Подключение беспроводных модулей связи

Павел Пронин
C++ разработчик

Проверка связи

Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включён звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти

Поставьте в чат:

\+ если меня видно и слышно

— если нет

Павел Пронин

О спикере:

- Разработчик на C++ более 8-ми лет
- Опыт в разработке беспилотных автомобилей
- С 2022 года разработчик в компании разработки мобильных игр Playrix (компания разрабатывает такие игры как homescapes и gardegscapes)

Вспоминаем прошлое занятие

Вопрос: Какие линии используются в интерфейсе SPI?

Вспоминаем прошлое занятие

Вопрос: Какие линии используются в интерфейсе SPI?

Ответ: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK или SCK (Serial Clock), CS или SS (Chip Select, Slave Select).

Вспоминаем прошлое занятие

Вопрос: Какие существуют способы подключения нескольких ведомых устройств по SPI?

Вспоминаем прошлое занятие

Вопрос: Какие существуют способы подключения нескольких ведомых устройств по SPI?

Ответ: независимое подключение и каскадное подключение

Вспоминаем прошлое занятие

Вопрос: Какие существуют режимы передачи данных у SD карт?

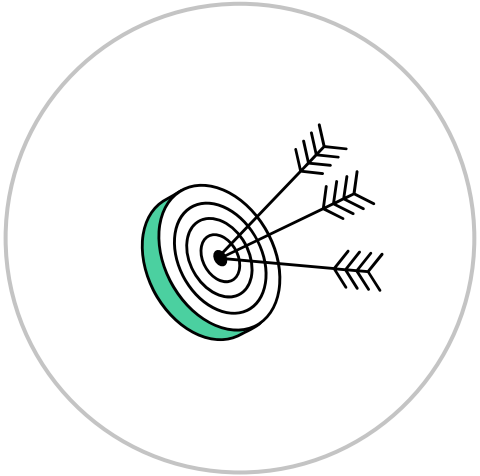
Вспоминаем прошлое занятие

Вопрос: Какие существуют режимы передачи данных у SD карт?

Ответ: SPI режим и SD режим

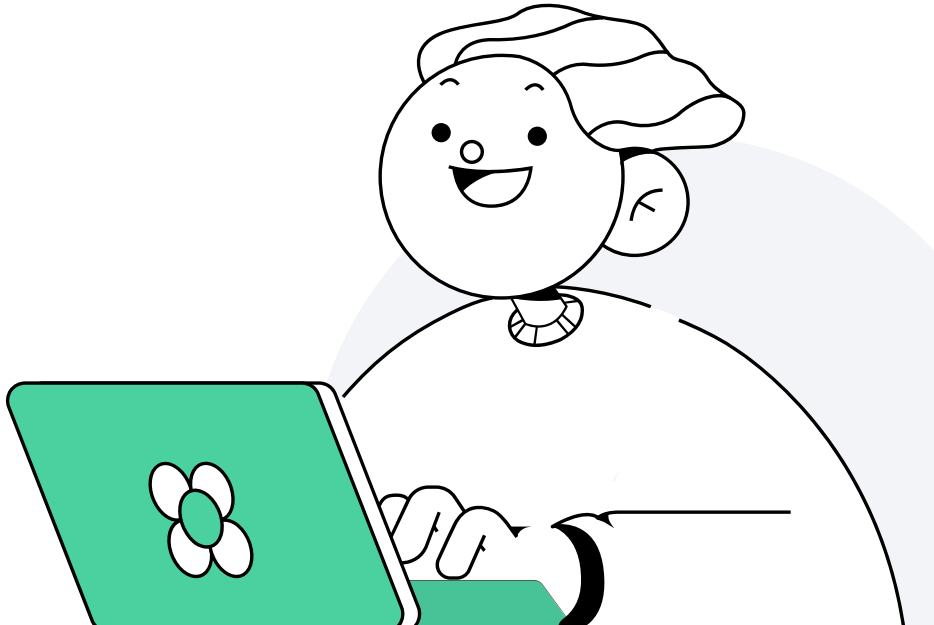
Цели занятия

- Узнаем, какие бывают стандарты беспроводной связи
- Познакомимся с модулем ESP32
- Научимся подключать модуль ESP32 к Wi-Fi сети
- Научимся реализовывать на модуле ESP32 HTTP клиент



План занятия

1. Какие существуют стандарты беспроводной связи
2. Как использовать модуль ESP32
3. Как создать WEB клиент на ESP32
4. Итоги



Какие существуют стандарты беспроводной связи

1

Государственная комиссия по радиочастотам

ГКРЧ — межведомственный координационный орган, действующий при Министерстве цифрового развития, связи и массовых коммуникаций Российской Федерации.

Частотные диапазоны для любительской радиосвязи

ГКРЧ разрешает для любительского использования следующие диапазоны (полный перечень):

- 28,0 - 29,7 МГц - разрешённая мощность до 10 Вт, разрешено использовать рации с любыми типами антенн (компактными, автомобильными, стационарными)
- 430 - 440 МГц - разрешённая мощность до 10 мВт, разрешено использовать радиостанции со встроенными антеннами
- 2300 - 2450 МГц - разрешённая мощность до 100 мВт

Источник

Описание стандарта Bluetooth

Первая официальная версия стандарта была выпущена компанией Ericsson в 1994 году. Разработчики назвали свое изобретение в честь короля Дании Харальда Гормссона по прозвищу «Синезубый», объединившего в 10 веке враждовавшие датские племена в единое королевство.

Источник

Типы устройств с поддержкой Bluetooth

В настоящее время существует два типа устройств с поддержкой Bluetooth:

- **Bluetooth Classic (BR/EDR)**, используется в беспроводных громкоговорителях, автомобильных информационно-развлекательных системах и наушниках;

- **Bluetooth Low Energy (BLE)**, т.е. Bluetooth с низким энергопотреблением, который появился в версии стандарта Bluetooth 4.0.

  Он чаще всего применяется в приложениях, чувствительных к энергопотреблению (например в устройствах с батарейным питанием) или в устройствах, передающих небольшие объемы данных с большими перерывами между передачами (например, разнообразные сенсоры параметров окружающей среды или управляющие устройства, такие как беспроводные выключатели).

Основные параметры BLE

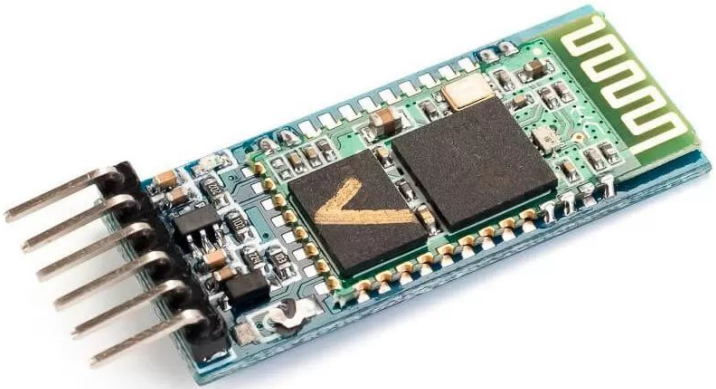
Наиболее важные технические параметры:

- Используемый частотный диапазон 2.400 - 2.4835 ГГц
- Весь частотный диапазон поделен на 40 каналов по 2 МГц каждый.
- Максимальная скорость передачи данных по радиоканалу (начиная с Bluetooth версии 5) 2Мбит/с.
- Дальность передачи сильно зависит от физического окружения, а также используемого режима передачи. Типичная дальность передачи: 10-30 метров.
- Потребление электроэнергии может изменяться в широких пределах. Типичное потребление BLE-трансивера во время передачи данных не превышает 15 мА.
- BLE предназначен для передачи данных по каналу с низкой пропускной способностью.
- Версии Bluetooth (в части BLE) являются обратно совместимыми. Тем не менее возможности связи будут ограничены функциями более старой версии.

Bluetooth модуль HC-05

Контроллер Arduino не поддерживают беспроводную связь. Если она необходима, то можно использовать отдельные модули беспроводной связи. Модуль HC-05 подключается по последовательному порту, а управление им происходить с помощью AT-команд. Полный их перечень приведен по ссылк

Пример команды:
AT+VERSION?\r\n
Ответ:
+VERSION:2.0-20100601
OK

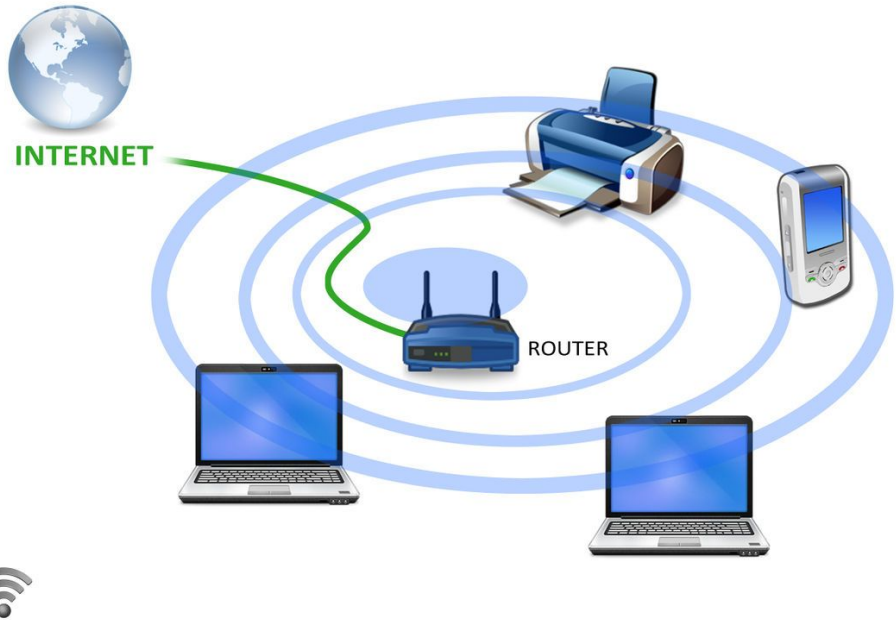


Источник

Описание стандарта Wi-Fi

Технология беспроводной коммуникации была создана в 1990-е в CSIRO (Государственное объединение научных и прикладных исследований, Австралия). «Wi-Fi» — это всего лишь товарный знак, зарегистрированный альянсом Wi-Fi Alliance в 1999 году.

IEEE 802.11 — набор стандартов связи для коммуникации в беспроводной локальной сетевой зоне частотных диапазонов 0,9; 2,4; 3,6; 5 и 60 ГГц.



Источник

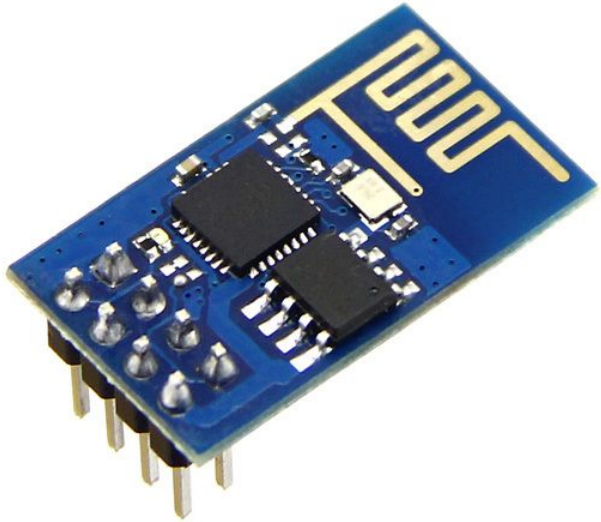
Пример стандартов Wi-Fi

В настоящее время используются следующие стандарты Wi-Fi:

- 802.11 — 1 Мбит/с и 2 Мбит/с, 2,4 ГГц;
- 802.11a — 54 Мбит/с, 5 ГГц;
- 802.11b — 5,5 и 11 Мбит/с, 2,4 ГГц;
- 802.11g — 54 Мбит/с, 2,4 ГГц;
- 802.11n — 600 Мбит/с, 2,4-2,5 ГГц или 5 ГГц;
- ...
- 802.11be — до 30 Гбит/с, 2,4; 5 и 6 ГГц.

Wi-Fi модуль ESP8266

Общение с компьютером или микроконтроллером осуществляется с через UART с помощью набора AT-команд. Кроме того, модуль можно использовать как самостоятельное устройство, для этого необходимо в него загрузить свою прошивку. Полное описание микросхемы доступно по ссылке, частичный перевод описания приведен здесь.

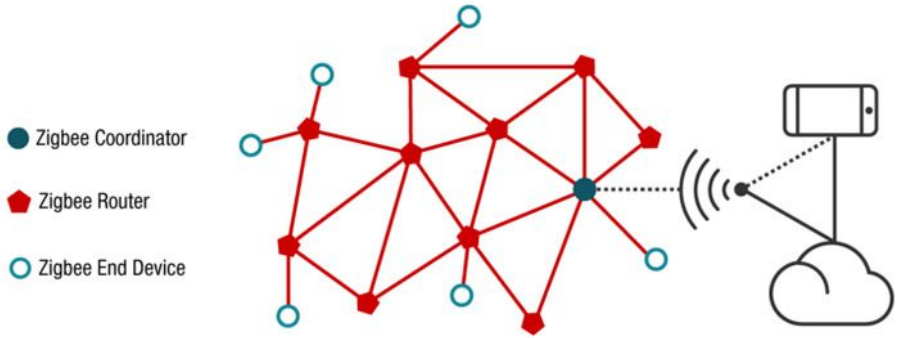


Источник

Описание стандарта ZigBEE

Zigbee — спецификация сетевых протоколов верхнего уровня — уровня приложений APS и сетевого уровня NWK, — использующих сервисы нижних уровней — уровня управления доступом к среде MAC и физического уровня PHY, регламентированных стандартом IEEE 802.15.4. Спецификация Zigbee ориентирована на приложения, требующие гарантированной безопасной передачи данных при относительно небольших скоростях.

Источник



Типы устройств ZigBEE

- Координатор
- Маршрутизатор (роутер)
- Конечное устройство

Координатор — узел, организовавший сеть.
Он выбирает политику безопасности сети, разрешает или запрещает подключение к сети новых устройств, а также при наличии помех в радиоэфире инициирует процесс перевода всех устройств в сети на другой частотный канал.

Маршрутизатор (роутер) — узел, который имеет стационарное питание и следовательно может постоянно участвовать в работе сети.

Координатор также является роутером. На узлах этого типа лежит ответственность по маршрутизации сетевого трафика. Роутеры постоянно поддерживают специальные таблицы маршрутизации, которые используются для прокладки оптимального маршрута и поиска нового, если вдруг какое-либо устройство вышло из строя.

Например, роутерами в сети ZigBee могут быть умные розетки, блоки управления осветительными приборами или любое другое устройство, которое имеет подключение к сети электропитания.

Конечное устройство — это устройство, которое подключается к сети через родительский узел – роутер или координатор – и не участвует в маршрутизации трафика.

Все общение с сетью для них ограничивается передачей пакетов на «родительский» узел либо считыванием поступивших данных с него же. «Родителем» для таких устройств может быть любой роутер или координатор. Конечные устройства большую часть времени находятся в спящем режиме и отправляют управляющее или информационное сообщение. Это позволяет им долго сохранять энергию встроенного источника питания.

Сравнение ZigBEE, Bluetooth и Wi-Fi

Zigbee обеспечивает более дешевое и надежное соединение за счет снижения скорости связи

Технология	Wi-Fi	Bluetooth	ZigBee
Стандарт связи	IEEE 802.11	IEEE 802.15.4	IEEE 802.15.4
Скорость передачи данных	300+ Мбит/с	до 3 Мбит/с	250 Кбит/с
Энергопотребление	Высокое	Низкое	Низкое
Частотный диапазон	2,4 ГГц	2,4 ГГц	2,4 ГГц
Поддержка IP-технологий	+	—	—
Топология	«звезда»	«звезда»	«mesh»

Источник

ZigBEE модуль XBee

Работа с XBee организована как пересылка сообщений через последовательное (serial) соединение.

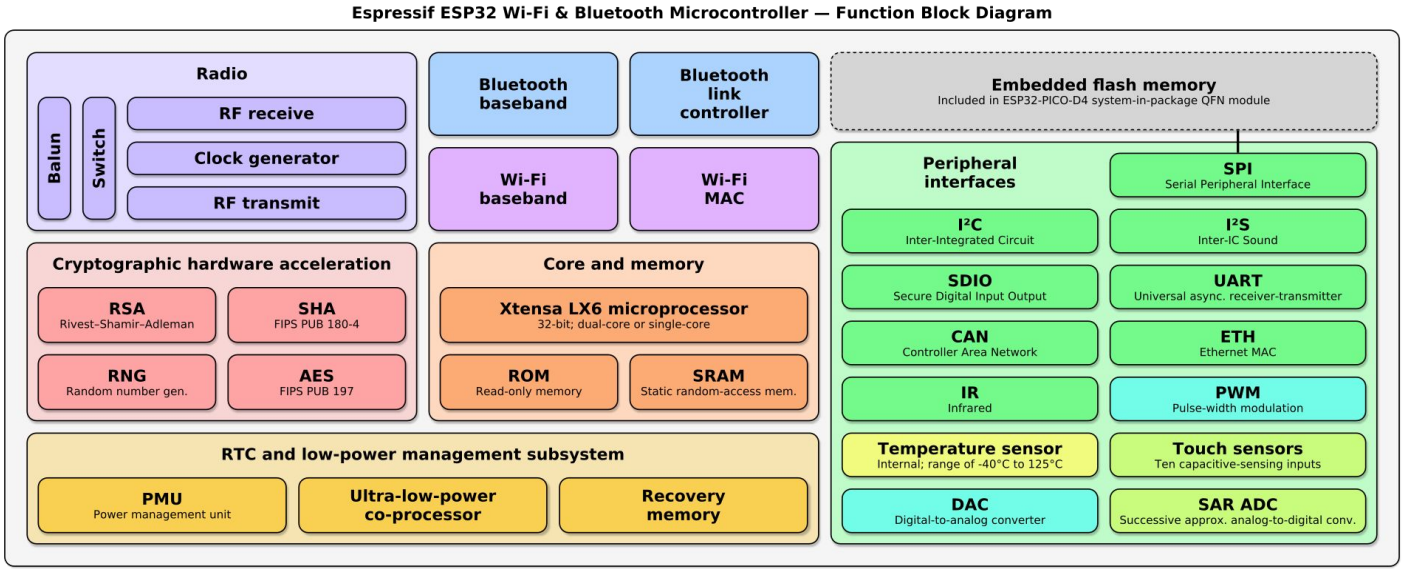
Источник

Как использовать модуль ESP32

2

Микропроцессор ESP32

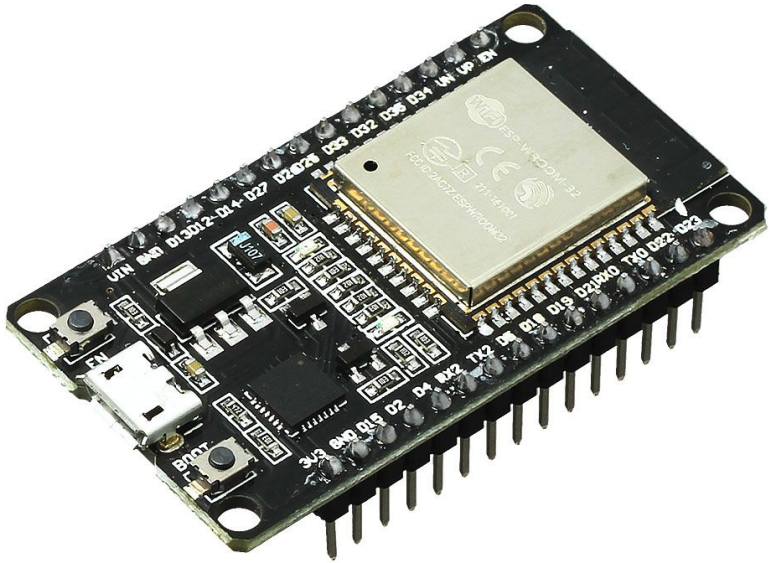
ESP32 — серия недорогих микропроцессоров с малым энергопотреблением китайской компании Espressif Systems. Представляют собой систему на кристалле с интегрированным контроллерами радиосвязи Wi-Fi, Bluetooth и Thread.



Источник

Плата ESP32 DevKit

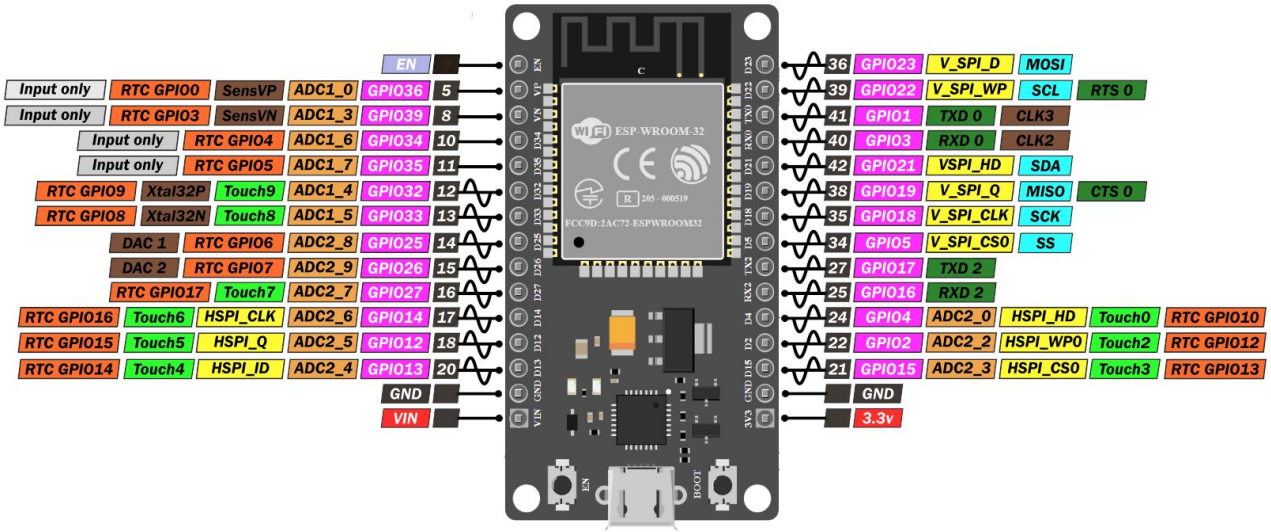
На базе платы ESP32 DevKit можно разрабатывать проекты, где требуется беспроводная передача данных по Wi-Fi и/или Bluetooth.
Эти платы поддерживаются средой разработки Arduino IDE и с ней совместимо большинство библиотек.



Источник

Распиновка ESP32 DevKit

Основное напряжение питания ESP32 DevKit является 3,3 В, а не 5 В. Выходы для логической единицы выдают 3,3 В, а в режиме входа ожидают принимать не более 3,3 В. Более высокое напряжение может повредить микроконтроллер!



Источник

Библиотека WiFi для работы с сетью WiFi

При подключении библиотеки автоматически вызывается конструктор, который создает объект с именем WiFi. Поэтому в пользовательском коде вызывать конструктор не нужно.

int begin(const char* ssid)
int begin(const char* ssid, uint8_t key_idx, const char* key)
int begin(const char* ssid, const char *passphrase)

— инициализирует сетевые настройки библиотеки WiFi и возвращает текущий статус соединения.

Параметры:

- ssid: имя WiFi-сети, к которой необходимо подсоединиться
- key_idx: сети, использующие алгоритм шифрования WEP, могут содержать до 4 различных ключей. Данный параметр позволяет задать, какой из ключей необходимо использовать
- key: шестнадатеричная строка, используемая в качестве ключа безопасности WEP-сети
- passphrase: сети, использующие алгоритм шифрования WPA, используют пароль в виде строки

Возвращаемое значение: WL_CONNECTED - если соединение с сетью установлено
WL_IDLE_STATUS - если включено питание, но соединение с сетью не установлено

Библиотека WiFi для работы с сетью WiFi

int disconnect(void) — отключает устройство от текущей сети
Параметры: нет
Возвращаемое значение: всегда возвращает true

void setDNS(IPAddress dns_server1)
void setDNS(IPAddress dns_server1, IPAddress dns_server2) — позволяет задать адрес DNS-сервера (сервера доменных имен)
Параметры:

- dns_server1, dns_server2: IP-адрес первичного и вторичного DNS-сервера

Возвращаемое значение: нет

char* SSID(uint8_t networkItem) — возвращает SSID-имя текущей сети
Параметры:

- networkItem: (необязательный параметр) позволяет задать сеть, о которой необходимо считать информацию

Возвращаемое значение: SSID-имя сети, с которой установлено соединение

Библиотека WiFi для работы с сетью WiFi

uint8_t* BSSID(uint8_t* bssid) — возвращает MAC-адрес маршрутизатора, с которым установлено соединение
Параметры:

- bssid: массив из 6 байт

Возвращаемое значение: массив из 6 байт, содержащий MAC-адрес маршрутизатора, к которому подключено устройство

int32_t RSSI(uint8_t networkItem) — определяет мощность сигнала, принимаемого от маршрутизатора
Параметры:

- networkItem: (необязательный параметр) позволяет задать сеть, о которой необходимо считать информацию

Возвращаемое значение: текущее значение RSSI (мощность принимаемого сигнала в dBm)

Библиотека WiFi для работы с сетью WiFi

int8_t scanNetworks() — осуществляет поиск WiFi-сетей, доступных в радиусе действия, и возвращает их количество
Параметры:

- networkItem: (необязательный параметр) позволяет задать сеть, о которой необходимо считать информацию

Возвращаемое значение: количество найденных сетей

uint8_t encryptionType(uint8_t networkItem) — возвращает тип шифрования текущей сети
Параметры:

- networkItem: (необязательный параметр) позволяет задать сеть, о которой необходимо считать информацию

Возвращаемое значение: значение, характеризующее тип шифрования: TKIP (WPA) = 2, WEP = 5, CCMP (WPA) = 4, NONE = 7, AUTO = 8

Библиотека WiFi для работы с сетью WiFi

uint8_t getSocket() — возвращает первый доступный сокет
Параметры: нет
Возвращаемое значение: первый доступный сокет

uint8_t* macAddress(uint8_t* mac) — возвращает тип шифрования текущей сети
Параметры:

- mac: массив из 6 байт для хранения MAC-адреса

Возвращаемое значение: 6 байт, содержащие MAC-адрес устройства

uint8_t status() — возвращает статус подключения
Параметры: нет
Возвращаемое значение: WL_CONNECTED – если соединение с WiFi-сетью успешно установлено, WL_NO_SHIELD – если не подключен WiFi-модуль WL_IDLE_STATUS – временный статус, WL_CONNECT_FAILED – соединение не установлено, WL_NO_SSID_AVAIL – нет доступных SSID и др.

Библиотека WiFi для работы с сетью WiFi

IPAddress localIP() — возвращает IP-адрес устройства
Параметры: нет
Возвращаемое значение: IP-адрес устройства в виде объекта IPAddress

IPAddress subnetMask() — возвращает маску подсети для устройства
Параметры: нет
Возвращаемое значение: маска подсети для устройства в виде объекта IPAddress

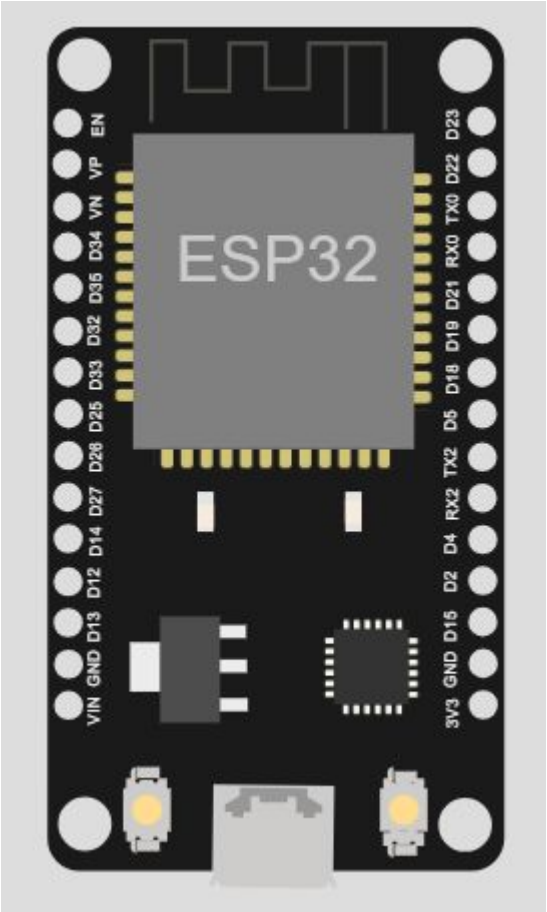
IPAddress subnetMask() — возвращает IP-адрес шлюза для устройства
Параметры: нет
Возвращаемое значение: IP-адрес шлюза для устройства

Подключение модуля ESP32 к сети Wi-Fi

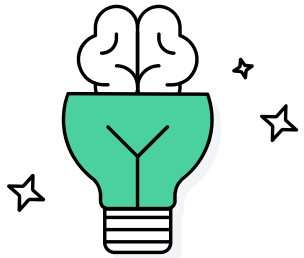
```
#include "WiFi.h"

const char* ssid = "Wokwi-GUEST";
const char* password =  "";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  Serial.print("Hostname: ");
  Serial.println(WiFi.getHostname());

  Serial.print("ESP Mac Address: ");
  Serial.println(WiFi.macAddress());

  Serial.print("Subnet Mask: ");
  Serial.println(WiFi.subnetMask());

  Serial.print("Gateway IP: ");
  Serial.println(WiFi.gatewayIP());
  Serial.print("DNS: ");
  Serial.println(WiFi.dnsIP());
}

void loop() {}
```

Подключение модуля ESP32 к сети Wi-Fi

Имитируется подключение к сети Wokwi-GUEST без пароля



Практическое задание №1



Практика: подключение модуля ESP32 к сети Wi-Fi

Задание:

1) соберите схему в симуляторе WOKWI, состоящую только из платы ESP32;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

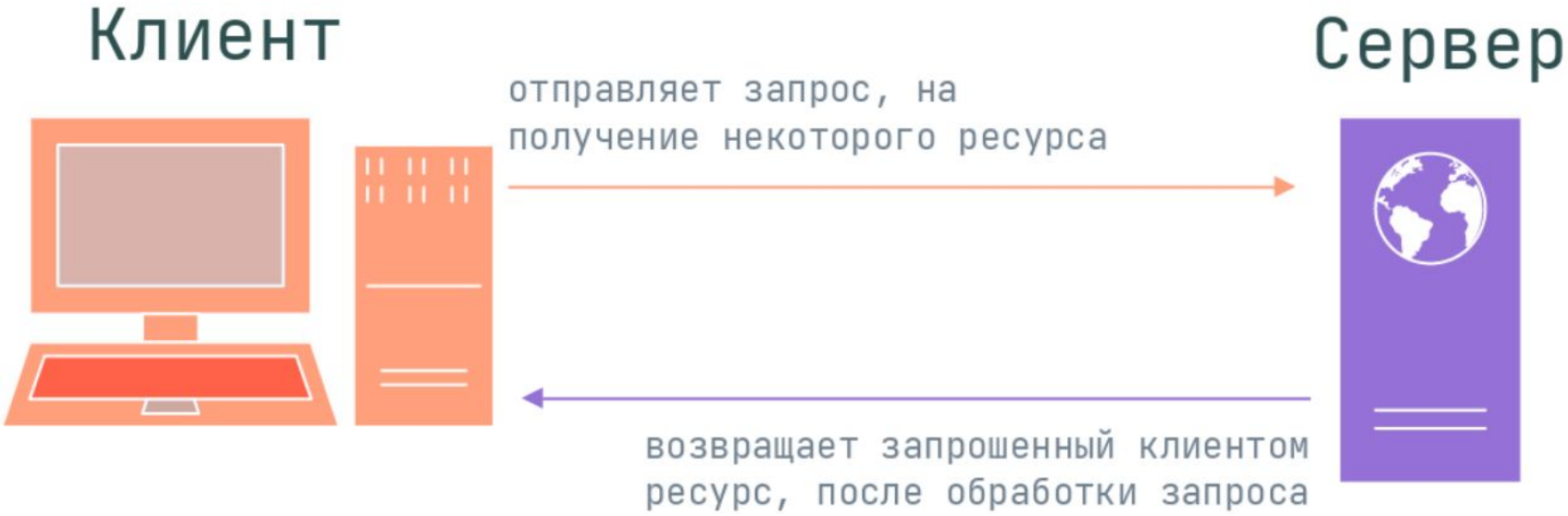
Время выполнения: 5 минут

Как создать WEB клиент на ESP32

3

Протокол HTTP

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.



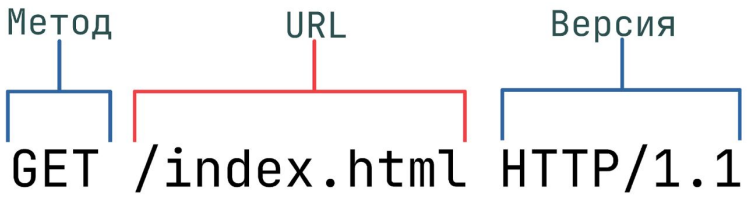
Источник

HTTP-сообщения: запросы и ответы

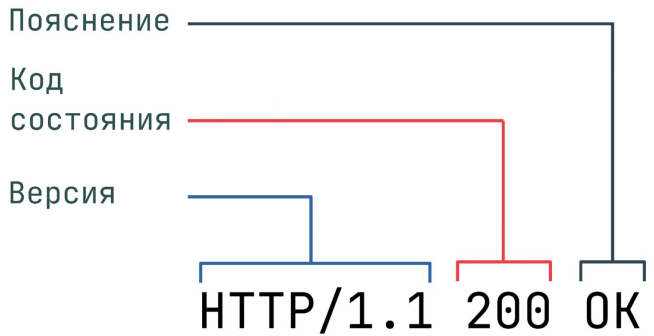
Данные между клиентом и сервером в рамках работы протокола передаются с помощью HTTP-сообщений. Они бывают двух видов:

- Запросы (HTTP Requests) — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок.
- Ответы (HTTP Responses) — сообщения, которые сервер отправляет в ответ на клиентский запрос.

Описание протокола доступно по ссылке, его перевод.



Источник



Библиотека HTTPClient для HTTP протоколом

Библиотека содержит много методов для работы клиента по HTTP протоколу.
Далее приведена только часть из них

HTTPClient() — конструктор, создает объект класса HTTPClient.
Параметры: нет
Возвращаемое значение: нет

bool begin(String url) — инициализирует подключение клиента к заданному серверу.
Параметры:

- url: сетевой адрес сервера

Возвращаемое значение: true - если инициализация прошла успешно, false - в противном случае

Библиотека HTTPClient для HTTP протоколом

int GET() — формирует запрос GET
Параметры: нет
Возвращаемое значение: -1 - нет информации, >0 - код ответа на запрос

String getString(void) — возвращает ответ на запрос GET
Параметры: нет
Возвращаемое значение: строка с текстом ответа

Считывание данных о текущей погоде

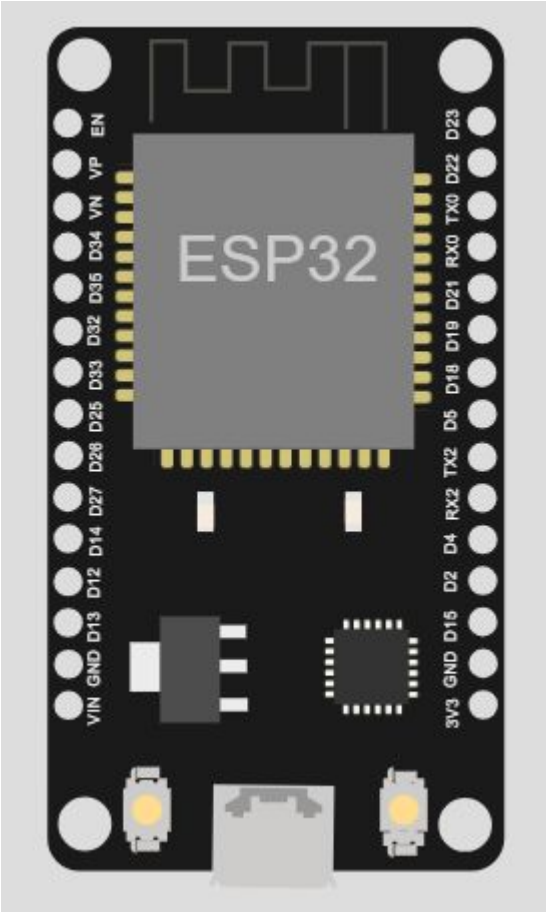
```
#include <WiFi.h>
#include <HTTPClient.h>

// Вводим имя и пароль точки доступа
const char* ssid = "Wokwi-GUEST";
const char* password = "";
//бесплатный сервис получения данных о погоде
const String endpoint = "http://api.openweathermap.org/data/2.5/weather?q=Moscow,ru,pt&APPID=";
//индивидуальный пароль
const String key = "cdecb72a7903eb3bb1964c39615f1764";

void setup()
{
  Serial.begin(115200);
  // делаем небольшую задержку на открытие монитора порта
  delay(1000);
  // подключаемся к Wi-Fi сети
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Соединяемся с Wi-Fi..");
  }
  Serial.println("Соединение с Wi-Fi установлено");
}
```

Считывание данных о текущей погоде

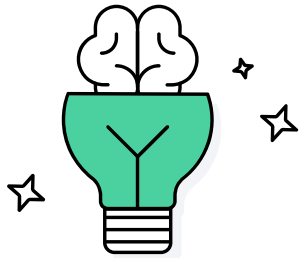
```
void loop()
{
  // выполняем проверку подключения к беспроводной сети
  if ((WiFi.status() == WL_CONNECTED))
  {
    // создаем объект для работы с HTTP
    HTTPClient http;
    // подключаемся к веб-странице OpenWeatherMap с указанными параметрами
    http.begin(endpoint + key);
    int httpCode = http.GET();  // Делаем запрос

    // проверяем успешность запроса
    if (httpCode > 0)
    {
      // выводим ответ сервера
      String payload = http.getString();
      Serial.println(httpCode);
      Serial.println(payload);
    }
    else {
      Serial.println("Ошибка HTTP-запроса");
    }
    http.end(); // освобождаем ресурсы микроконтроллера
  }
  delay(30000);
}
```

Считывание данных о текущей погоде

Имитируется подключение к сети Wokwi-GUEST без пароля



Практическое задание №2



Практика: считывание данных о текущей погоде

Задание:

1) соберите схему в симуляторе WOKWI, состоящую только из платы ESP32;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

Время выполнения: 5 минут

Обработка данных о текущей погоде

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

// Вводим имя и пароль точки доступа
const char* ssid = "Wokwi-GUEST";
const char* password = "";

const String endpoint = "http://api.openweathermap.org/data/2.5/weather?q=Moscow,ru,pt&APPID=";
const String key = "cdecb72a7903eb3bb1964c39615f1764";

void setup()
{

  Serial.begin(115200);
  // делаем небольшую задержку на открытие монитора порта
  delay(1000);

  // подключаемся к Wi-Fi сети
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Соединяемся с Wi-Fi..");
  }
  Serial.println("Соединение с Wi-Fi установлено");
}
```

Обработка данных о текущей погоде

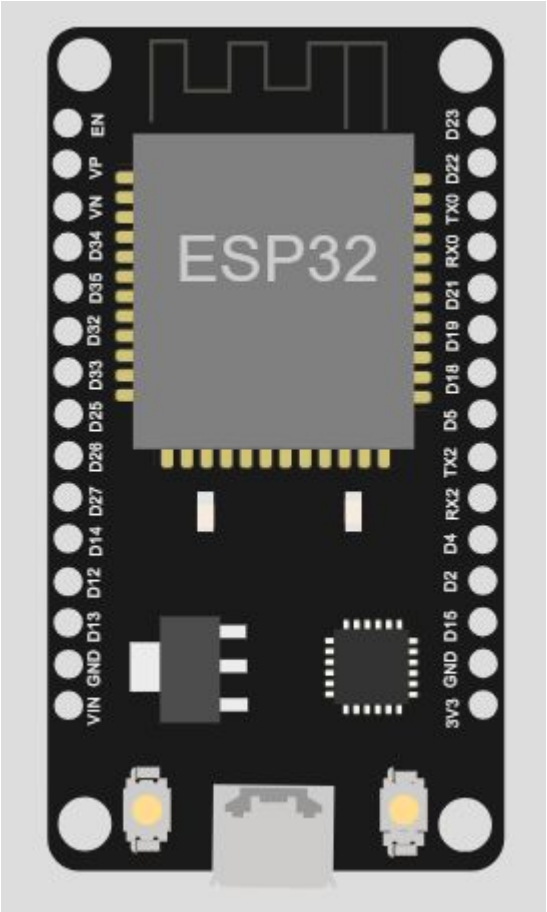
```
void loop() {
  // выполняем проверку подключения к беспроводной сети
  if ((WiFi.status() == WL_CONNECTED))
  {
    // создаем объект для работы с HTTP
    HTTPClient http;
    // подключаемся к веб-странице OpenWeatherMap с указанными параметрами
    http.begin(endpoint + key);
    int httpCode = http.GET();  // Делаем запрос

    // проверяем успешность запроса
    if (httpCode > 0)
    {
      // выводим ответ сервера
      String payload = http.getString();
      Serial.println(httpCode);
      //обработка полученных данных
      handleReceivedMessage(payload);
    }
    else {
      Serial.println("Ошибка HTTP-запроса");
    }
    http.end(); // освобождаем ресурсы микроконтроллера
  }
  delay(30000);
}
```

Обработка данных о текущей погоде

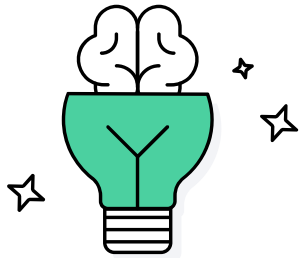
```
void handleReceivedMessage(String message)
{
  StaticJsonDocument<1500> doc;    //Memory pool. Размер с запасом
  //разбор полученного сообщения как форматированного текста JSON
  DeserializationError error = deserializeJson(doc, message);
  // Если разбор прошел успешно
  if (error)
  {
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.c_str());
    return;
  }
  Serial.println();
  Serial.println("----- DATA FROM OPENWEATHER ----");
  const char* name = doc["name"];
  Serial.print("City: ");
  Serial.println(name);
  int timezone = doc["timezone"];
  Serial.print("Timezone: ");
  Serial.println(timezone);
  int humidity = doc["main"]["humidity"];
  Serial.print("Humidity: ");
  Serial.println(humidity);

  Serial.println("-----------------------------");
}
```

Обработка данных о текущей погоде

Имитируется подключение к сети Wokwi-GUEST без пароля



Практическое задание №3



Практика: обработка данных о текущей погоде

Задание:

1) соберите схему в симуляторе WOKWI, состоящую только из платы ESP32;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

Время выполнения: 5 минут

Итоги

4

Итоги занятия

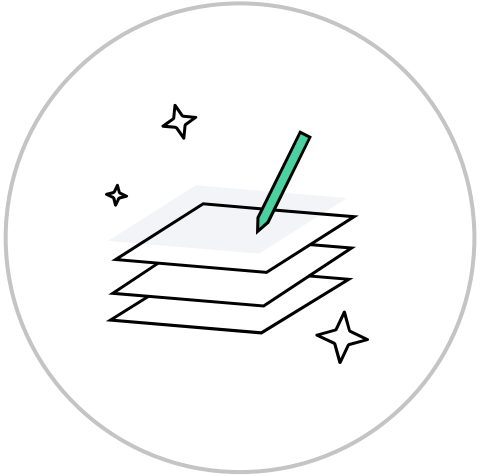
Сегодня мы

1. Узнали основные стандарты беспроводной связи для IoT
2. Познакомились с модулем ESP32
3. Научились подключать модуль ESP32 к сети WiFi
4. Научились реализовывать HTTP клиент на модуле ESP32

Домашнее задание

Давайте посмотрим ваше домашнее задание.

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Задавайте вопросы и пишите отзыв о лекции

Павел Пронин
C++ разработчик