность получать доступ к веб-ресурсам или взаимодействовать с ними путем передачи гипертекстовых сообщений между клиентом и сервером.

Более того, HTTP - это протокол без состояния. Это означает, что получатель не сохраняет информацию о сеансе из предыдущего запроса. HTTP определяет правила для того, как веб-браузеры и веб-серверы должны взаимодействовать. Он также определяет, как веб-серверы должны реагировать на конкретный HTTP-запрос.

Особенности HTTP [30]:

- HTTP - это модель запрос-ответ, которая позволяет браузерам взаимодействовать с веб-серверами.
- Пока клиент и сервер знают, как обращаться с содержимым данных, они могут обмениваться данными любого типа. Следовательно, этот протокол не зависит от типа информации.
- Это протокол без состояния, т.е. каждый запрос клиента является новым запросом. Получателю не обязательно сохранять информацию о сеансе из предыдущего запроса.
- HTTP - это протокол, не требующий установления соединения. Клиент и сервер взаимодействуют только во время текущего запроса и ответа.

Преимущества HTTP

- Как и TCP, HTTP также использует схему адресации.
- Перегрузка сети меньше, поскольку требуется меньше TCP-подключений.
- HTTP обеспечивает меньшую задержку между последующими запросами, поскольку отсутствует “рукопожатие”. Оно происходит на начальном этапе установления соединения.
- Он способен сообщать об ошибках без закрытия TCP-соединения.
- Сравнительно малое использование ресурсов процессора и памяти.

Рукопожатие SSL/TLS — это название этапа установки HTTP-соединения.

Недостатки HTTP

- HTTP не идеален для соединений "точка-точка".
- Это не идеально подходит для сотовых телефонов.
- HTTP менее безопасен, поскольку он не задействует никаких методов шифрования.
- Протокол не обеспечивает надежного обмена данными.
- До тех пор, пока клиент не получит все данные с сервера, соединение не прекратится. Следовательно, сервер не будет доступен для других клиентов до тех пор, пока клиент не разорвет соединение.

2.7.3. Сравнение протоколов TCP и HTTP

Выбор между TCP и HTTP на самом деле был бы неуместен, поскольку оба этих протокола связи имеют свои собственные цели. TCP - это протокол, ориентированный на подключение, который обеспечивает передачу данных между клиентом и сервером.

С другой стороны, HTTP - протокол запроса-ответа, который требует, чтобы клиент и сервер установили TCP-соединение для обмена запросом и ответом. Этот процесс включает в себя серию сеансов, в ходе которых клиент отправляет запрос серверу, а сервер отвечает на этот запрос запрашиваемой информацией.

Следовательно, нет выбора между TCP и HTTP, поскольку HTTP полагается на TCP для установления соединения между клиентом и сервером.

В таблице 2 приведена сравнительная характеристика протоколов TCP и HTTP по основным параметрам.

Таблица 2.

TCP	**HTTP**
TCP расшифровывается как протокол управления передачей.	HTTP расшифровывается как протокол передачи гипертекста.
Это протокол транспортного уровня.	Это протокол прикладного уровня.
TCP не требует какого-либо порта для осуществления передачи данных.	HTTP использует порт с номером 80. Это порт, который сервер обычно прослушивает или ожидает получить что-либо от клиента.
Этот протокол работает относительно медленнее, чем HTTP.	HTTP работает быстрее, чем TCP.
Целью TCP является обеспечение возможности передачи данных между исходным хостом и конечным хостом.	Этот протокол позволяет пользователям искать и извлекать нужные ресурсы в Интернете.
Протокол с отслеживанием состояния.	Протокол без состояния.
TCP используется в различных других протоколах, таких как HTTP, FTP, HTTPS, SMTP и Telnet.	HTTP широко используется для веб-приложений.

2.8. Протокол MQTT
2.8.1. Общие сведения

MQTT является протоколом передачи данных методом издатель-подписчик (publisher/subscriber).

Протокол создан крайне простым и экономным с учётом его суровых условий эксплуатации (одним из первых его применений было обеспечение контакта фрагментов нефтепровода с центральными звеньями используя спутниковую сеть). Он отлично подходит для устройств с ограниченным временем автономной работы или устройств малой мощности (датчики, сенсоры, камеры, телефоны).

Протокол MQTT используется для потоковой передачи данных для сетей с низкой пропускной способностью, непредсказуемой стабильностью или высокой задержкой, а также между устройствами с ограниченным временем автономной работы и малой мощностью CPU.

На данный момент протокол MQTT является идеальным транспортным протоколом в IoT. Он построен на базе протокола TCP/IP, однако в нем присутствует ответвление для работы по Bluetooth, UDP, ZigBee и в других сетях IoT, отличных от TCP/IP по названием MQTT-SN .

MQTT обладает следующими основными свойствами [25]:

- Полагается на TCP/IP для базовых задач связи
- Идеально подходит для распределённых коммуникаций «один ко многим» и разъединённых приложений
- Разработан для доставки сообщений по шаблонам «максимум один раз», «минимум один раз» и «ровно один раз»
- Нейтрален к содержимому сообщения
- Оснащён функцией LWT (Last Will and Testament, «последняя воля и завещание») для уведомления сторон об аномальном отключении клиента

Участник сети протокола MQTT может сам осуществлять функцию издателя, потребителя либо обе одновременно. Отличительной характеристикой протокола MQTT является особое понимание каналов: каждый из них обрабатывается как путь к файлу

2.8.2. Принцип работы протокола MQTT

Система обмена данными, осуществляемая протоколом MQTT, состоит из сервера-издателя и сервера-брокера, а также клиентов, которых может быть один или несколько. Издатель (pub) не нуждается в каких-либо настройках по числу либо расположению подписчиков (sub), которые ожидают сообщения. Может существовать несколько брокеров, распространяющих сообщения в системе MQTT. Подписчикам же не нужна никакая настройка на определенного издателя.

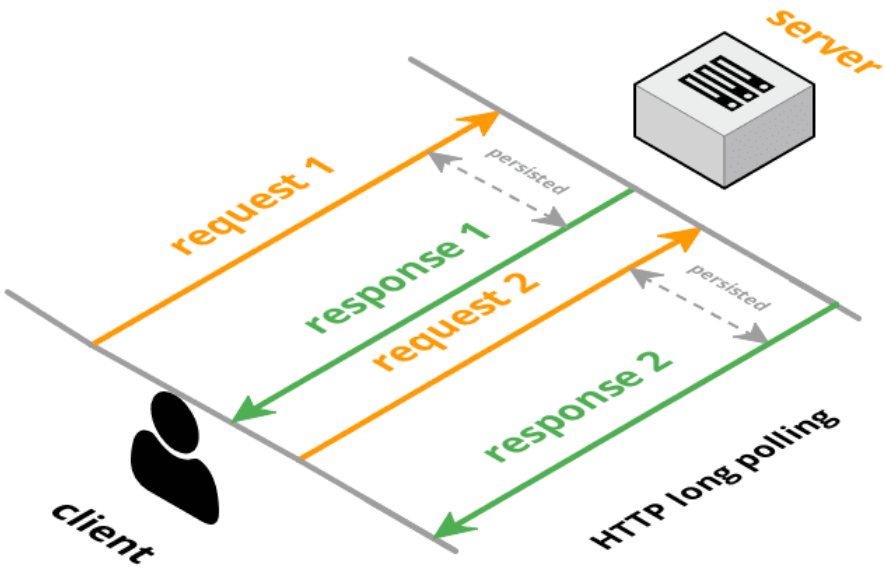

Рисунок 31. Структура протокола MQTT

Протокол MQTT осуществляет метод постройки целой иерархии каналов связи, пример такой иерархии приведен на рисунке . В каждый момент наличия у издателя новых данных для отправки клиентам, сообщение дополняется наличием контроля доставки. Клиенты на более высоком уровне способны получать каждое переданное сообщение, а клиенты низкого уровня могут лишь получать данные, которые относятся только к 1-2 базовым каналам, находящихся на нижнем уровне иерархии. Такая конструкция позволяет облегчить передачу информации с размером от 2 байт до 256 мегабайт.

Поскольку MQTT является бинарным протоколом, вся информация в сети, переданная или полученная брокером протокола MQTT, будет закодирована в двоичном формате. Это значит, чтобы получить исходные данные, необходимо декодировать сообщение.

Брокеры протоколы MQTT способны сообщения, которые связанные с каналами, не имеющих в данный момент подписчиков. В этой ситуации все

сообщения будут сохранены либо же отброшены, это зависит от полученных в управляющем сообщении инструкций. Данная процедура необходима например, если новым подключенным абонентам необходима получить последнюю записанную точку данных без ожидания следующей отправки сообщений.

Следует отметить, что протокол MQTT осуществляет передачу учётных данных безопасности открытым текстом, в ином случае не поддерживается аутентификация или функции безопасности. В данной ситуации важную роль осуществляет использование фреймворка SSL, который помогает защищать передаваемые данные от несанкционированного перехвата или подделки.

Два важных компонента, которые содержатся в каждом сообщении протокола MQTT:

- **Байт 1**: содержит флаг дублирования, инструкции для сохранения сообщений и информацию об уровне качества обслуживания (QoS), а также тип сообщения (запрос от клиента на доступность подключения, запрос ping, подтверждение подписки и т. д.).
- **Байт 2**: содержит информацию об оставшейся длине сообщения, полезной нагрузке и прочих данных, содержащихся в заголовке.

Флаг по название QoS, содержащийся в байте 1 предлагается рассмотреть подробнее, так как он лежит в основе переменной функциональности, поддерживаемой протоколом MQTT. QoS-флаги содержат в себе значения, которые основаны на срочности и характере сообщения:

- 0 - не более одного раза: сервер сначала срабатывает и далее забывает. При этом сообщения могут быть продублированы или потеряны.
- 1 - по крайней мере один раз: получатель должен подтвердить доставку сообщения. Возможно дублирования сообщения, однако его доставка гарантирована
- 2 - ровно один раз: сервер должен обеспечить доставку данных. Все сообщения поступают без потери или дублирования и только один раз.

2.9. Протокол ESP-NOW

2.9.1. Основные сведения

Протокол ESP-NOW является альтернативным высокоуровневым (таблица 1) протоколом связи WiFi между парами сопряженных устройств посредством передачи малых пакетов данных. Ускоряет процесс обмена пакетами то, что дополнительные процедуры, связанные с реализацией работы протокола Wi-Fi не производятся.

ESP-NOW реализует такие функции, как смешанные зашифрованный или незашифрованный обмен данными между парами сопряженных устройств; возможность установкаи функции обратного вызова для отправки данных на прикладной уровень, например информации о сбое в обмене данных; передача до 250 байт полезной информации.

Ограничения и особенности протокола ESP-NOW:

- Протокол WiFi не используется
- Требуется только начальное сопряжение.
- Соединение не разрывается после сопряжения.
- На одном устройстве поддерживаются максимум 20 пар, включая зашифрованные. 6 пар поддерживается в режиме SoftAP или SoftAP + Station.
- Скорость передачи на частоте 2,4 ГГц составляет менее 1 Мбит/с, Это значит, что протокол ESP-NOW функционирует в тех же канал и частотах, что и протокол WiFi.
- Не поддерживается шифрование многоадресной рассылки.
- Поддерживается только множественная раздача сопряженным парам устройств. Широковещательная передача не поддерживается.

2.9.2. Принцип работы протокола ESP-NOW

ESP-NOW поддерживается связанный список на нижнем уровне протокола, он содержит данные о сопряженном и локальном устройствах, включая их MAC-адреса. Протокол также сохраняет часто используемые данные для прикладного уровня, для избежания дополнительных расходов из-за повторной обработки связанного списка. Для отправки и получения данных используются данные об подключенных устройствах, они состоят из:

1) Данные о сопряженном устройстве:
   - LMK - это локальный мастер-ключ размером в 16 байт, необходимый для шифрования полезной части данные в течении связи в паре.
   - MAC-адрес - адрес сопряженного устройства размером в 6 байт, который совпадает с адресом устройства-отправителя.
   - Режим - режим локального устройства (1 байт) который устанавливает тип передающего интерфейса (например STA или SoftAP) протокола ESP-NOW. Режим сопряженного устройства не влияет на какую-либо функцию, а только сохраняет информацию о режиме для прикладного уровня. При работе в режиме STA WiFi может применяться Station и SoftAP. В режиме WiFi — только SoftAP.
   - Канал - канал в 1 байт, позволяющий вести обмен пакетами данных между устройствами, сопряженными в пару. Он не влияет ни на какие данные, а лишь хранит информацию для прикладного уровня о канале. Значение определяется прикладным уровнем и варьируется от 0 до 255. К примеру, 0 означает, что канал сейчас не определен, а значение 1-14 имеют действующие каналы. Функции, которые определены прикладным уровнем, могут быть назначены остальными значениями.

2) Данные о локальном устройстве:

   - PMK - главные мастер-ключ размера 16 байт, необходимый для шифрования на присоединенном устройстве (методом KOK в API) ключа. Протокол ESP-NOW изначально имеет поддержку PMK, из-за чего никакая начальная настройка не требуется.
   - Режим - режим (1 байт) локального устройства устанавливающий передающий WiFi интерфейс (SoftAP или STA) ESP-NOW.

Формат пакета ESP-NOW приведен на рисунке 32.

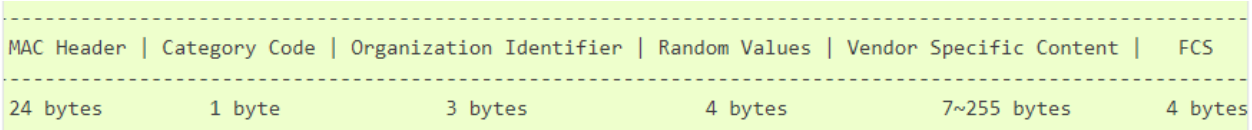

Рисунок 32. Формат пакета в протоколе ESP-NOW

Пакет содержит:

- 24 байта - заголовок MAC-адреса.
- 1 байт - категория, определяющая категорию создателя пакета.

- 3 байта - ID организации, являющимся уникальным идентификатором
- 4 байта - случайное значение, необходимое для защиты передаваемых данных.
- 7-255байт - данные создателя отправляемого пакета.
- 4 байта – FCS, контрольная сумма.

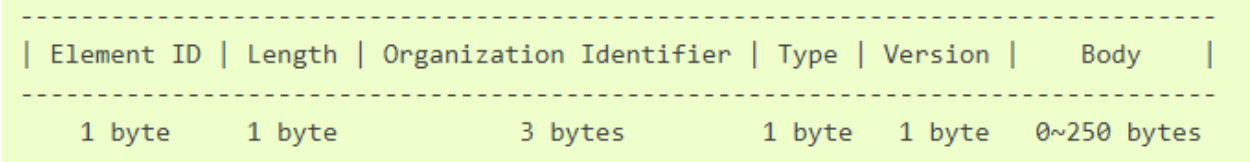

Рисунок 33. Формат кадра данных создателя

Формат данных создателя пакета приведен на рисунке 33. Они содержат следующие поля:

- 1 байт – ID-идентификатор.
- 1 байт – длина - общая длина строки, содержащей ID организации, типа, версии и пользовательских данных.
- 3 байта - ID организации, имеющая свой уникальный идентификатор, который является первыми тремя байтами MAC-адреса.
- 1 байт - тип протокола ESP-NOW.
- 1 байт - текущая версия протокола ESP-NOW.
- 0-250 байт - пользовательские данные.

3. Исследование применимости протоколов различных уровней к устройствам на базе лабораторного макета

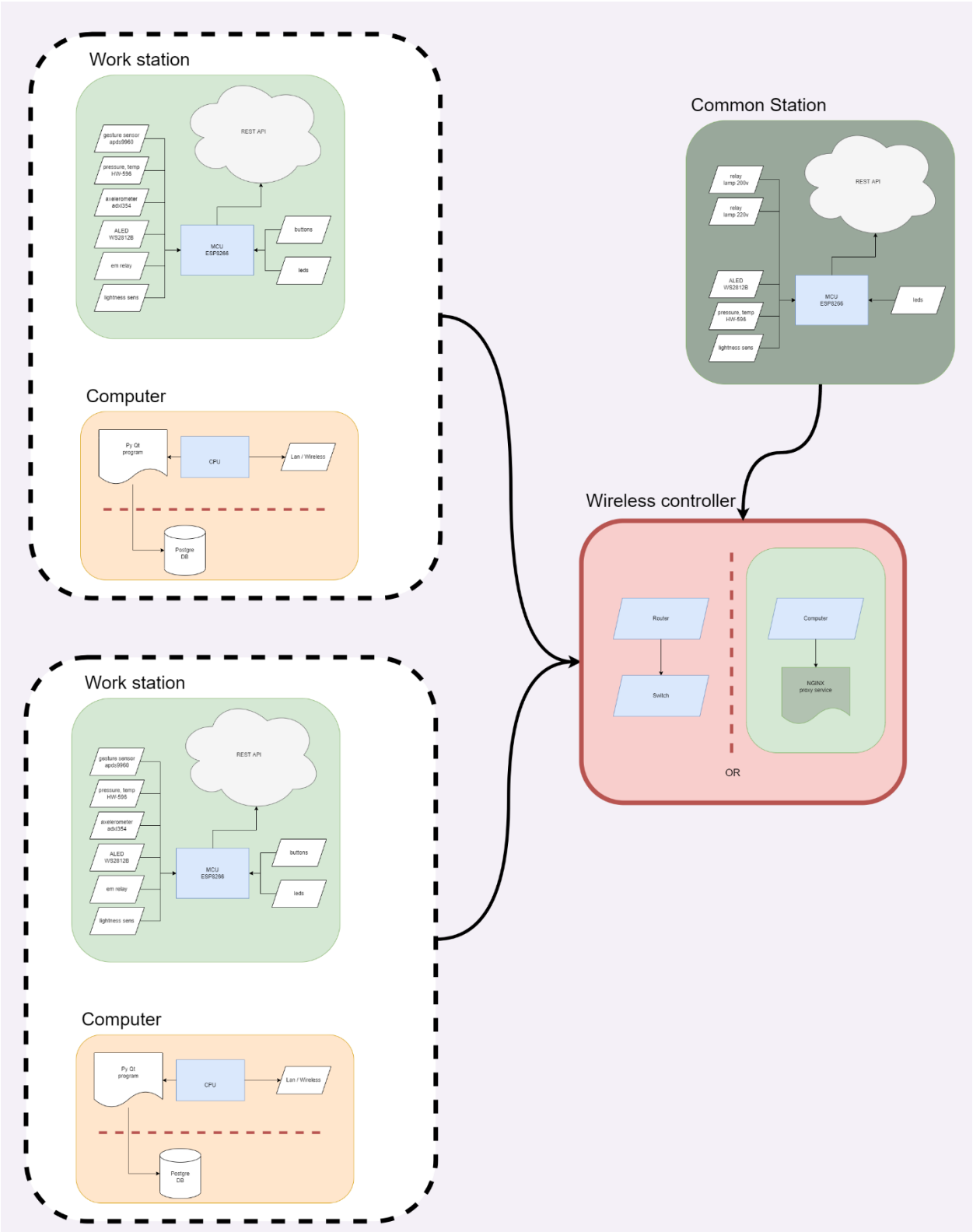

Рисунок 34. IoT-диаграмма лабораторного макета

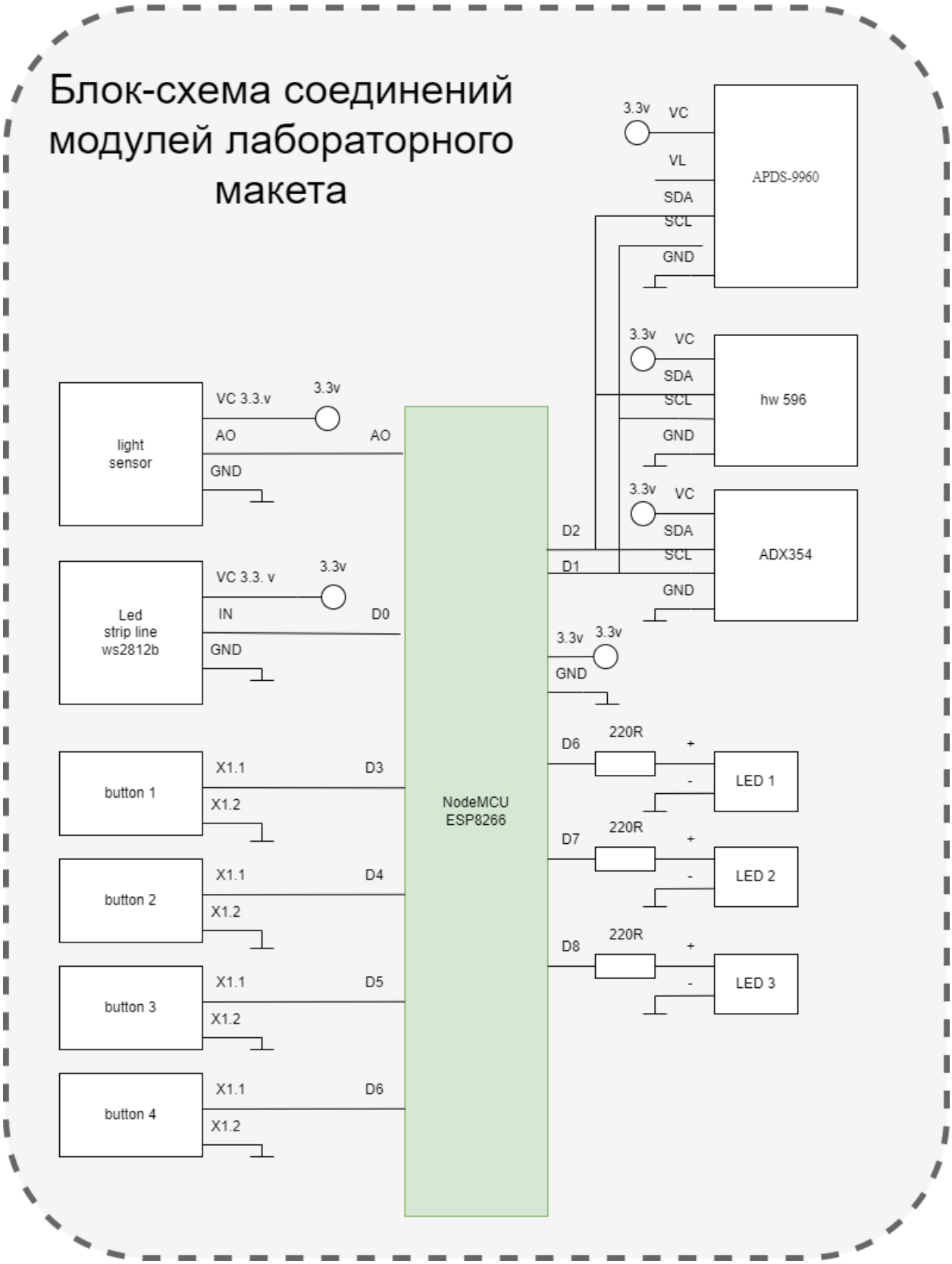

Рисунок 35. Блок-схема лабораторного макета

На рисунке 34 приведена IoT-диаграмма лабораторного макета. Она дает представление о методах обмен данными составляющих установки. Как можно наблюдать, процесс передачи данных от лабораторного макета осуществляется беспроводным методом. Информация с нескольких установок попадает на локальный сервер, доступ к которому через маршрутизатор имеют персональный компьютеры внутри локальной беспроводной Wi-Fi сети.

На рисунке 35 показана схема соединения устройств макета. Рассмотрим на её базе применение протоколов различных уровней.

3.1. Физический уровень

Первый уровень OSI описывает физическую среду, необходимую для передачи необработанных двоичных данных между узлами.

В нашем случае, в макете на физическом уровне используется протокол Wi-Fi. Он позволяет осуществлять беспроводную передачу данных с установки на сервер, к которому могу получить доступ остальные устройства сети. Стандарт протокола Wi-Fi рекомендуется выбирать исходя из количества экспериментальных установок и прочих подключаемых устройств для просмотра полученных данных (персональных компьютеров, смартфонов). Наиболее подходящим под данные задачи является стандарт 802.11n, работающий на частоте 2.4 ГГц. Он обеспечивает наиболее стабильное соединения в сочетании с высокой скоростью при достаточно большом количестве подключенных устройств (Рисунок 19. Описание стандартов протокола Wi-Fi).

Также можно рассмотреть применение технологии Fast Ethernet (Рисунок 20. Разновидности технологий Ethernet) для передачи данных на физическом уровне. Это позволит кратно увеличить количество подключаемых устройств без потери скорости соединения. Однако главным недостатком является необходимость подключения всех устройств к сети посредством кабеля, что увеличит материальные затраты, а также создаст физические неудобства, возрастающие с увеличением количества подключенных устройств.

Итоговая сравнительная характеристика применения протоколов Wi-Fi и Ethernet представлена на таблице 3.

Таблица 3

	Wi-Fi 802.11n, 1 антенна	Fast Ethernet
Тип соединения	Беспроводной	Проводной
Скорость передачи	55 Мб/с	100 Мб/с
Количество устройств	100	Неограниченно
Тип устройств	Любой с поддержкой беспроводной сети	Имеющий LAN-порт
Дальность связи	150 м*	Ограничена длиной кабеля

*- в зависимости от строения помещения

3.2. Сетевой уровень

Этот уровень позволяет устройствам, расположенным в разных сетях, взаимодействовать друг с другом. Сетевой уровень использует адреса и маршрутизацию пакетов для обеспечения того, чтобы правильное сообщение достигло нужной стороны.

Наиболее подходящим рассматривается применения протокола IPv6. Данный протокол является улучшенной версией протокола IPv4. Он обеспечивает более эффективную маршрутизацию, т.к. уменьшает размер таблицы маршрутизации, обладает более простой формой заголовка и обеспечивает большую полезную нагрузку, чем IPv4.

Главное различие протоколов - это структура IP-адреса. IPv4 использует четыре однобайтовых десятичных числа, разделенных точкой (111.111.0.1). IPv6 — шестнадцатеричные числа, разделенные двоеточиями (2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d)

Главным недостатком протокола IPv6 является одновременно его главной особенностью – это большая длина адреса, а также отсутствие совместимости с устройствами, поддерживающими протокол IPv4.

Главной причиной использования технологии IPv6 является проблема глобальной нехватки IP-адресов, использующих протокол IPv4.

Итоговая сравнительная характеристика протоколов IPv4 и IPv6 приведена в таблице 4.

Таблица 4.

	IPv4	IPv6
Длина адреса	32 бита	128 бит
Формат поля	12 заголовков	8 заголовков
Контрольная сумма	Есть	Отсутствует
Настройка адреса	Ручная	Автоматическая
Публичный адрес	Уникальный	Юникаст-адреса

Юникаст (Unicast) - одноадресная передача, при которой IP-адрес связан только с одним конкретным узлом в сети. Это также называют взаимно-однозначным соответствием.

3.2.1. Исследование времени отклика сети с использованием протоколов IPv4 и IPv6

Для исследования зависимости времени отклика от размера пакета в IPv4 и IPv6 воспользуемся программой ping и ping6 соответственно, они позволяют самостоятельно задавать размер пакета.

Для измерения времени отклика протокола IPv4 воспользуемся командой ping -s 1500 192.168.254.198;

для измерения времени отклика протокола IPv6 воспользуемся командой ping6 -s 1500 2012::2.

Результаты измерений приведены в таблице 5 и проиллюстрированы на рисунке 36.

Таблица 5

	Размер пакета, бит									
Протокол	64	128	256	512	1024	1280	1518	3000	6000	9000
IPv4, мс	0,24	0,25	0,25	0,26	0,35	0,39	0,45	0,72	1,27	1,86
IPv6, мс	0,24	0,24	0,25	0,25	0,33	0,37	0,42	0,71	1,26	1,85

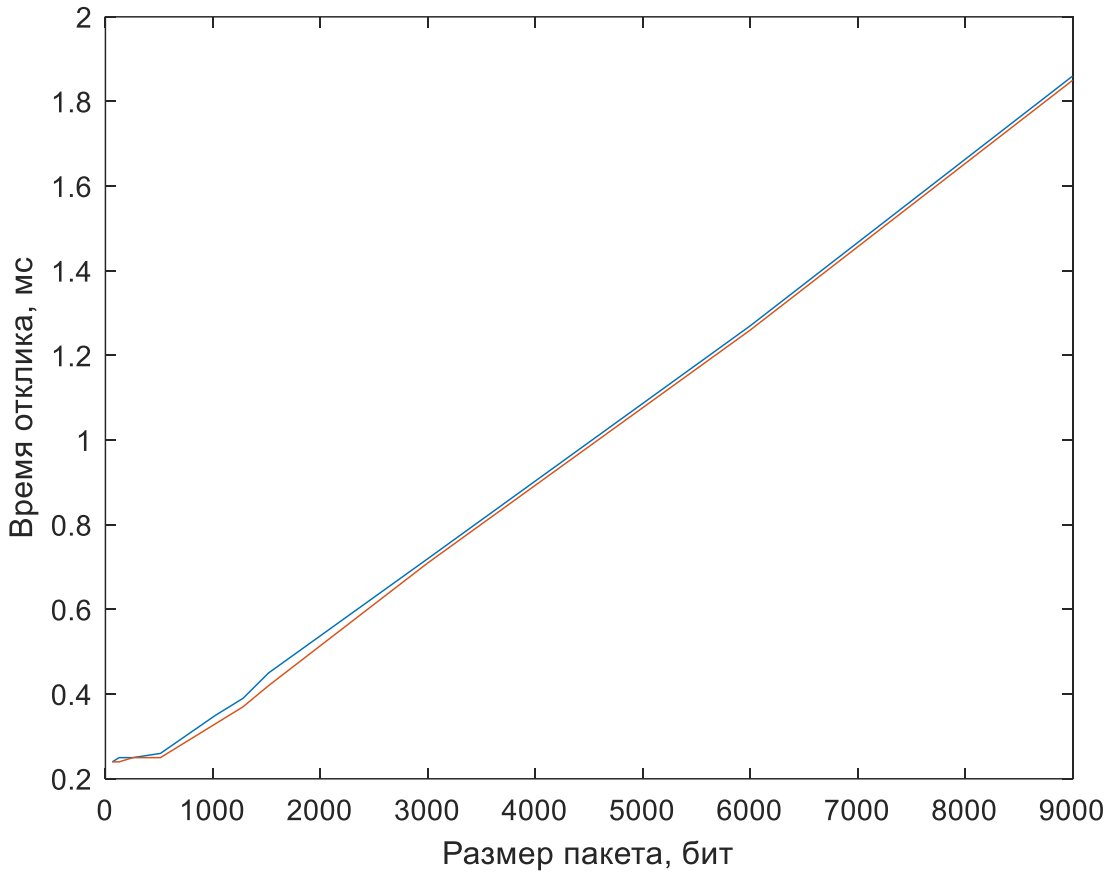

Рисунок 36. Зависимость времени отклика от размера пакета при использовании протокола IPv4 (синий) и IPv6 (оранжевый)

3.3. Транспортный уровень

Транспортный уровень обрабатывает все, что связано с транспортировкой пакетов и включает в себя сквозную доставку, подтверждение успешной

передачи данных и контроль потока и ошибок. Задача данного уровня заключается в повторной передаче данных в случае обнаружения ошибки.

Протоколами транспортного уровня являются UDP и TCP. Ключевым различием между данными протоколами является скорость. UDP является быстрым, простым и эффективным протоколом, однако он не поддерживает повторной передачи потерянных пакетов и не гарантирует доставку в целом.

Также TCP и UDP заключается в том, что UDP не проверяет готовность получателя и может доставлять пакеты вразнобой, TCP же способен обеспечивать упорядоченную доставку данных от пользователя к серверу и наоборот.

Поскольку основной задачей нашего лабораторного макета являет получение экспериментальных данных без потерь и в строгом порядке, то логично, что следует выбрать протокол TCP, который способен обеспечить гарантированную точность и последовательность передачи полученной информации.

Протокол UDP же находит свое применение в сферах, где не так критична потеря нескольких пакетов во всем потоке и главенствует скорость передачи информации. Это могут быть потоковое видеовещание (например сигнал с видеокамер наблюдения, телекамер, IPTV).

Основные различия протоколов TCP и UDP приведены в таблице 6

Таблица 6

	TCP	UDP
Размер заголовка	20-60 байт	8 байт
Формат передачи данных	Поток	Датаграммы
Гарантия доставки	Да	Нет
Упорядоченность данных	Да	Нет
Состояние соединения	Требуется установленное соединение для передачи данных	Без соединения
Повторная передача данных	Повторная передача нескольких кадров в случае потери одного из них	Отсутствие повторной передачи потерянных пакетов

3.3.1. Исследование производительности сети с применением протоколов TCP и UDP

Целью измерения скорости передачи данных будем использовать программу iperf, которая работает по клиент-серверной архитектуре. Клиент генерирует различные типы трафика (в нашем случае TCP и UDP) и посылает на сервер [33].

Измерения будем проводить путем изменения параметра “M”, который характеризирует максимальный размер TCP сегмента MSS. Для этого установим максимальный MTU на сетевых интерфейсах компьютера следующей командой: ifconfig eth0 mtu 9000. В качестве базовой методики тестирования используем методику RFC-2544, которая подразумевает произведение измерений разными значениями кадров от 64 до 1518 байт. Результаты для протокола TCP приведены в таблице 7 и проиллюстрированы на рисунке 37.

Таблица 7

	Размер пакета, бит									
Протокол	64	128	256	512	1024	1280	1518	3000	6000	9000
TCP/IPv4, Мб/с	54,8	74,3	86,1	92,7	95,8	97,1	97,5	98,1	98,5	99
TCP/IPv6, Мб/с	51,2	72,5	84,9	91,1	93,2	95,5	96,1	97,4	98,1	98,8

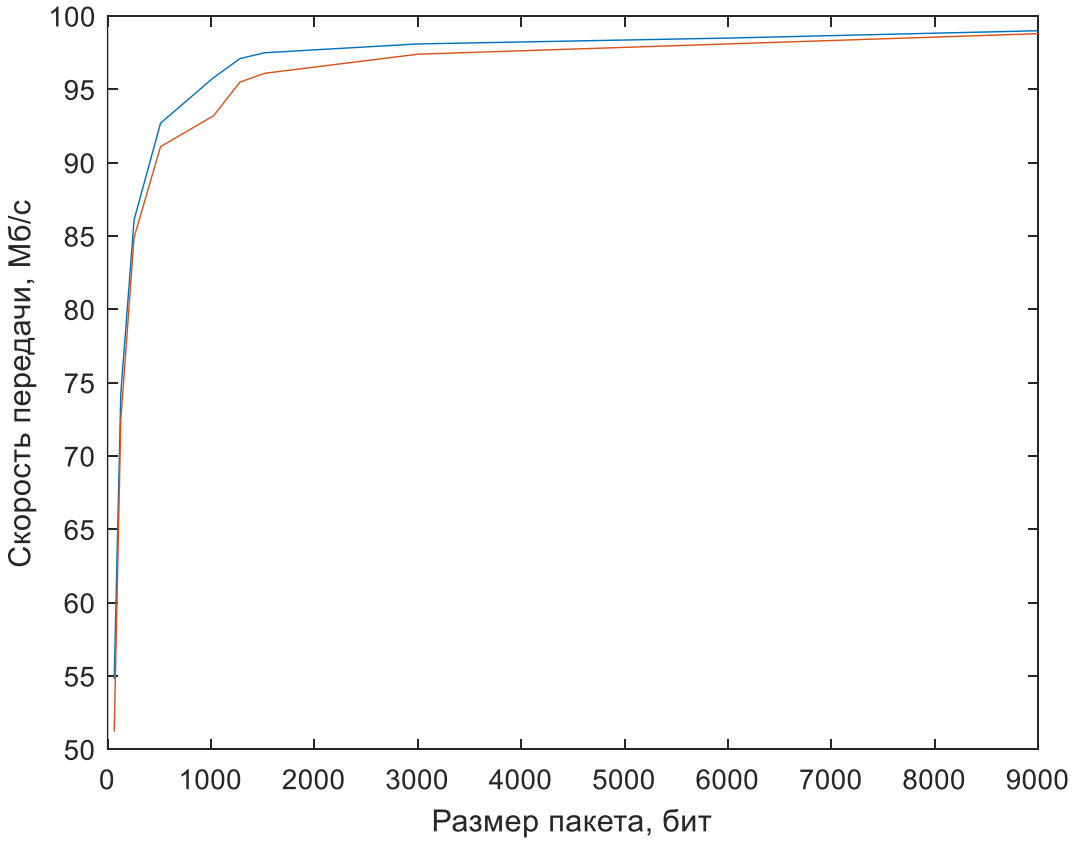

Рисунок 37. Производительность сети с применением протокола TCP/IPv4 (синий) и TCP/IPv6 (оранжевый)

Результаты для протокола UDP приведены в таблице 8 и проиллюстрированы на рисунке 38.

Таблица 8

	Размер пакета, бит									
Протокол	64	128	256	512	1024	1280	1518	3000	6000	9000
UDP/IPv4, Мб/с	57,1	76,6	88,3	94,5	97,1	98,2	98,5	99,1	99,5	99,9
UDP/IPv6, Мб/с	53,4	74,3	85,7	92,2	95,4	96,6	96,8	98,2	98,4	99

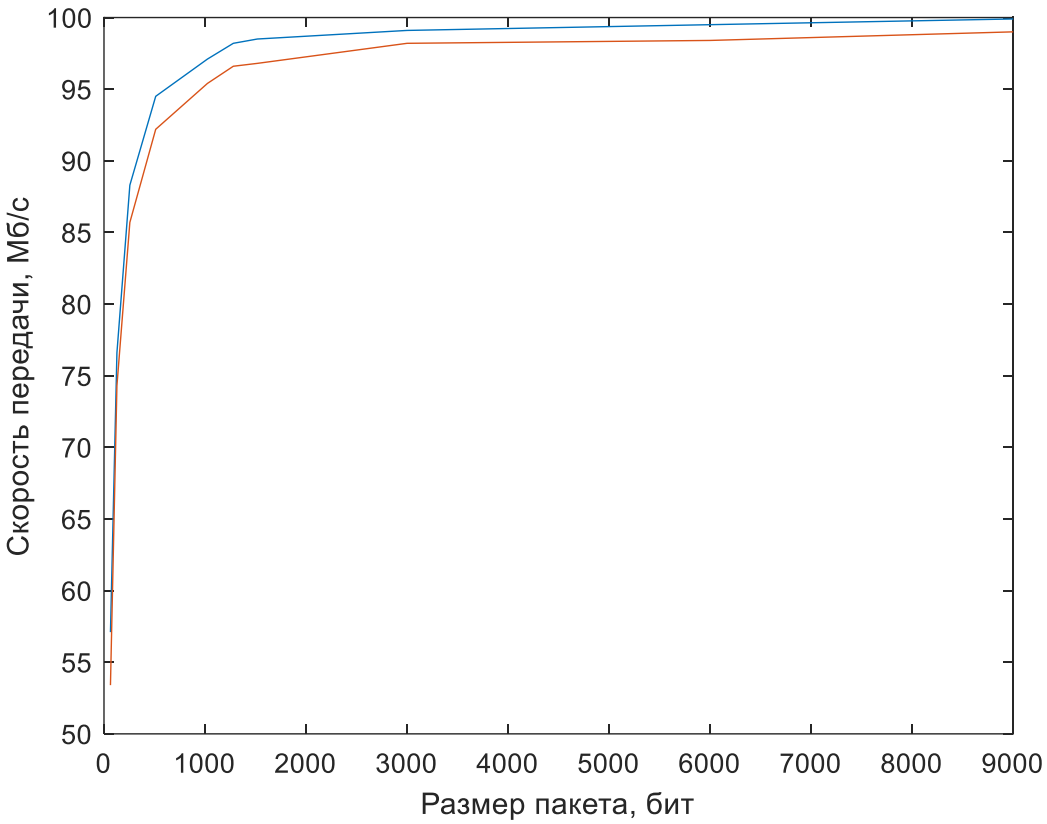

Рисунок 38. Производительность сети с применением протокола UDP/IPv4 (синий) и UDP/IPv6 (оранжевый)

3.3.2. Влияние ошибок интерфейса на производительность протокола TCP

При использовании канала Wi-Fi 802.11n буферизация, необходимая принимающему устройству для обеспечения максимальной производительности, представляет собой объем данных, который может быть передан между получаемыми подтверждениями приема (ACK). Пропускная способность канала Wi-Fi 802.11n составляет 55 Мбит/с. Если обмен данными происходит напрямую между сервером центра обработки данных и пользователем, то время двойного прохождения (время приема-передачи - RTT) должно быть как можно меньше, примерно 1-2 мс (0,002 с). Для канала Wi-Fi 802.11n принимающая система должна иметь возможность буферизовать количество данных, равное произведению полосы пропускания на задержку [39]:

BDP = 55 Мбит/с × 0,002 с

BDP = 55 Мбит/с (1 байт / 8 бит) × 0,002 с

BDP = 6 900 000 байт × 0,002 с

BDP = 13 800 байт

В данном случае, значение BDP меньше размера окна TCP (размер окна TCP по умолчанию = 32 000 байт [39]), то полоса пропускания тракта станет ограничивающим фактором для пропускной способности. Тогда для сохранения заполнения канала буферизации, механизм работы TCP влияет на максимальную пропускную способность. В этом случае передающая система отправляет данные всего окна TCP и ожидает подтверждения от получателя, а затем снова осуществляет передачу. Приложение не может использовать механизм окна отправки, при котором бы TCP полностью заполнит полосу пропускания канала. Поэтому новые данные можно отправлять только после получения подтверждения (ACK). В этом случает максимальная пропускная способность, для источника и получателя, представляет собой размер окна, деленный на время двойного прохождения (время для возврата подтверждения ACK). Тогда размер блока (количество данных, отправленных в одно окно), деленный на время двойного прохождения – это и есть лучшая пропускная способность, которой можно достичь:

Максимальная пропускная способность = размер блока / RTT

Максимальная пропускная способность в бит/с = [байт × 8 (бит в байте)] / RTT

3.3.3. Ожидаемая производительность TCP с учетом ошибок

Рассмотрим допустимую частоту ошибок. В соответствии со стандартами 802.11n приемлемым коэффициентом битовых ошибок (BER) считается 1 ошибка на 10^{10} битов.

Потеря 1 бита за 1×10^{10} бит/с = потеря 1 бита на $1{,}25 \times 10^{9}$ байт/с

Если предположить, что средний пакет имеет длину 1000 байт, показатель BER для Wi-Fi 802.11n будет составлять потерю одного пакета из $1{,}25 \times 10^{5}$ пакетов. В процентном выражении $1/(1{,}25 \times 10^{5}) = 8 \times 10^{-5} = 0{,}00008\% \approx 0{,}0001\%$.

Однако потеря пакетов в каналах TCP может происходить и из-за проблем с производительностью и конфигурацией серверов и сетевых устройств. Из-за того, что пакеты теряются и их необходимо передавать повторно, производительность TCP снижается. Фактическое влияние потерь на максимальную пропускную способность определяется уравнением Матиса:

Максимальная скорость в бит/с < (MSS/RTT) × (1/ $\sqrt{p}$),

где

- MSS = максимальный размер сегмента в байтах
- RTT = время двойного прохождения (приема-передачи) в секундах
- p = вероятность потери пакета

Теперь мы можем применить уравнение Матиса к рассматриваемому в примере каналу Wi-Fi 802.11n. Для MSS используем значение 1460 байт, так как такой объем данных умещается в один пакет TCP. Предположим, что приложение будет передавать блок данных размером 1460 байт и перед отправкой следующих данных ожидать подтверждения (ACK). Поскольку подобный обмен данными происходит внутри кампуса, примем значение RTT составляет 0,002 секунды. Тогда максимальная скорость передачи одного файла со стандартным значением BER для Wi-Fi 802.11n с потерями 0,0001% будет:

Максимальная скорость в соответствии с уравнением Матиса в бит/с < (MSS/RTT) × (1/ $\sqrt{p}$)

Максимальная скорость в бит/с < (1460/0,002) × (1/$\sqrt{0{,}0001}$))

Максимальная скорость в бит/с < 7,3 × 10^7 бит/с

Максимальная скорость в бит/с < 73 Мбит/с

Прогнозируемая по уравнению Матиса скорость превышает максимальную скорость 55 Мбит/с, которая была рассчитана без учета потерь, поэтому максимальной скоростью будет меньшее из двух рассчитанных значений, то есть 55 Мбит/с. Это разумный результат, поскольку каналы, соответствующие допустимому значению BER для Wi-Fi 802.11n, не оказывают отрицательного воздействия на производительность TCP.

Если рассматривать пороговый уровень беспокойства, то потери составляют 1 пакет из 10^3 или 0.001%.

Максимальная скорость в соответствии с уравнением Матиса в бит/с < (MSS/RTT) × (1/(1/$\sqrt{0{,}001}$))

Максимальная скорость в бит/с < (1460/0,002) × (1/(1/$\sqrt{0{,}001}$))

Максимальная скорость в бит/с < 2,3 × 10^7 бит/с

Максимальная скорость в бит/с < 23 Мбит/с

Как можно наблюдать, если потеря пакетов увеличится хотя бы в 10 раз от приемлемого уровня, то скорость передачи упадет больше, чем в 2 раза.

Если продолжать увеличивать потери пакетов ещё в 10 раз, то скорость упадет до значения 7.3 Мбит/с, что уже ниже максимально в 7 раз.

Как видно из расчетов, потеря пакетов оказывает существенное влияние на скорость передачи данных в сети Wi-Fi 802.11n. На рисунке 39 приведена зависимость скорости передачи при различном числе потерь пакетов.

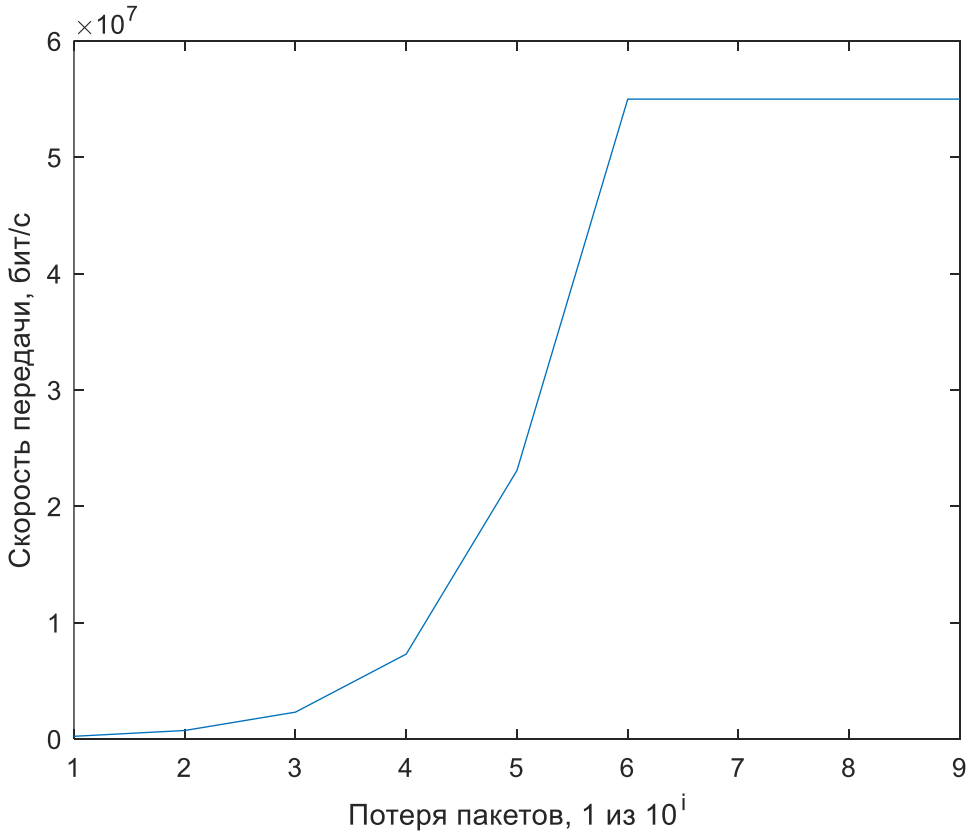

Рисунок 39. Зависимость скорости передачи от количества потерянных пакетов.

3.4. Прикладной уровень

Уровень приложения (уровень рабочего стола) — это среда, где мы можем фактически работать с уровнями OSI.

Поскольку основой лабораторного макета является микроконтроллер ESP-8266, то нам представлен широкий выбор различных шин и протоколов. В их число входят HTTP, MQTT, I2C, I2S, SPI, ESP-NOW (Таблица 1. Протоколы обмена данных уровней OSI).

Выбор протокола зависит от того, какой из них способен эффективно взаимодействовать и обеспечить передачу данных со всех подключенных к лабораторному макету датчиков, а также последующую из передачу на сервер.

В таблице 9 приведены шины, поддерживаемые датчиками лабораторного макета

Таблица 9

Прибор	Шина
Датчик жестов APDS-9960	I2C
Датчик давления BMP-180	I2C
Акселерометр ADX-354	SPI, I2C

Поскольку все датчики работают на шине I2C, то её использование для подключения устройств оказывается наиболее приемлемым.

Принцип работы шины I2C заключается в использовании двух линий: данных (SDA) и синхронизации (SCL). Они служат для передачи информации. Каждое устройство распознается по уникальному адресу и может работать как передатчик или приёмник, в зависимости от назначения устройства. Устройства могут быть классифицированы как ведущие и ведомые при передаче данных. Ведущее - это устройство, которое инициирует передачу данных и вырабатывает сигналы синхронизации. При этом любое адресуемое устройство считается ведомым по отношению к ведущему. Терминология устройств на шине I2C приведена на рисунке 40.

Термин (англ)	Термин (рус)	Описание
Transmitter	Передатчик	Устройство, посылающее данные в шину
Receiver	Приемник	Устройство, принимающее с шины
Master	Ведущий	Начинает пересылку данных, вырабатывает синхроимпульсы, заканчивает пересылку данных
Slave	Ведомый	Устройство, адресуемое ведущим
Multi-master	-	Несколько ведущих могут пытаться захватить шину одновременно, без нарушения передаваемой информации
Arbitration	Арбитраж	Процедура, обеспечивающая **Multi-master**
Synchronization	Синхр.	Процедура синхронизации двух устройств

Рисунок 40. Терминология шины I2C.

- SCL или CLOCK (Signal Clock) – линия, необходимая для тактирования, то есть для управления передачей данных и согласования всех устройств между собой.
- SDA или DATA (Signal Data) - линия передачи данных.

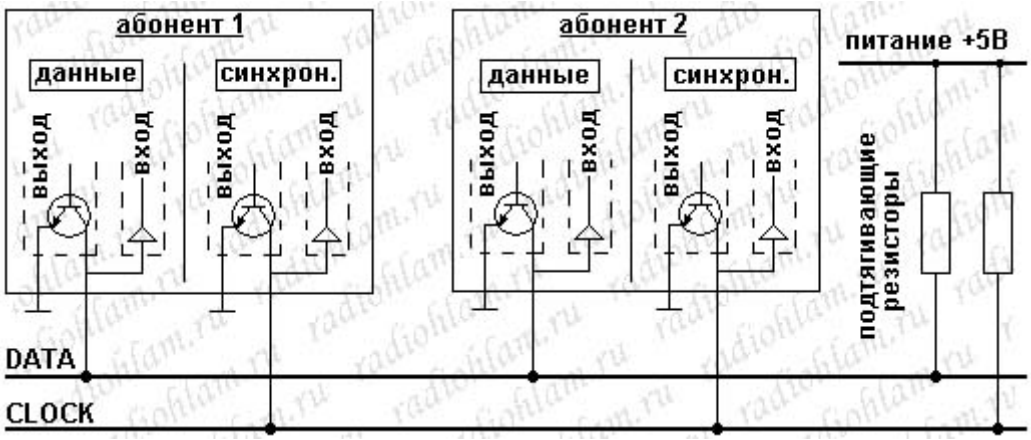

Рисунок 41. Физическая реализация шины I2C.

Специальные ситуации на шине отмечают сигналы START и STOP. Переход линии SDA из ВЫСОКОГО состояния в НИЗКОЕ, когда SCL находится в ВЫСОКОМ состоянии означает START. Переход линии SDA из НИЗКОГО состояния в ВЫСОКОЕ при SCL в ВЫСОКОМ состоянии означает STOP. Сигналы СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после сигнала СТАРТ. Шина считается освободившейся через определенное время после сигнала СТОП. Определение сигналов СТАРТ и СТОП устройствами, подключенными к шине достаточно легко, если в них встроены необходимые цепи. Однако микроконтроллеры без таковых цепей должны осуществлять считывание значения линии SDA как минимум дважды за период синхронизации для того, чтобы определить переход состояния.

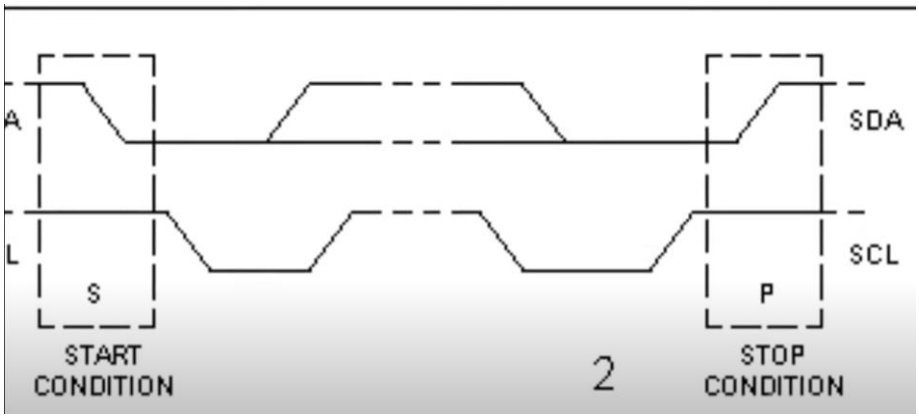

Рисунок 42. Сигналы START и STOP.

3.4.1. Передача данных на сервер через протоколы MQTT и HTTP.

Описание и принцип работы протоколов MQTT и HTTP был приведен ранее. Сейчас же проведем сравнительный анализ этих проколов для выявления наиболее подходящего под нашу задачу.

Протокол MQTT превосходит HTTP, если имеющиеся у устройства регулярно обмениваются данными. MQTT может поддерживать соединение открытым как можно дольше, отправляя только один пакет данных. В отличие от HTTP-связи, которая требует открывать и закрывать соединение для каждого пакета данных, который необходимо отправить. Использование MQTT позволяет значительно снизить загрузку процессора, что является крайне весомым преимуществом, особенно при увеличении числа подключенных лабораторных макетов.

Кроме того, данные в MQTT не нужно шифровать или дешифровать для отправки. HTTP-документ же всегда основан на тексте - необходимо зашифровать и расшифровать любой формат данных, отличный от текстового.

Размер пакета данных:

- Пакет MQTT содержит по крайней мере 2 байта, если соединение уже открыто. В противном случае пакет MQTT-CONNECT зависит от его команд (lastWill и т.д.)
- HTTP-пакет содержит более 8 байт.
- HTTP полагается на текст. Base64 кодирует и декодирует любой двоичный код. Это создает большую нагрузку на центральный процессор.
- Данные в MQTT могут представлять собой данные любого типа – кодирование не требуется.

Время передачи для одного сообщения:

- Каждый HTTP-сеанс (Открытие TCP / IP-соединение - Отправка HTTP-запроса - Получение ответа - Закрытие соединения) занимает приблизительно 247-289 миллисекунд [44]
- Пакеты MQTT (например, пакет CONNECT, пакет PUBLISH) передаются со скоростью около 191-207 миллисекунд [34]

Можно наблюдать, что между MQTT и HTTP нет большой разницы с точки зрения того, сколько времени им требуется для передачи данных, когда они отправляют только одно сообщение/пакет данных.

Передача нескольких сообщений:

MQTT раскрывает свой потенциал, когда через одно соединение отправляется более одного сообщения. На рисунке 43 приведено сравнение среднего объема данных, передаваемых за одно сообщение (байт)

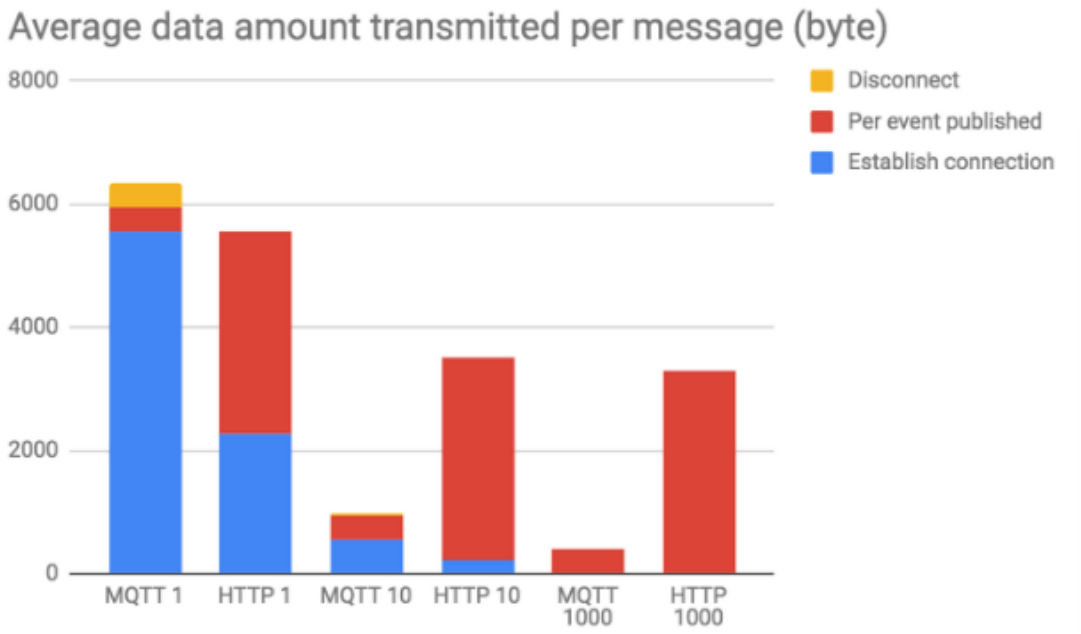

Рисунок 43. Средний объем данных, передаваемых за одно сообщение (байт).

На приведенном выше графике цифры от 1 до 1000 указывают на разное количество сообщений, переданных по одному и тому же соединению. т.е. MQTT 1 - одно единственное сообщение, отправленное с использованием одного открытого соединения, 10 – 10 сообщений и т.д.

Как показано выше, протокол MQTT работает очень эффективно, если используется уже открытое соединение. Он превосходит HTTP как по использованию данных (т.к. не требуется повторное подключение), так и по времени, затрачиваемому на передачу данных (примерно на 25%).

Это происходит потому, что MQTT не перестраивает соединение каждый раз.

Надежность:

Надежность в протоколе необходима, когда клиенту нужно узнать, были ли отправленные данные получены и подтверждены или нет. В MQTT это обеспечивается уровнями QoS (Quality of Service - Качество обслуживания). Для реализации надежности в HTTP, необходимо прибегнуть к вводу дополнительных мер.

Основные цели использование протоколов:

Наиболее существенное различие заключается в назначении протоколов HTTP и MQTT. Из всех сравнений выше вытекает, что модели pub/sub (MQTT) и one-to-many (HTTP) предназначены для разных целей.

Протокол MQTT лучше HTTP для использования в интернете вещей (IoT) для поддержания так называемых “пулов подключения” (connection pool), позволяющих осуществлять двунаправленную связь для нескольких устройств, подключенных одним объектом. Кроме того, MQTT обеспечивает связь в режиме реального времени, поскольку не руководствуется последовательной иерархией. HTTP должен обрабатывать каждый запрос в иерархическом порядке (первым пришел-первым обслужен).

Итог:

HTTP и MQTT созданы для разных целей, даже несмотря на то, что модель pub-sub настройки MQTT похожа на протокол обмена запрос-ответ HTTP.

MQTT является идеальным выбором для использования в домене Интернета вещей (IoT) благодаря двунаправленной связи. Он обеспечивает надежную передачу, предотвращает перегрузку данных (при условии, что клиенты используют открытое соединение) и может быстрее передавать сообщения от издателя к подписчику. Однако все это нивелируется, если соединение не будет открыто. В этом случае HTTP лучше, так как был разработан именно для этого процесса: запрашивать один документ со всеми компонентами за один сеанс подключения.

HTTP же является хорошим выбором для отображения информации, MQTT это решение при необходимости частой связи (обмен

сообщениями/данными). Главным преимуществом MQTT является способность поддерживать соединения открытыми и способ обработки форматов данных. Оба варианта обеспечивают успешную и надежную передачу сообщений. С другой стороны, если нет нужды в частом обмене данными, следует выбрать HTTP.

Итоговая таблица со сравнением основных черт протоколов MQTT и HTTP представлена на таблице 10.

Таблица 10

	MQTT	HTTP
Модель	Издатель/подписчик	Клиент-сервер
Подключение	Через connect-пакеты	Каждый раз новая сессия
Сессия	Пока брокер не разорвет соединения	Одна сессия запрос-ответ
Скорость передачи одного сообщения	6336 Байт	5546 Байт
Скорость передачи нескольких сообщений	500-1000 Байт	3250-3500 Байт
Безопасность	На основе протокола передачи	На основе протокола передачи
Кодирование	Без кодирования/декодирования	Кодирование/декодирование Base64
Надежность	Обеспечивается QoS	Необходимо вводить дополнительные меры

Исходя из всего перечисленного выше, можно сделать вывод о преимущественном использовании прокола MQTT для лабораторной макета. Он способен обеспечить скоростную передачу необходимой информации с сенсоров, а также снизить нагрузку на центральный процессор сервера и микроконтроллера ESP-8266, что особенно необходимо при подключении нескольких таких макетов к одному серверу.

3.4.2. Исследование скорости передачи данных при использовании протоколов MQTT и HTTP

Исследование заключается в измерении скорости передачи данных на облачный сервер, используя протоколы HTTP и MQTT QoS0 [45].

Для обработки всех запросов протоколов на аутентификацию и подключения к серверу была использована программа Tsung - распределённая система нагрузочного и стресс-тестирования. Данная программа настроена на генерацию большого количества сообщений с одного компьютера. Tsung настроена на тип debug, для возможности длительного ведения журнала событий. Результаты измерений приведены на рисунке 44.

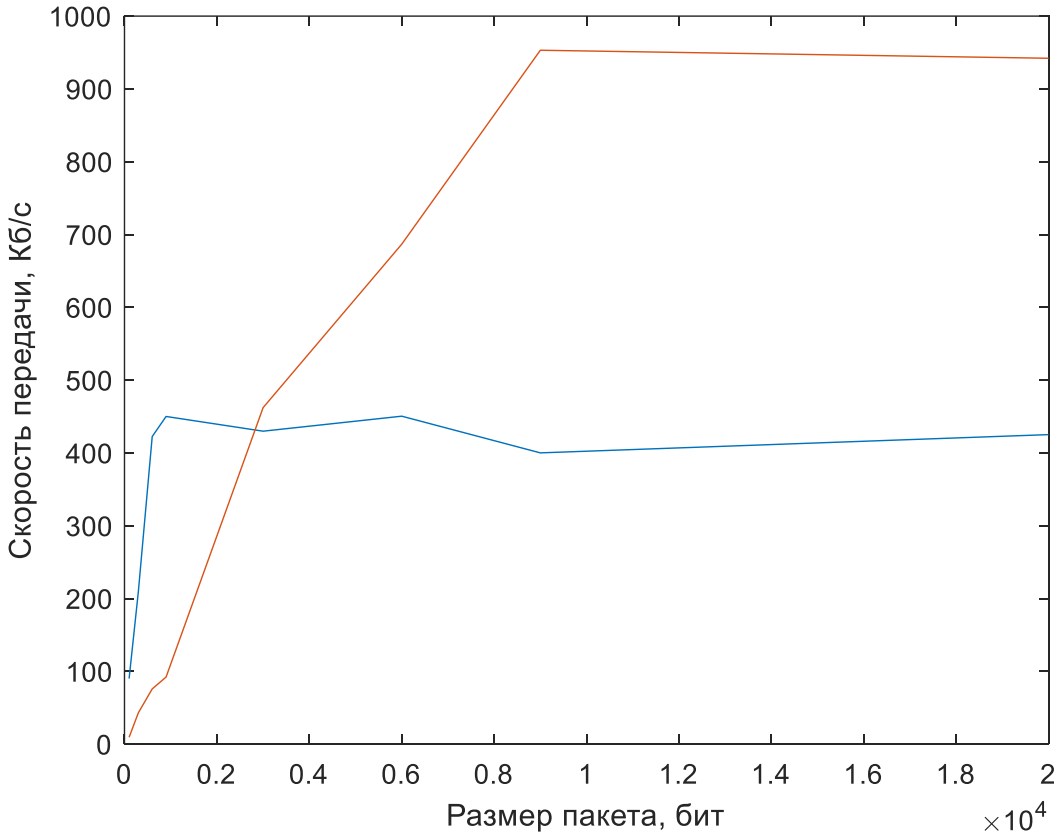

Рисунок 44. Зависимость скорости передачи данных от размера пакета при использовании протокола HTTP (синий) и MQTT (оранжевый)

Как видно, при малом размере пакета большую производительность показывает протокол HTTP, однако, когда число отправляемых запросов превышает 3000, скорость передачи при использовании протокола MQTT

возрастает. Предельная скорость для протокола MQTT в данном случае составляет 900 Кб/с. Максимальная скорость передачи протокола HTTP установилась в районе 450 Кб/с.

Как видно из результатов исследования, протокол MQTT выигрывает относительно HTTP при длительном обмене большим числом данных через открытое соединение, что подтверждает сравнительный анализ, приведенный в пункте 3.4.1.

4. Безопасность жизнедеятельности

4.1. Класс защиты от поражения электрическом током

Согласно ГОСТ IEC 61140-2012 [40] существует 4 класса электрооборудования:

0 класс – электрическое оборудование, в котором основную изоляцию используют в качестве меры предосторожности для основной защиты, а защита при повреждении не предусмотрена.

1 класс – электрическое оборудование, в котором основную изоляцию используют в качестве меры предосторожности для основной защиты, а защитное соединение – в качестве меры предосторожности для защиты при повреждении.

2 класс – электрическое оборудование, в котором основную изоляцию используют в качестве меры предосторожности для основной защиты, а дополнительную изоляцию – в качестве меры предосторожности для защиты при повреждении, или в котором основную защиту и защиту при повреждении обеспечивают усиленной изоляцией.

3 класс – электрическое оборудование, в котором ограничение напряжения значением сверхнизкого напряжения используют в качестве меры предосторожности для основной защиты, а защита при повреждении не предусмотрена.

Стоит упомянуть, что 0 класс электрооборудования рекомендуется исключить, однако он включен в ГОСТ IEC 61010-1-2014, так как упоминается в некоторых стандартах на современные изделия.

Для 1 класс характерно использование основной изоляции для защиты в рабочих условиях, а защищенность при повреждении обеспечивается посредством подсоединения открытой проводящей части к защитному проводнику. Для предохранения от поражения током при работе с электрооборудованием 1 категории заземление оборудуется через розетку и вилку. Каждый компьютер, стиральная машина, микроволновая печь имеют такие вилки и розетки, в которые уже встроено заземление. Если же такого заземления нет, то считается, что приборы относятся к классу 0. При 2 классе используется двойная или усиленная изоляция. Электрическое оборудование 3 класса применяется только в помещениях с повышенной опасностью и при сверхнизких напряжениях – 12, 36, 40, 50 В. Например, в электроустановках такие напряжения подаются на электропаяльники, светильники переносные в смотровых ямах гаражей должны иметь сверхнизкое напряжение до 50 В.

Питание на лабораторный макет поступает через микроконтроллер ESP-8266 посредством подключения кабеля Micro-USB, работающем на напряжении 3.3В, что является сверхнизким значением. Следовательно, лабораторная установка может быть классифицирована как электрооборудование 3 класса.

Так как электрическое оборудование относится к 3 класс, то согласно ГОСТ IEC 61010-1-2014 [41] в нем должна быть предусмотрена защита от прямого и косвенного прикосновения.

Для защиты токопроводящих часть установки от различных видом прикосновений следует использовать кожухи и защитные барьеры.

Из ГОСТ IEC 61010-1-2014 следует, что безопасные значения напряжения для доступных частей при нормальном применении представляют собой 33 В (среднеквадратичное), 46,7В (пиковое) для переменного тока и 70 В для постоянного тока. В условия единичной неисправности значения

напряжения не должны превышать 55 В (среднеквадратичное), 78В (пиковое) для переменного тока и 140 В для постоянного тока.

Кожухи и защитные барьеры должны соответствовать требованиям, установленным в части механической жесткости.

Если кожухи или защитные барьеры обеспечивают защиту с помощью изоляции, они должны соответствовать требованиям основной изоляции.

Если кожухи или защитные барьеры обеспечивают защиту с помощью ограничения доступа, зазоры и пути утечки между доступными частями и частями, опасными для жизни, должны соответствовать требованиям и применимым требованиям к основной изоляции.

4.2. Код IK

В нашем случает защитный кожух должен соответствовать требования механической жесткости ГОСТ IEC 62262-2015 [42]. Они представляют собой то, что оборудование не должно приводить к возникновению опасности при механических воздействиях, которые могут происходить при его нормальном применении. Нормальный уровень энергии механической защиты составляет 5 Дж. Допускаются уровни ниже 5 Дж, но не менее 1 Дж при условии, что выполняются все следующие условия:

a) минимальный уровень должен быть подтвержден оценкой риска, выполняемой изготовителем;

b) при установке оборудования для его назначенного применения к нему должна быть обеспечена невозможность случайного прикосновения лиц, не имеющих прав доступа к нему, и обычных людей;

c) при нормальном применении доступ к оборудованию должен быть возможен только при необходимости, например для настройки, программирования или технического обслуживания;

d) на оборудовании должна быть маркировка кода IK в соответствии с IEC 62262, а номинальный уровень энергии и метод испытания должны быть указаны в сопроводительных документах. При неметаллических корпусах и

минимальной номинальной температуре окружающей среды ниже 2°С указанное в сопроводительных документах значение уровня энергии должно быть значением, применимым при наименьшей номинальной температуре окружающей среды. Если используемые воздействующие механические энергии находятся в интервале значений IK, приведенных в IEC 62262, маркировка IK должна быть для ближайшего минимального значения. Состав кода IK приведен на рисунке 45.

Рисунок 45. Состав кода IK

Так как уровень энергии механической защиты составляет 5 Дж, то в соответствии с ГОСТ IEC 62262-2015 для исследуемого лабораторного макета может быть присвоен код IK08.

4.3. Код IP

Код IP - система кодификации, применяемая для обозначения степеней защиты, обеспечиваемых оболочкой, от доступа к опасным частям, попадания внешних твердых предметов, воды, а также для предоставления дополнительной информации, связанной с такой защитой. Степень защиты, согласно ГОСТ 14254-2015 [43], обеспечиваемая оболочкой, указывается кодом IP формат которого изображенном на рисунке 46, пример использования букв в коде представлен на рисунке 47.

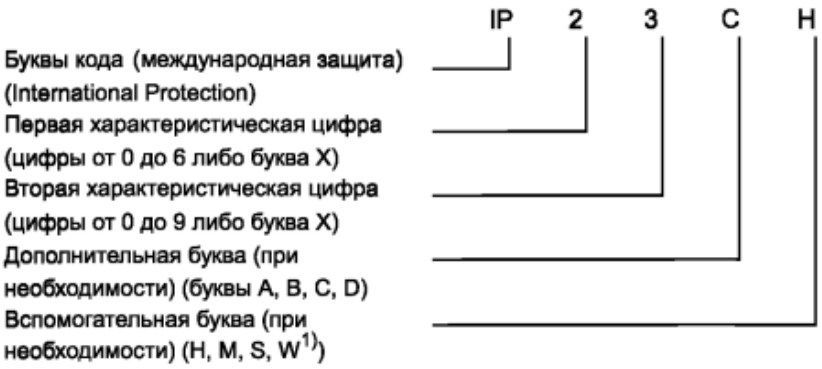

Рисунок 46. Состав кода IP

Элемент	Цифры или буквы	Значение для защиты оборудования	Значение для защиты людей
Буквы кода	IP	—	—
Первая характеристическая цифра	0	От проникновения внешних твердых предметов: нет защиты	(*исключено*) (*T*)
	1	диаметром ≥ 50 мм	
	2	диаметром ≥ 12,5 мм	
	3	диаметром ≥ 2,5 мм	
	4	диаметром ≥ 1,0 мм	
	5	пылезащищенное	
	6	пыленепроницаемое	
Вторая характеристическая цифра	0	От вредного воздействия в результате проникновения воды: нет защиты	—
	1	вертикальное каплепадение	
	2	каплепадение (номинальный угол 15°)	
	3	дождевание	
	4	сплошное обрызгивание	
	5	действие струи	
	6	сильное действие струи	
	7	временное непродолжительное погружение	
	8	длительное погружение	
	9	горячая струя воды под высоким давлением	

Рисунок 47. Расшифровка основных цифр кода IP

Для присвоения кода IP для защиты от проникновения посторонних предметов и жидкостей следует уточнить условия эксплуатации лабораторного макета.

Установка рассчитана на использование в университетской лаборатории. Оно представляет собой помещение со средней влажностью 50%

без источников жидкости, с большим количеством мелких твердых предметов и пыли. Следовательно, в соответствии с ГОСТ 14254-2015, значение первой цифры кода IP рекомендуется выбрать от 4 до 5, чтобы обеспечить защиту оборудования от твердых предметов и пыли для его долговременного корректного функционирования. Вторую цифру кода IP рекомендовано выбрать в пределах от 0 до 1, так как не предполагается взаимодействие установки с жидкостью.

ЗАКЛЮЧЕНИЕ

В ходе работы было проведено описание всех составляющих устройств лабораторного макета, а также описание принципа действия различных протоколов, использующихся данными устройствами. Эти сведения могут в последующем быть использованы в качестве методических указаний или теоретического обоснования работы макета на занятиях кафедры РЭС и лаборатории.

Были проведены исследования с целью выявления наиболее подходящих протоколов уровней модели OSI. В ходе них исследовалась скорость передачи данных при различных режимах работы и уровнях потери пакетов. Также был проведен сравнительный анализ протоколов по основным критериям.

В результате были выявлены оптимальные протоколы для обеспечения наиболее эффективной работы лабораторной установки. Данными протоколами являются Wi-Fi 802.11n на физическом, IPv6 на сетевом, TCP на транспортном, MQTT на прикладном уровне модели OSI.

Также был разработан код программы взаимодействия устройств, составляющих лабораторный макет, для получения данных с цифровых и аналоговых датчиков и последующей передачи их на ПК, а так же буферизацию на сервере.

Данные результаты могут быть использованы в дальнейшем для проектирования схожих установок или устройств похожего назначения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ESP 8266 datasheet [Электронный ресурс] URL: https://www.chipdip.ru/product/esp-06
2. ESP8266. Введение. [Электронный ресурс] URL: https://kit.alexgyver.ru/tutorials/wemos-basic/
3. Обзор платы NodeMCU ESP8266 и ее использование в Arduino IDE [Электронный ресурс] URL: https://radioprog.ru/post/863
4. ESP 8266 documentation [Электронный ресурс] URL: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf
5. APDS9960 datasheet [Электронный ресурс] URL: https://content.arduino.cc/assets/Nano_BLE_Sense_av02-4191en_ds_apds-9960.pdf
6. Распознавание жестов с помощью APDS-9960 [Электронный ресурс] URL: https://habr.com/ru/articles/424947/
7. Подключение APDS-9960 датчика жестов Ардуино [Электронный ресурс] URL: https://роботехника18.рф/apds-9960/#1
8. APDS-9960 light and gesture sensor [Электронный ресурс] URL: https://tasmota.github.io/docs/APDS-9960/
9. BMP180 (Hw-596) datasheet [Электронный ресурс] URL: https://iarduino.ru/lib/datasheet%20bmp180.pdf
10. Обзор датчика давления BMP180 (BMP080) [Электронный ресурс] URL: https://robotchip.ru/obzor-datchika-davleniya-bmp180/
11. Взаимодействие BMP180, датчика атмосферного давления и температуры, с Arduino [Электронный ресурс] URL: https://radioprog.ru/post/796
12. ADX-354 datasheet [Электронный ресурс] URL: https://ru.mouser.com/datasheet/2/609/adxl354_355-1503915.pdf
13. Подключение акселерометра ADXL345 к Arduino [Электронный ресурс] URL: https://arduino-ide.com/modules/38-podkljuchenie-akselerometra-adxl345-k-arduino.html
14. ESP8266 data transfer protocols [Электронный ресурс] URL: https://tttapa.github.io/ESP8266/Chap05%20-%20Network%20Protocols.html
15. Стандарты Wi-Fi [Электронный ресурс] URL: https://wifigid.ru/poleznoe-i-interesnoe/standarty-wi-fi
16. Как работает Wi-Fi [Электронный ресурс] URL: https://wifigid.ru/besprovodnye-tehnologii/kak-rabotaet-wi-fi

17. Ethernet [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=Ethernet
18. Протокол IP — протокол интернет. Формат заголовка IP-пакета. [Электронный ресурс] URL: https://zvondozvon.ru/tehnologii/protokoli/ip
19. Протокол UDP [Электронный ресурс] URL: http://mypractic.ru/urok-69-protokol-udp-sozdanie-udp-servera-i-klienta-s-pomoshhyu-biblioteki-uipethernet.html
20. Протокол TCP [Электронный ресурс] URL: https://zvondozvon.ru/tehnologii/protokoli/tcp
21. Протокол UDP — преимущества, недостатки и применение [Электронный ресурс] URL: https://zvondozvon.ru/tehnologii/protokoli/udp
22. HTTP протокол передачи Гипертекста — основа системы WWW [Электронный ресурс] URL: https://zvondozvon.ru/tehnologii/protokoli/http
23. Протокол HTTP [Электронный ресурс] URL: http://mypractic.ru/urok-70-protokol-http-sozdanie-web-servera-na-arduino-ispolzovanie-html-koda.html
24. Обзор протокола HTTP [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Overview
25. EP8266. Управление устройствами по MQTT протоколу [Электронный ресурс] URL: https://projectalt.ru/publ/arduino_i_esp/ep8266_upravlenie_po_mqtt_protokolu/3-1-0-30
26. Протокол MQTT: концептуальное погружение [Электронный ресурс] URL: https://habr.com/ru/articles/463669/
27. Как настроить I2C-связь на Arduino [Электронный ресурс] URL: https://arduinoplus.ru/i2c-svyaz-arduino/#_I2C
28. Потеря пакетов TCP/IP [Электронный ресурс] URL: https://vpautine.ru/oshibki/potera-paketov-interneta-kak-ispravit
29. Ошибки интерфейса TCP/IP [Электронный ресурс] URL: https://networkguru.ru/vliyanie-oshibok-interfejsa-na-proizvoditelnost-tcp/
30. TCP vs HTTP - What are the Key Differences? [Электронный ресурс] URL: https://www.techgeekbuzz.com/blog/tcp-vs-http/
31. Отличия TCP и UDP [Электронный ресурс] URL: https://selectel.ru/blog/tcp-vs-udp/
32. Шина I2C принцип работы [Электронный ресурс] URL: https://sysadminmosaic.ru/_media/i2c/i2c_description_rus.pdf
33. Метод анализа сетей IPv4 и IPv6 [Электронный ресурс] URL: https://moluch.ru/archive/210/51389/

34. MQTT vs HTTP [Электронный ресурс] URL: https://cedalo.com/blog/http-vs-mqtt-for-iot/#What_is_MQTT
35. Протокол MQTT. Особенности, варианты применения, основные процедуры MQTT http://lib.tssonline.ru/articles2/fix-corp/protokol-mqtt-osobennosti-varianty-primeneniya-osnovnye-protsedury-mqtt-protocol.
36. Применение протокола MQTT: мониторинг оборудования, сред и отправка важных данных https://mcs.mail.ru/blog/protokol-peredachi-dannyh-mqtt
37. Что такое MQTT и для чего он нужен в IIoT? Описание протокола MQTT https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/#oborudovanie
38. Пример применения MQTT https://future2day.ru/protokol-mqtt/
39. Протокол MQTT установка https://aiu.susu.ru/iot/mqtt
40. ГОСТ IEC 61140-2012 “Защита от поражения электрическим током. Общие положения безопасности установок и оборудования” Москва, Стандартинформ, 2014.
41. ГОСТ IEC 61010-1-2014 “Безопасность электрических контрольно-измерительных приборов и лабораторного оборудования” Москва, Стандартинформ, 2015.
42. ГОСТ IEC 62262-2015 “Степени защиты, обеспечиваемой оболочками от наружного воздействия (код IK)” Москва, Стандартинформ, 2016.
43. ГОСТ 14254-2015 “Степени защиты, обеспечиваемые оболочками (Код IP)” Москва, Стандартинформ, 2016.
44. Performance Analysis of Internet of Things Protocols Based Fog/Cloud over High Traffic [Электронный ресурс] URL: https://www.researchgate.net/publication/323943358_Performance_Analysis_of_Internet_of_Things_Protocols_Based_FogCloud_over_High_Traffic

ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ

```
#define PIN_LED1 12  //объявление констант (названий пинам 12 (D6), 13 (D7), 15 (D8))
#define PIN_LED2 13
#define PIN_LED3 15
#define PIN_LIGHT_A0 A0    // объявления названия аналоговому пину A0

#define FASTLED_ESP8266_RAW_PIN_ORDER
#include <FastLED.h>     // подключаем библиотеку для управления светодиодной лентой
#define NUM_LEDS_IN_STRIPLINE 8    // указываем количество светодиодов на ленте
#define DATA_PIN 16   // указываем пин для подключения ленты
CRGB ledsLine[NUM_LEDS_IN_STRIPLINE];    // выделим память (массив) для управления лентой

#include <SFE_BMP180.h>   // подключаем библиотеку для управления барометром
#include <Wire.h>   // подключаем библиотеку для связи через интерфейс I2C/TWI
SFE_BMP180 pressure;    // создаем экземпляр класса SFE_BMP180 и называем его «pressure» (выделяем память)
double baseline; // исходное давление

#include <Adafruit_Sensor.h>    // подключаем библиотеку для ???
#include <Adafruit_ADXL345_U.h>    // подключаем библиотеку для управления акселерометром
/* Assign a unique ID to this sensor at the same time */
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345); // Создаём объект, экземпляр класса ADXL345_U

#include <SparkFun_APDS9960.h>     // подключаем библиотеку для работы с датчиком жестов, приближения, освещенности, цвета APDS-9960
// Global Variables
SparkFun_APDS9960 apds = SparkFun_APDS9960();   // Создаём объект, экземпляр класса SparkFun_APDS9960
uint16_t ambient_light = 0;   // переменные фиксированной длины (16 бит)
uint16_t red_light = 0;
uint16_t green_light = 0;
uint16_t blue_light = 0;
```

```
/*char webPageMain[] PROGMEM = R"=====(
<safasf>
</asfasf>
)=====";*/

void setup() {
  Serial.begin(115200);
//  while (!Serial);
  Serial.println("Begin scanning\n");

  Serial.println(webPageMain);

  LedCheck();   // проверка светодиодов
  LightCheck();   // проверка фотодетектора
  StripLineCheck();   // проверка линии светодиодов
  PresureSensorCheck();   // проверка барометра
  AxelerometerCheck();    // проверка акселерометра
  GuesturesSensorCheck();   //проверка датчика освещенности
}

void loop() {
}
  // функция проверки светодиодов
void LedCheck()
{
  pinMode(PIN_LED1, OUTPUT);    // перевод цифровых портов в режим вывода
  pinMode(PIN_LED2, OUTPUT);
  pinMode(PIN_LED3, OUTPUT);

  digitalWrite(PIN_LED1, 1);    // подача цифрового сигнала TRUE
  Serial.println("\nLED1 - ON");
  delay(2000);
  digitalWrite(PIN_LED1, 0);
  digitalWrite(PIN_LED2, 1);
  Serial.println("LED1 - OFF");
  Serial.println("LED2 - ON");
  delay(2000);
  digitalWrite(PIN_LED2, 0);
```

```
    digitalWrite(PIN_LED3, 1);
    Serial.println("LED2 - OFF");
    Serial.println("LED3 - ON");
    delay(2000);
    digitalWrite(PIN_LED3, 0);
    Serial.println("LED3 - OFF");
    Serial.println("If all 3 leds is Flasing - connection right\n");
    delay(2000);
  }
    // функция проверки фотодатчика (зачем если есть датчик освещенности???) чтобы показать возможности аналогового пина??
  void  LightCheck()
  {
     int val = analogRead(PIN_LIGHT_A0);   // считываем значение с аналогового пина
     if(val != 0 && val != 1023)
     {
       Serial.print("Lightness sens: OK\nValue: ");
       Serial.println(val);
     }
     else
       Serial.println("Ligtness sens FAILED\nCheck the connection\n");
  }

    // функция проверки линии светодиодов
  void  StripLineCheck()
  {
          FastLED.addLeds<WS2812B, DATA_PIN, RGB>(ledsLine, NUM_LEDS_IN_STRIPLINE);   // основные настройки адресной ленты: WS2812B - модель ленты,
    int decay = 80;   // задержка при мигании светодиодов в мс

    for(int i = 0; i < NUM_LEDS_IN_STRIPLINE; i++)     // цикл мигания каждого светодиода каждым цветом
    {
      ledsLine[i] = CRGB::Red;    // запись значения красный в массив
      FastLED.show();   // передача значения на светодиод
      delay(decay);   // задержка
      ledsLine[i] = CRGB::Black;    // запись значения черный в массив
      FastLED.show();
```

```
    delay(decay);

    ledsLine[i] = CRGB::DarkViolet ;  // фиолетовый
    FastLED.show();
    delay(decay);
    // Now turn the LED off, then pause
    ledsLine[i] = CRGB::Black;
    FastLED.show();
    delay(decay);

    ledsLine[i] = CRGB::NavajoWhite;    // бежевый
    FastLED.show();
    delay(decay);
    // Now turn the LED off, then pause
    ledsLine[i] = CRGB::Black;
    FastLED.show();
    delay(decay);

    ledsLine[i] = CRGB::Salmon;   // оранжевый (розовый)
    FastLED.show();
    delay(decay);
    // Now turn the LED off, then pause
    ledsLine[i] = CRGB::Black;
    FastLED.show();
    delay(decay);
  }

  Serial.println("\nIf all of the leds are blinked a default colors, then status: OK");
  Serial.println("If not all the led are blinked an default colors, then status: EROR\n");
}

  // функция измерения давления с учетом температуры
double getPressure()
{
  char status;
  double T,P,p0,a;

  // чтобы рассчитать давление, сначала нужно измерить температуру
```

```
// запускаем измерение температуры; если функция
// будет выполнена успешно, она вернет количество
// миллисекунд, потребовавшихся на измерение,
// а если неуспешно, то вернет «0»:

status = pressure.startTemperature();
if (status != 0)
{
 // ждем, когда завершится измерение:

 delay(status);

 // извлекаем данные о температуре; обратите внимание,
 // что измеренные данные хранятся в переменной «T»;
 // если функция будет выполнена успешно, она вернет «1»,
 // а если нет, то «0»

 status = pressure.getTemperature(T);      // извлекаем данные о температуре;
 if (status != 0)
 {
  // запускаем измерение давления; параметр отвечает
  // за частоту дискретизации данных; допустимые значения
  // для параметра – от «0» до «3», где «3» - это
  // самое высокое разрешение, но и самая долгая задержка;
  // если функция будет выполнена успешно, она вернет
  // количество миллисекунд, потребовавшихся на ожидание,
  // а если неуспешно, то «0»:

  status = pressure.startPressure(3);
  if (status != 0)
  {
   // ждем, когда завершится измерение:
   delay(status);

   // извлекаем данные о давлении; обратите внимание,
   // что измеренные данные хранятся в переменной «P»;
   // Используйте '&P' чтобы указать путь к P.
   // также обратите внимание, что этой функции
   // требуются данные о температуре (переменная «T»);
```

```
            // если температура стабильна, то для многократного
            // измерения давления вы можете измерить температуру всего один раз
            // Функция возвращает 1 в случае успеха, 0 в случае провала

            status = pressure.getPressure(P,T);
            if (status != 0)
            {
              return(P);    //возвращение измеренного значения в мбар (гПа)
            }
            else Serial.println("ошибка при получении данных о измеренном давлении");
          }
          else Serial.println("ошибка при запуске измерения давления");
        }
        else Serial.println("ошибка при получении данных о измеренной температуре");
      }
      else Serial.println("ошибка при запуске измерения температуры");
    }
      // функция проверки барометра
    void PresureSensorCheck()
    {
      if (pressure.begin())    // инициализируем датчик (извелечение калибровочных данных)
        Serial.println("Инициализация датчика давления BMP180 прошла успешно");
      else
      {
        Serial.println("Инициализация датчика давления BMP180 провалилась (проблемы с подключением)");
      }

      baseline = getPressure();   // получение значения давления в мбар (гПа)

      Serial.print("исходное давление: ");
      Serial.print(baseline);
      Serial.println(" мбар");
      Serial.println("if you seen a pressure var, then Status: OK");
       Serial.println("if you seen a zero or invalid var, then Status: ERROR, check the wiring\n");
    }
     // функция проверки акселерометра (НЕ ОСОБО РАЗОБРАЛСЯ!)
```

```
void AxelerometerCheck()
{
  // Инициализация сенсора
 if(!accel.begin())
 {
    Serial.println("Проблема подключения акселерометра ADXL345 (проверьте соединение)");
  while(1);
 }

 // Set the range to whatever is appropriate for your project
 accel.setRange(ADXL345_RANGE_16_G);

 int i = 0;
 // Display the results (acceleration is measured in m/s^2)
 while(i++ < 10)
 {
 // Get a new sensor event
 sensors_event_t event;
 accel.getEvent(&event);

  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");
                Serial.print("Z:          "); Serial.print(event.acceleration.z); Serial.print("  ");Serial.println("m/s^2 ");
  delay(300);
 }

 Serial.println("If you see a  several items data - then status: OK");
 Serial.println("If no data coming, then status: ERROR - check the wiring\n");
}
  // функция проверки датчика освещенности
void GuesturesSensorCheck()
{
 Serial.println(F("SparkFun APDS-9960 - ColorSensor"));

 if ( apds.init() ) {    // проверка нициализации APDS-9960
  Serial.println(F("APDS-9960 initialization complete"));
 } else {
```

```
    Serial.println(F("Something went wrong during APDS-9960 init!"));
  }

  if ( apds.enableLightSensor(false) ) {     // проверка включения режима определения уровня освещённости
    Serial.println(F("Light sensor is now running"));
  } else {
    Serial.println(F("Something went wrong during light sensor init!"));
  }
  // Wait for initialization and calibration to finish
  delay(300);

  int i = 0;
  while(i++ < 500)
  {
     //  проверка чтения различных цветов (ambient, red, green, blue), занесение результатов в соответствующие переменные
    if (  !apds.readAmbientLight(ambient_light) ||
        !apds.readRedLight(red_light) ||
        !apds.readGreenLight(green_light) ||
        !apds.readBlueLight(blue_light) ) {
      Serial.println("Error reading light values");
    } else {
      Serial.print("Ambient: ");
      Serial.print(ambient_light);    // вывод общего уровня освещенности
      Serial.print(" Red: ");
      Serial.print(red_light);    // вывод уровня освещенности в красном спектре
      Serial.print(" Green: ");
      Serial.print(green_light);
      Serial.print(" Blue: ");
      Serial.println(blue_light);
    }

    // Wait 1 second before next reading
    delay(500);
  }
}
```