ESP8266

Микросхема ESP8266 – один из самых популярных инструментов для организации беспроводной связи в проектах умного дома. С помощью беспроводного контроллера можно организовывать связь по интерфейсу WiFi, обеспечивая проектам Arduino выход в интернет и возможность дистанционного управления и сбора данных. На основе ESP8266 созданы такие популярные платы как **WeMos** и **NodeMcu**, а также огромное количество самодельных проектов. В этой статье, мы узнаем, что из себя представляет ESP82266, какие бывают ее разновидности, как работать с ESP8266 в среде Arduino IDE.

Описание ESP8266

ESP8266 – микроконтроллер с интерфейсом WiFi, который имеет возможность исполнять программы из флеш-памяти. Устройство было выпущено в 2014 году китайской фирмой Espressif и практически сразу же стало популярным.

Контроллер недорогой, обладает небольшим количеством внешних элементов и имеет следующие технические параметры:

- Поддерживает Wi-Fi протоколы 802.11 b/g/n с WEP, WPA, WPA2;
- Обладает 14 портами ввода и вывода, SPI, I2C, UART, 10-бит АЦП;
- Поддерживает внешнюю память до 16 МБ;
- Необходимое питание от 2,2 до 3,6 В, потребляемый ток до 300 мА в зависимости от выбранного режима.

Важной особенностью является отсутствие пользовательской энергонезависимой памяти на кристалле. Программа выполняется от внешней SPI ПЗУ при помощи динамической загрузки необходимых элементов программы. Доступ к внутренней периферии можно получить не из документации, а из API набора библиотек. Производителем указывается приблизительное количество ОЗУ – 50 кБ.

Особенности платы ESP8266

- Удобное подключение к компьютеру – через USB кабель, питание от него же;
- Наличие встроенного преобразователя напряжения 3,3В;
- Наличие 4 Мб флеш-памяти;
- Встроенные кнопки для перезагрузки и перепрошивки;
- Все порты выведены на плату на две гребенки с шагом 2,5 мм.

Сферы применения модуля ESP8266

- Автоматизация;
- Различные системы для умного дома: Беспроводное управление, беспроводные розетки, управление температурой, дополнение к сигнализационным системам;
- Мобильная электроника;
- ID метки;
- Детские игрушки;
- Mesh-сети.

AT-команды

Когда модуль подключён к терминалу компьютера, мы можем отправить самую простую команду - "**AT**". В ответ на неё модуль должен отправить ответ "**OK**".

Синтаксис AT-команд:

Тип	Формат	Описание
Тест	AT+<x>=?	Запрос параметров и диапазона возможных значений
Запрос	AT+<x>?	Запрос текущих значений параметров
Установка	AT+<x>=<...>	Установка значений параметров
Выполнение	AT+<x>	Выполнение команд

Все команды заканчиваются символами "\r\n".

Основные AT-команды:

Команда	Описание
AT	Пишет в ответ "OK"
AT+RST	Перезапускает модуль ESP8266
AT+GMR	Возвращает версию SDK модуля и процессора AT команд. Пример: AT version:0.21.0.0 SDK version:0.9.5
AT+GLSP=<время>	Включение режима сна на указанное число миллисекунд. Модуль проснётся через указанное время.
ATE[0|1]	Отправка полученных AT команд обратно в терминал. ATE0 - эхо выключено ATE1 - эхо включено
AT+RESTORE	Восстановление значение по умолчанию из флеш-памяти
AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>	Настройка режима работы UART
AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>	То же, что и AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
AT+SLEEP?	Получить текущий режим сна
AT+SLEEP=<sleep mode>	Режим сна: • 0 — режим сна выключен • 1 — режим неглубокого сна • 2 — режим модемного сна

Распиновка esp8266

ESP-01

ESP-02

ESP-03

ESP-04

Существует огромное количество разновидностей модуля ESP8266. На рисунке представлены некоторые из них. Наиболее популярным вариантом является **ESP 01**

ESP-05

ESP-06

ESP-07

ESP-08

Исполнение программы требуется задавать состоянием портов GPIO0, GPIO2 и GPIO15, когда заканчивается подача питания. Можно выделить 2 важных режима – когда код исполняется из UART (GPIO0 = 0, GPIO2 = 1 и GPIO15 = 0) для перепрошивки флеш-карты и когда исполняется из внешней ПЗУ (GPIO0 = 1, GPIO2 = 1 и GPIO15 = 0) в штатном режиме.

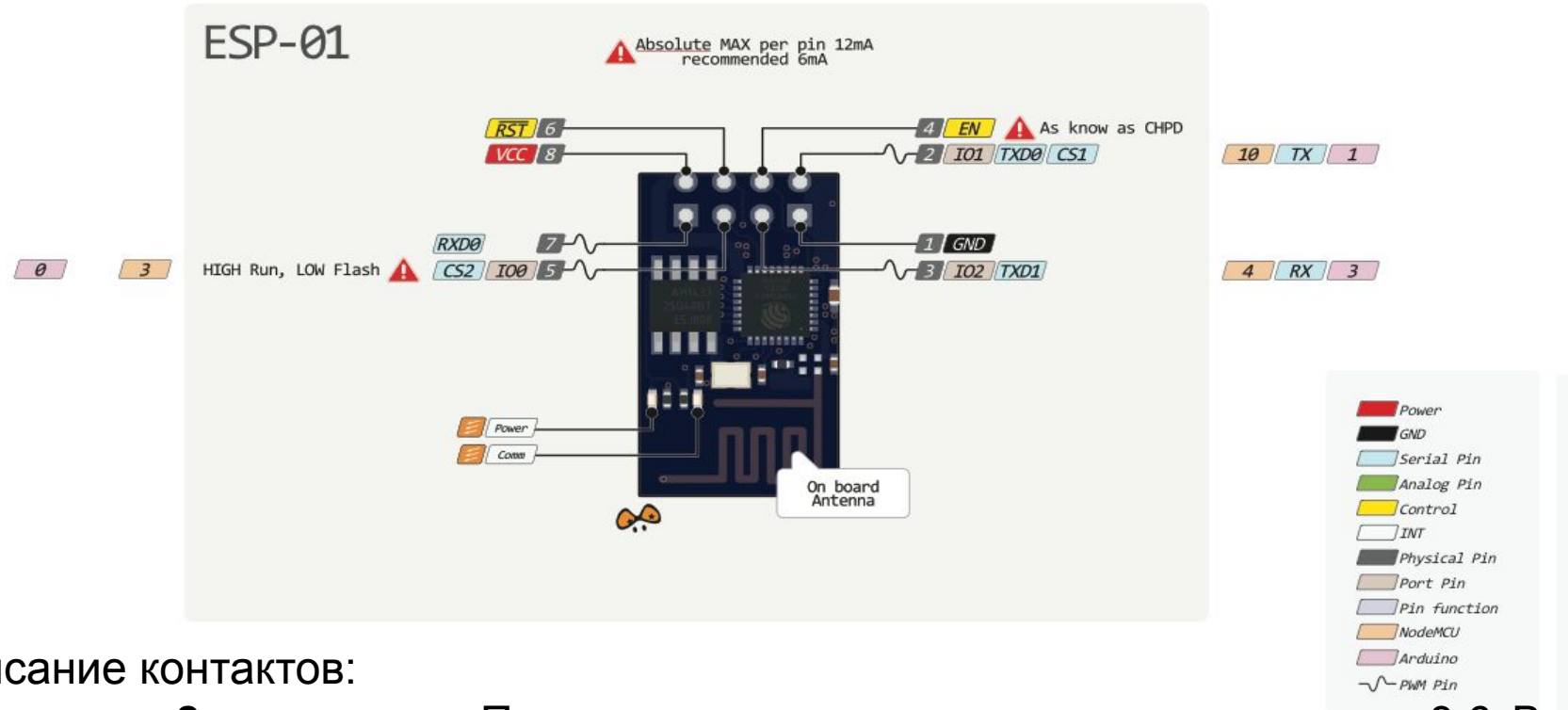
ESP-09

ESP-10

ESP-11

ESP8266 Antenna

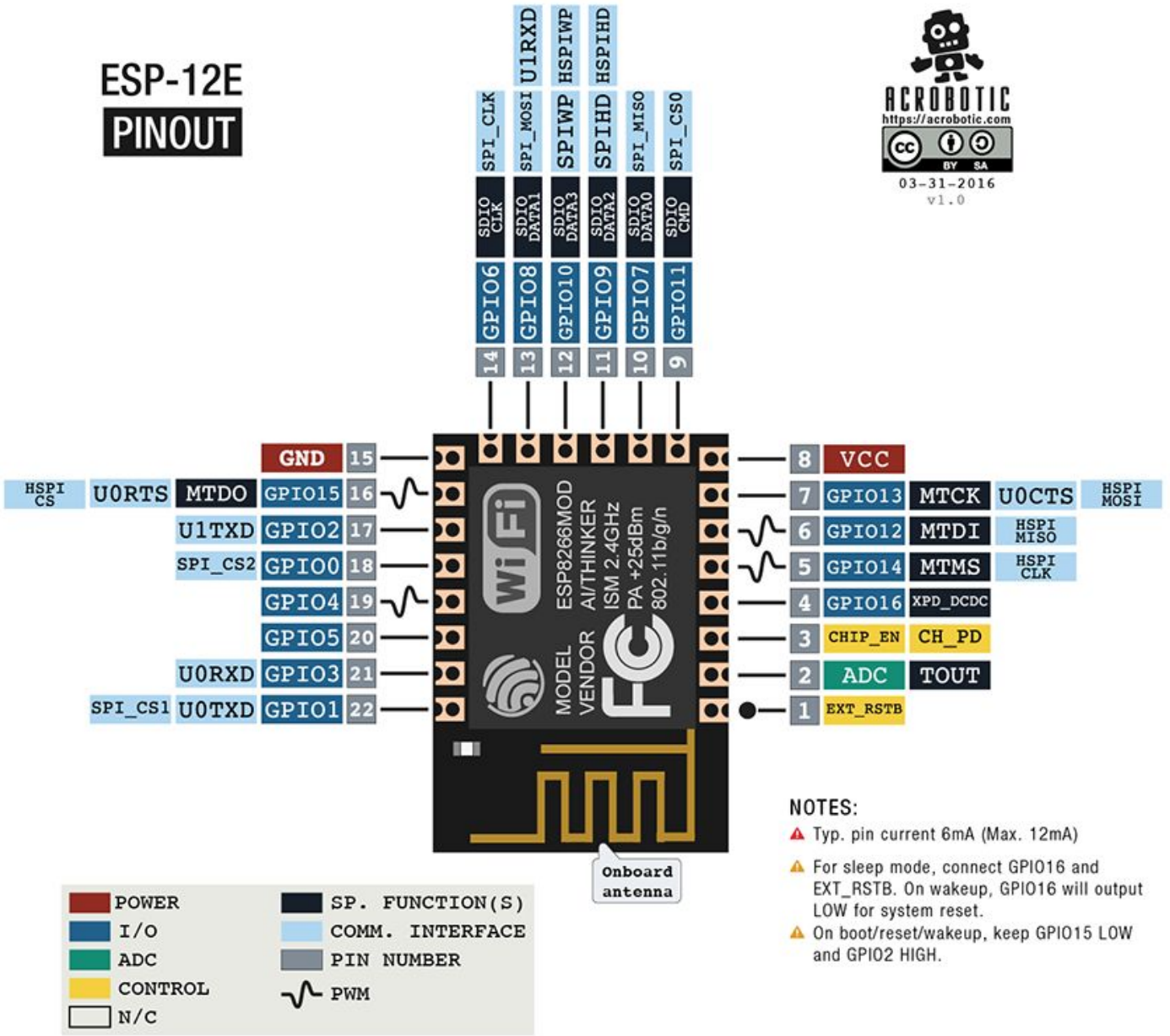
Распиновка для **ESP01** изображена на картинке.



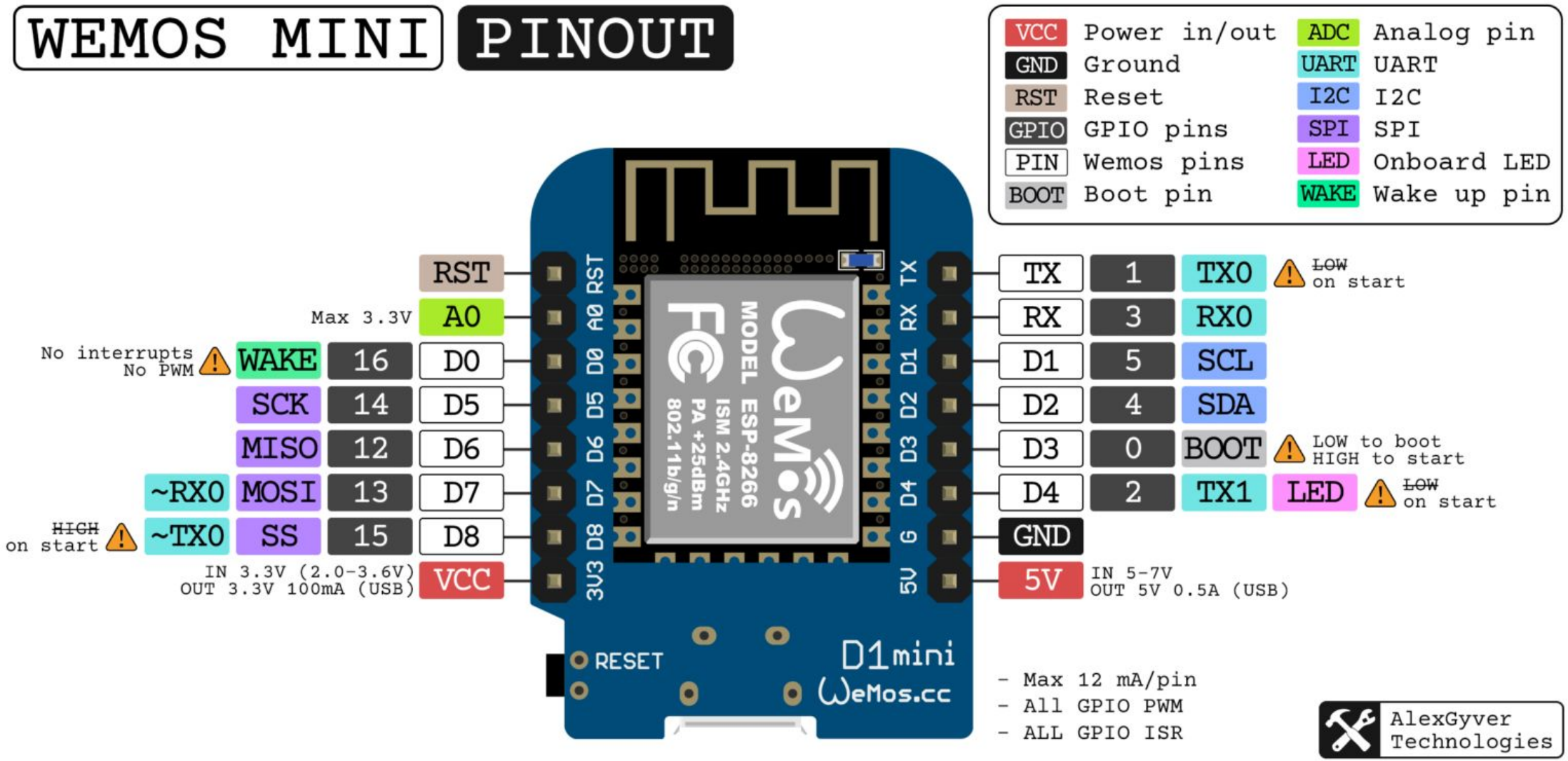
Описание контактов:

- **1 – земля, 8 – питание.** По документации напряжение подается до 3,6 В – это важно учесть при работе с Ардуино, на которую обычно подают 5 В.
- **6 – RST**, нужна для перезагрузки микроконтроллера при подаче на него низкого логического уровня.
- **4 – CP_PD**, также используется для перевода устройства в энергосберегающий режим.
- **7 и 0 – RXD0 и TXD0**, это аппаратный UART, необходимый для перепрошивки модуля.
- **2 – TXD0**, к этому контакту подключается светодиод, который загорается при низком логическом уровне на GPIO1 и при передаче данных по UART.
- **5 – GPIO0**, порт ввода и вывода, также позволяет перевести устройство в режим программирования (при подключении порта к низкому логическому уровню и подачи напряжения) .
- **3 – GPIO2**, порт ввода и вывода.

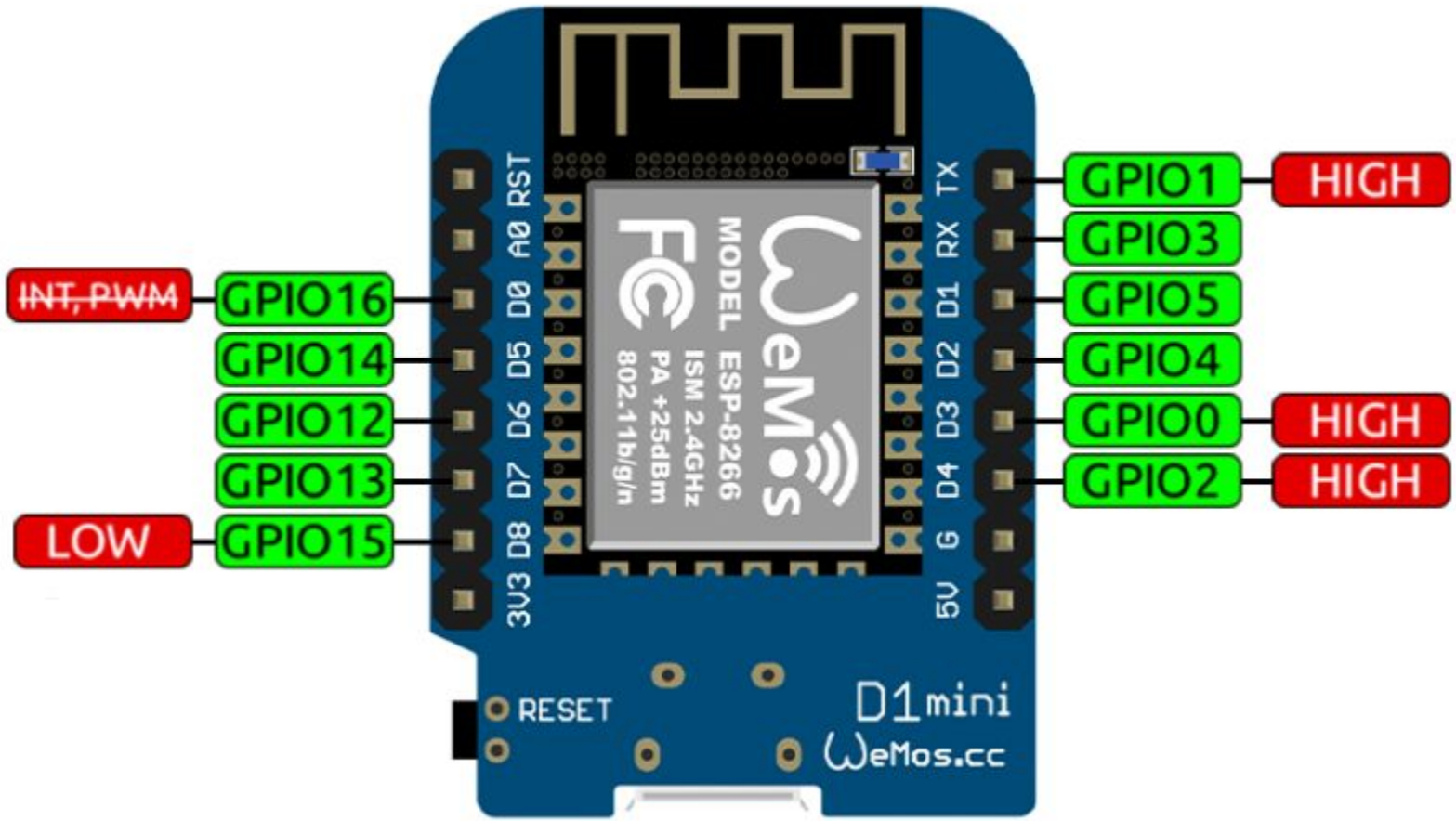
Распиновка **ESP-12**



Распиновка **Wemos Mini**



Особенности пинов



Основные отличия Ардуино от ESP8266

•ESP8266 имеет больший объем флеш-памяти, при этом у ESP8266 отсутствует энергонезависимая память;
•Процессор ESP8266 быстрее, чем у Ардуино;
•Наличие Wi-Fi у ESP8266;
•ESP8266 потребляен больше тока, чем для Ардуино;

Программирование ESP8266 в Arduino IDE

Программный комплект разработчика esp8266 включает в себя:
•Компилятор из пакета GNU Compiler Collection.
•Библиотеки, стеки протоколов WiFi, TCP/IP.
•Средство загрузки информации в программу контроллера.
•Операционная IDE.

Изначально модули ESP8266 поставляются с прошивкой от фирмы-изготовителя. С ее помощью можно управлять модулем с внешнего микроконтроллера, реализовывать работу с Wi-Fi как с модемом. Также существует множество других готовых прошивок. Некоторые из них позволяют настраивать работу модуля при помощи WEB-интерфейса.

Можно программировать из среды Arduino IDE. При ее помощи можно легко писать скетчи и загружать их в ESP8266, прошивать ESP8266, при этом не требуется сама плата Ардуино. Arduino IDE поддерживает все виды модулей ESP8266.

Отличия от AVR Arduino

Деление на 0

В отличие от AVR, деление на 0 приводит к критической ошибке и перезагрузке микроконтроллера. Стараемся этого избегать.

min() и max()

В ядре esp8266 функции `min()` и `max()` реализованы как функции, а не как макросы, поэтому должны использоваться с данными одного типа. Использование переменных разного типа приведёт к ошибке компиляции.

map()

В функции `map(val, min, max, to min, to max)` нет защиты от деления на 0, поэтому если min равен max – микроконтроллер зависнет и перезагрузится. Если min и max задаются какими-то внешними условиями – проверяйте их равенство вручную и исключайте вызов `map()` с такими аргументами.

Типы данных

- Тип `int` является синонимом `long` (`int32_t`) и занимает 4 байта. В AVR `int` это `int16_t`, то есть 2 байта.
- Тип `char` является синонимом `byte` – принимает значения 0.. 255 в отличие от -128.. 127 в AVR.
- Тип `double` имеет полную двойную точность – 8 байт. В AVR это 4 байта.
- Указатель занимает 4 байта, так как область памяти тут 32-битная. В AVR – 2 байта.

Функция analogRead()

ESP8266 имеет крайне убогий одноканальный АЦП.

- Сам АЦП в esp8266 может измерять напряжение в диапазоне 0.. 1.0V. На платах (NodeMCU, Wemos Mini) стоит делитель напряжения, который расширяет диапазон до более удобных 3.3V.
- Разрешение – 10 бит, т. е. значения 0.. 1023 как на Arduino
- Частый вызов `analogRead()` замедляет работу WiFi. При вызовах чаще нескольких миллисекунд WiFi полностью перестаёт работать.
- Результат `analogRead()` имеет кеширование до 5 мс, то есть полученные данные могут запаздывать на это время.
- АЦП может использоваться для измерения напряжения питания МК: для этого нужно вызвать `ADC_MODE(ADC_VCC);` до `void setup()`, а само напряжение питания можно получить из `ESP.getVcc()`.

Функция analogWrite()

- Работает на всех пинах, кроме GPIO16.
- Разрядность ШИМ по умолчанию 8 бит (0.. 255) на версиях ядра 3.x. **На ранних версиях – 10 бит (0.. 1023)**. Скажем спасибо индусам за совместимость.
  - Разрядность можно настроить в `analogWriteResolution(4...16 бит)`.
- Частота ШИМ по умолчанию 1 кГц.
  - Частоту можно настроить в `analogWriteFreq(100.. 40000 Гц)`.
- ШИМ реализован программно, поэтому на повышенной частоте и разрядности будет тормозить выполнение программы!

Аппаратные прерывания

- Настраиваются точно так же, через `attachInterrupt()`.
- Работают на всех пинах, кроме GPIO16.
- Функция-обработчик должна быть объявлена с атрибутом `IRAM_ATTR`:

```
void setup() {
attachInterrupt(1, myIsr, RISING);
}
IRAM_ATTR void myIsr() {
}
```

Либо с `ICACHE_RAM_ATTR` (на старых версиях ядра), вот так:

```
void ICACHE_RAM_ATTR myIsr() {
}
void setup() {
attachInterrupt(1, myIsr, RISING);
}
```

- В обработчике нельзя использовать динамическое выделение и перераспределение памяти (new, malloc, realloc), соответственно *менять* String-строки тоже нельзя.
- В прерывании нельзя использовать задержки.

Функция yield()

В реализации esp8266 функция `yield()` выполняет другую задачу и использовать её как на AVR не получится

EEPROM

EEPROM в esp8266 является эмуляцией из Flash памяти, поэтому мы можем выбрать нужный размер.

- Перед началом работы нужно вызвать `EEPROM.begin(4.. 4096)` с указанием размера области памяти в байтах.
- Для применения изменений в памяти нужно вызвать `EEPROM.commit()`.
- В некоторых версиях SDK отсутствует `EEPROM.update()` и `EEPROM.length()`.
- У Flash памяти небольшой ресурс – всего около 10’000 перезаписей. У фирменной памяти Winbond (можно найти на некоторых моделях ESP-12 и прочих) – около 50’000 перезаписей.

В остальном работа с библиотекой EEPROM.h ничем не отличается.

Важно: EEPROM реализован следующим образом: после запуска `EEPROM.begin(4.. 4096)` содержимое EEPROM указанного размера **дублируется в оперативной памяти**. После **любого изменения** и вызова `EEPROM.commit()` **стирается весь блок Flash памяти** (4 кБ) и записывается заново. Таким образом ресурс “EEPROM” памяти у ESP вырабатывается довольно быстро и **весь сразу**, а не по ячейкам.

Serial (UART)

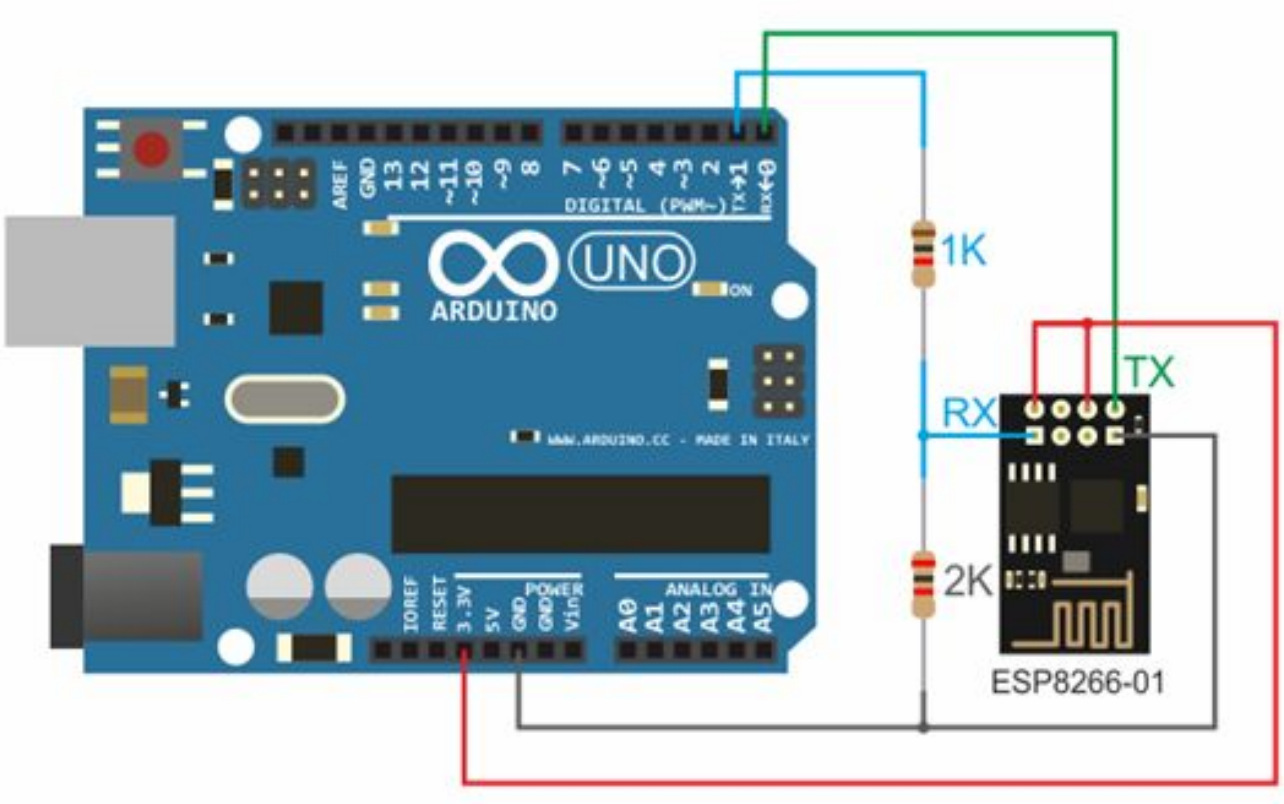
- В отличие от реализации для AVR, можно изменить размер буфера на приём: `Serial.setRxBufferSize(размер)` в байтах. Вызывать **перед** `Serial.begin()`, по умолчанию 256 байт.
- Можно настроить работу только на приём или только на отправку для освобождения пина: `Serial.begin(скорость, SERIAL_8N1, режим)`, где режим:
  - `SERIAL_TX_ONLY` – только отправка
  - `SERIAL_RX_ONLY` – только приём
  - `SERIAL_FULL` – приём и отправка (по умолчанию)
- Можно перенести Serial на другие пины при помощи `Serial.swap()`, вызывать **после** `Serial.begin()`. Пины переместятся на GPIO15/D8 (TX) и GPIO13/D7 (RX). Если вызвать ещё раз – переместятся обратно на GPIO1 (TX) и GPIO3 (RX). И так по кругу.
- У esp8266 есть второй аппаратный UART, но его приёмная нога (RX) занята одним из пинов для работы с памятью и не выведена на плате Wemos Mini. Нога TX находится на GPIO2/D4, то есть можно работать только на отправку, но на практике и это может пригодиться. В программе просто работаем с объектом `Serial1`, настроив его только на отправку.

В настоящий момент для **ESP8266** можно реализовать следующие функции:

- **Основные функции языка Wiring**. Управлять портами GPIO можно точно так же, как и пинами на плате Ардуино: pinMode, digitalRead, digitalWrite, analogWrite. Команда analogRead(A0) позволяет считать значения АЦП. При помощи команды analogWrite (pin, value) можно подключить ШИМ на нужном выходе GPIO. При value=0 ШИМ отключается, максимальное значение достигает константы, равной 1023. С помощью функций attachInterrupt, detachInterrupt можно выполнять прерывание на любом порте GPIO, кроме 16.
- **Тайминг и delay**. Используя команды millis и micros можно вернуть мс и мкс, которые прошли с момента старта. Delay позволяет приостановить исполнение программы на нужное время. Также функция delay(…) позволяет поддерживать нормальную работу Wi-Fi, если в скетче присутствуют большие элементы, которые выполняются более 50 мс. Yield() – аналог функции delay(0).
- **Serial и Serial1 (UART0 и UART1).** Работа Serial на ESP8266 аналогична работе на ардуино. Запись и чтение данных блокируют исполнение кода, если FIFO на 128 байт и программный буфер на 256 байт заполнены. Объект Serial пользуется аппаратным UART0, для него можно задать пины GPIO15 (TX) и GPIO13 (RX) вместо GPIO1(TX) и GPIO3(RX). Для этого после функции Serial.begin(); нужно вызвать Serial.swap();. Аналогично Serial1 использует UART1, который работает на передачу. Необходимый пин для этого GPIO2.
- **Макрос PROGMEM**. Его работа аналогична работе в Ардуино. Позволяет перемещать данные read only и строковые постоянные во flash-память. При этом в ESP8266 не сохраняются одинаковые константы, что приводит к дополнительной трате флеш-памяти.
- **I2C.** Перед началом работы с шиной I2C выбираются шины с помощью функции Wire.pins(int sda, int scl).
- **SPI, OneWire** – поддерживаются полностью.

Использование esp8266 для связи Ардуино по WiFi

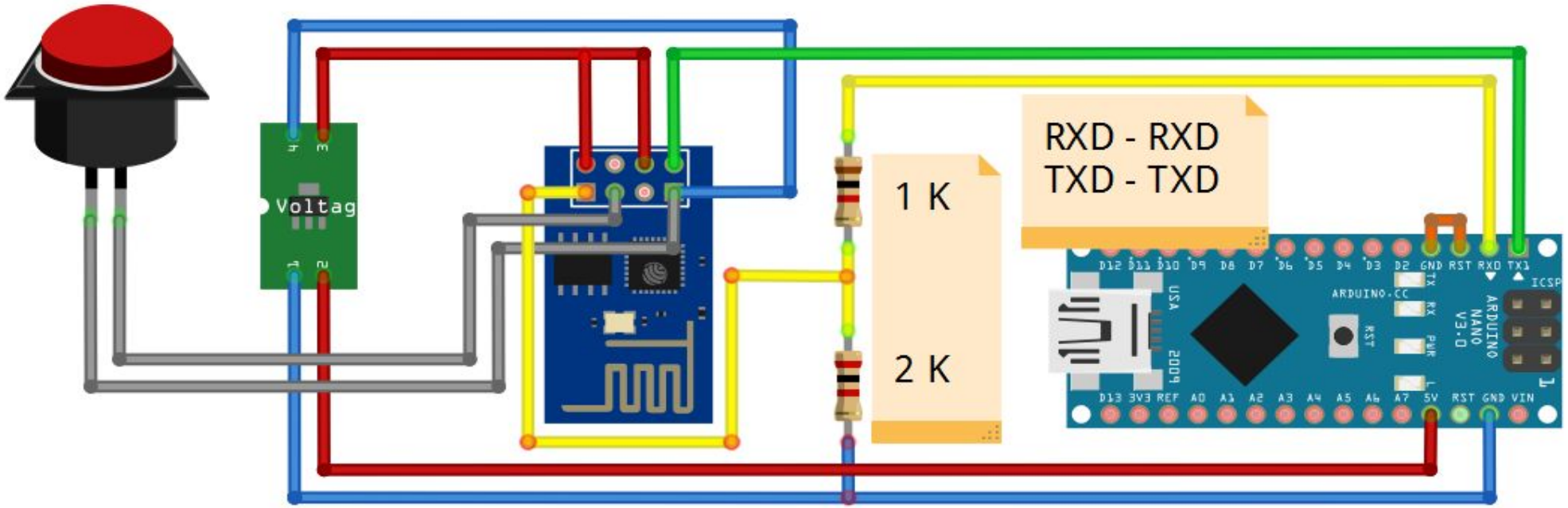
Перед подключением к Ардуино важно помнить, что у **ESP8266** напряжение питания не может быть выше 3,6, в то время как на пате Ардуино напряжение равно 5 В. Соединять 2 микроконтроллера нужно с помощью резистивных делителей. Перед подключением модуля нужно ознакомиться с распиновкой выбранного **ESP8266**. Схема подключения для **ESP8266-01** представлена на рисунке.



3,3 В с Ардуино – на **Vcc&CH_PD** на модуле **ESP8266**, **GND** с Ардуино – к **GND** с ESP8266, **0 – TX, 1 – RX**.

Для поддержки стабильной работы **ESP8266** необходим источник постоянного напряжения на 3,3 В и максимальный ток 250 мА. Если питание происходит от конвертера USB-TTL, могут происходить неполадки и сбои в работе.

Схема подключения ESP8266 к Arduino Nano



fritzing

Работа с библиотекой Wi-Fi для ESP8266 схожа с библиотекой для обыкновенного шилда. Имеется несколько особенностей:

- **mode(m)** – для выбора одного из трех режимов: клиент, точка доступа или оба режима единовременно.
- **softAP(ssid)** – нужен для создания открытой точки доступа.
- **softAP(ssid, password)** – создает точку доступа с паролем, который должен состоять не менее чем из 8 знаков.
- **WiFi.macAddress(mac) и WiFi.softAPmacAddress(mac)–** определяет MAC адрес.
- **WiFi.localIP() и WiFi.softAPIP()** – определение IP адреса.
- **printDiag(Serial);** – позволят узнать данные о диагностике.
- **WiFiUDP** – поддержка передачи и приема multicast пакета в режиме клиента.

Работа выполняется по следующему алгоритму:

- Подключение USB-TTL к USB и к ESP.
- Запуск Arduino IDE.
- Выбрать в меню инструменты нужный порт, плату, частоту и размер flash-памяти.
- Файл — Примеры — ESP8266WiFi — WiFiWebServer.
- Записать в скетче SSID и пароль сети Wi-Fi.
- Начать компиляцию и загрузку кода.
- Дождаться окончания процесса прошивки, отсоединить GPIO0 от земли.
- Поставить скорость 115200.
- Произойдет подключение, будет записан адрес IP.
- Открыть браузер, ввести в адресной строке номер IP/gpio/1
- Посмотреть монитор порта, если к выходу GPIO2 подключен светодиод, он должен загореться.

NodeMCU на базе esp8266

NodeMCU – это платформа, основанная на базе модуля esp8266. Используется для управления схемой на расстоянии при помощи интернета через Wi-Fi. Плата малогабаритная, компактная, стоит дешево, на лицевой стороне имеется разъем для USB. Рядом кнопки для отладки и перезагрузки микроконтроллера. Также установлен чип ESP8266. Напряжение питания – от 5 до 12 В, желательно подавать более 10 В.

Большим преимуществом платы является ее малое энергопотребление. Нередко их используют в схемах с автономным питанием. На плате расположены всего 11 портов общего назначения, из них некоторые имеют специальные функции:

- D1 и D2 – для интерфейса I2C/ TWI;
- D5-D8- для интерфейса SPI;
- D9, D10 – для UART;
- D1-D10 – могут работать как ШИМ.

Платформа имеет современное API для аппаратного ввода и вывода. Это позволяет сократить количество действий во время работы с оборудованием и при его настройке. С помощью прошивки NodeMCU можно задействовать весь рабочий потенциал для быстрой разработки устройства.

WeMos на базе esp8266

WeMos – еще один вид платформы, основанный на базе микроконтроллера esp8266. Соответственно, имеется Wi-Fi модуль, поддерживается Arduino IDE, имеется разъем для внешней антенны. Плата имеет 11 цифровых входов/выходов, которые (кроме D0) поддерживают interrupt/pwm/I2C/one-wire. Максимальное напряжение питания достигает 3,3 В. Также на платформе присутствует USB разъем. Аналоговый вход 1 с максимальным напряжением 3,2В.

Для работы с модулем нужно установить драйвер CH340 и настроить Ардуино IDE под ESP8266. Для этого нужно в меню настройки в строке «дополнительная ссылка для менеджера плат» добавить адрес http://arduino.esp8266.com/stable/package_esp8266com_index.json.

После этого требуется найти пакет esp8266 by ESP8266 и установить его. Затем нужно выбрать в меню инструменты микроконтроллер Wemos D1 R2 и записать нужный скетч.

Выводы по ESP8266

С помощью плат на основе микросхемы ESP8266 вы можете добавить в свои проекты возможности “большого интернета”, сделав их гораздо более интеллектуальными. Дистанционное управление, сбор и анализ данных на сервере, обработка голоса и работа с изображением – все это становится доступным, когда мы подключаем наш проект по WiFi к интернету. В следующих статьях мы подробно рассмотрим то, как можно программировать устройства на базе **esp8266**, а также уделим внимание таким популярным платам как **WeMos** и **NodeMcu**.