Лекция 10

**ESP8266** — [микроконтроллер](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80) ([SoC](https://ru.wikipedia.org/wiki/SoC)) [китайского производителя](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%9A%D0%9D%D0%A0) [Espressif Systems](https://ru.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/Espressif_Systems) с интерфейсом [Wi-Fi](https://ru.wikipedia.org/wiki/Wi-Fi). Помимо Wi-Fi, микроконтроллер отличается отсутствием флеш-памяти на кристалле[[1]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-ds-1), программы пользователя исполняются из внешней [флеш-памяти](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D0%B5%D1%88-%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) с интерфейсом [SPI](https://ru.wikipedia.org/wiki/Serial_Peripheral_Interface).

Микроконтроллер привлёк внимание в 2014 году в связи с выходом первых продуктов на его базе по необыкновенно низкой цене.

Весной 2016 года началось производство ESP8285, совмещающей ESP8266 и флеш-память на 1 МБайт. Осенью 2015 года Espressif Systems представила развитие линейки — микросхему [ESP32](https://ru.wikipedia.org/wiki/ESP32) и модули на её основе[[2]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-2).

**Содержание**

* [1Микроконтроллер](https://ru.wikipedia.org/wiki/ESP8266#%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80)
	+ [1.1ESP8285](https://ru.wikipedia.org/wiki/ESP8266#ESP8285)
	+ [1.2ESP32](https://ru.wikipedia.org/wiki/ESP8266#ESP32)
* [2Средства разработки](https://ru.wikipedia.org/wiki/ESP8266#%D0%A1%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)
	+ [2.1Прошивки](https://ru.wikipedia.org/wiki/ESP8266#%D0%9F%D1%80%D0%BE%D1%88%D0%B8%D0%B2%D0%BA%D0%B8)
	+ [2.2Начальная загрузка и обновление прошивки](https://ru.wikipedia.org/wiki/ESP8266#%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%B8_%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D1%88%D0%B8%D0%B2%D0%BA%D0%B8)
		- [2.2.1Обновление через Wi-Fi](https://ru.wikipedia.org/wiki/ESP8266#%D0%9E%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_Wi-Fi)
	+ [2.3Утилиты](https://ru.wikipedia.org/wiki/ESP8266#%D0%A3%D1%82%D0%B8%D0%BB%D0%B8%D1%82%D1%8B)
* [3Сетевая инфраструктура](https://ru.wikipedia.org/wiki/ESP8266#%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%BD%D1%84%D1%80%D0%B0%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0)
	+ [3.1Wi-Fi](https://ru.wikipedia.org/wiki/ESP8266#Wi-Fi)
	+ [3.2Локальная сеть](https://ru.wikipedia.org/wiki/ESP8266#%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C)
	+ [3.3Интернет](https://ru.wikipedia.org/wiki/ESP8266#%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)
		- [3.3.1Конфигурирование NAT](https://ru.wikipedia.org/wiki/ESP8266#%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_NAT)
		- [3.3.2DDNS](https://ru.wikipedia.org/wiki/ESP8266#DDNS)
		- [3.3.3Внешние IoT-сервисы](https://ru.wikipedia.org/wiki/ESP8266#%D0%92%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B5_IoT-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D1%8B)
* [4Internet of Things](https://ru.wikipedia.org/wiki/ESP8266#Internet_of_Things)
* [5Модули и отладочные платы](https://ru.wikipedia.org/wiki/ESP8266#%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B8_%D0%B8_%D0%BE%D1%82%D0%BB%D0%B0%D0%B4%D0%BE%D1%87%D0%BD%D1%8B%D0%B5_%D0%BF%D0%BB%D0%B0%D1%82%D1%8B)
* [6Аналогичные решения](https://ru.wikipedia.org/wiki/ESP8266#%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D1%8F)
* [7Забавные факты](https://ru.wikipedia.org/wiki/ESP8266#%D0%97%D0%B0%D0%B1%D0%B0%D0%B2%D0%BD%D1%8B%D0%B5_%D1%84%D0%B0%D0%BA%D1%82%D1%8B)
* [8См. также](https://ru.wikipedia.org/wiki/ESP8266#%D0%A1%D0%BC._%D1%82%D0%B0%D0%BA%D0%B6%D0%B5)
* [9Примечания](https://ru.wikipedia.org/wiki/ESP8266#%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D1%87%D0%B0%D0%BD%D0%B8%D1%8F)
* [10Литература](https://ru.wikipedia.org/wiki/ESP8266#%D0%9B%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%B0)
* [11Ссылки](https://ru.wikipedia.org/wiki/ESP8266#%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8)

**Микроконтроллер**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=1) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=1)]

* 80 MHz 32-bit процессор [Tensilica](https://ru.wikipedia.org/w/index.php?title=Tensilica&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/Tensilica) Xtensa L106. Возможен негарантированный разгон до 160 МГц.
* [IEEE 802.11](https://ru.wikipedia.org/wiki/IEEE_802.11) b/g/n [Wi-Fi](https://ru.wikipedia.org/wiki/Wi-Fi). Поддерживается [WEP](https://ru.wikipedia.org/wiki/Wired_Equivalent_Privacy) и WPA/WPA2.
* 14 портов ввода-вывода (из них возможно использовать 11), [SPI](https://ru.wikipedia.org/wiki/Serial_Peripheral_Interface), [I²S](https://ru.wikipedia.org/wiki/I%C2%B2S), [UART](https://ru.wikipedia.org/wiki/UART), 10-bit [АЦП](https://ru.wikipedia.org/wiki/%D0%90%D0%A6%D0%9F). [I²C](https://ru.wikipedia.org/wiki/I%C2%B2C) возможен только через [bit-banging](https://ru.wikipedia.org/wiki/Bit-banging).
* Питание 2,2…3,6 В. Потребление — до 215 мА в режиме передачи, 100 мА — в режиме приема, 70 мА — в режиме ожидания. Поддерживаются три режима пониженного потребления, все без сохранения соединения с точкой доступа: Modem sleep (15 мА), Light sleep (0.4 мА), Deep sleep (15 мкА)[[3]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-ESP8266_Low_Power_Solutions-3).

Микроконтроллер не имеет на кристалле пользовательской энергонезависимой памяти. **Исполнение программы** ведётся из внешней SPI ПЗУ путём динамической подгрузки требуемых участков программы в кеш инструкций. Подгрузка идёт аппаратно, прозрачно для программиста. Поддерживается до 16 МБ внешней памяти программ. Возможен Standard, Dual или Quad SPI интерфейс.

Производитель **не предоставляет документации** на внутреннюю периферию микроконтроллера. Вместо этого он даёт набор библиотек, через [API](https://ru.wikipedia.org/wiki/API) которых программист получает доступ к периферии. Поскольку эти библиотеки интенсивно используют ОЗУ контроллера, то производитель в документах не указывает точное количество ОЗУ на кристалле, а только приблизительную оценку того количества ОЗУ, что останется пользователю после сборки всех библиотек — порядка 50 кБ. По результатам исследования библиотеки ESP8266, предположительно он содержит 32 кБ кэша инструкций и 80 кБ ОЗУ данных.

Электрические параметры, цоколёвки, схемы включения можно найти в документах «0A-ESP8266EX\_\_Datasheet» и «0B-ESP8266\_\_System\_Description» из Espressif SDK[[4]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-SDK-4).

**Источник исполняемой программы** ESP8266 задаётся состоянием портов GPIO0, GPIO2 и GPIO15 в момент окончания сигнала Reset (то есть подачи питания). Наиболее интересны 2 режима:

* исполнение кода из UART (GPIO0 = 0, GPIO2 = 1 и GPIO15 = 0) - используется для перепрошивки подключённой флеш-памяти
* исполнение кода из внешней ПЗУ (GPIO0 = 1, GPIO2 = 1 и GPIO15 = 0) - используется как штатный режим работы

**ESP8285**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=2) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=2)]

Весной 2016 года Espressif Systems запустила массовое производство микросхемы ESP8285. Теперь в одной микросхеме находится как SoC ESP8266, так и 1 МБайт флеш-памяти[[5]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-5). Документацию на микросхему можно найти в документе «0A-ESP8285\_\_Datasheet».

**ESP32**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=3) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=3)]

*Основная статья*[*ESP32*](https://ru.wikipedia.org/wiki/ESP32)

Осенью 2015 года Espressif Systems представила развитие линейки — микросхему ESP32. В начале 2016 года инженерные образцы нового кристалла стали доступны партнерам компании для тестирования, в сентябре 2016 года ESP32 стал доступен как полноценный продукт[[6]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-6)[[7]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-7).

* Двухъядерный 32-бит Tensilica Xtensa® LX6 с FPU и MAC. 240 МГц (600 DMIPS).
* 448 кБайт ПЗУ, 520 кБайт ОЗУ. Внешние ОЗУ/ПЗУ на SPI интерфейсе, до 4\*16 МБайт. Внешняя память может быть криптографически защищена.
* Питание 2,2…3,6 В.
* Wifi 802.11, Bluetooth v4.2 (в том числе Low Energy).
* Увеличенное количество портов и периферии: ADC, DAC, 4 SPI, 2 I2S, 2 I2C, 3 UART, CAN. Интерфейс SD карт (как мастер так и слейв). Ethernet MAC.
* Корпус QFN-48.

**Средства разработки**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=4) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=4)]

[Программные средства разработки](https://ru.wikipedia.org/wiki/SDK) состоят из:

* [Компилятора](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Компилятор для Xtensa LX106 входит в пакет компиляторов [GNU Compiler Collection](https://ru.wikipedia.org/wiki/GNU_Compiler_Collection). Компилятор имеет открытые исходные тексты. В разных SDK могут содержаться разные сборки этого компилятора, немного отличающиеся поддерживаемыми опциями.
* Библиотек для работы с периферией контроллера, стеков протоколов WiFi, [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP).
* Средств загрузки исполняемого файла в память программ микроконтроллера.
* Опциональной [IDE](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8).

Espressif Systems свободно распространяет свой комплект разработчика. В этот комплект входит компилятор GCC, библиотеки Espressif и загрузочная утилита XTCOM. Библиотеки поставляются в виде скомпилированных библиотек, без исходных текстов. Espressif поддерживает две версии SDK: одна на основе [RTOS](https://ru.wikipedia.org/wiki/RTOS), другая на основе обратных вызовов (callback)[[4]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-SDK-4).

Помимо официальной SDK существует ряд проектов альтернативных SDK[[8]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-sdk3dparty-8). Эти SDK используют библиотеки Espressif Systems или предлагают собственный эквивалент библиотек Espressif Systems, полученный методами реверсинжиниринга.

* [«esp-open-sdk»](https://github.com/pfalcon/esp-open-sdk). Улучшенная версия SDK от Espressif. Содержит [GCC](https://ru.wikipedia.org/wiki/GCC) компилятор и некоторые библиотеки Espressif Systems. Только Линукс.
* [«Unofficial Development Kit»](http://programs74.ru/udkew.html) Михаила Григорьева. В комплект входит Windows-инсталлятор, компилятор GCC собственной сборки с интеграцией с графической IDE [Eclipse](https://ru.wikipedia.org/wiki/Eclipse_%28%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%29), актуальные комплекты библиотек и документации Espressif, некоторые утилиты. Имеется русскоязычный [форум](http://esp8266.ru/forum/threads/anonsy-unofficial-development-kit-for-espressif-esp8266.32/).
* [«Arduino IDE for ESP8266»](http://esp8266.ru/forum/threads/obschie-voprosy-po-arduino-ide-dlja-esp8266.224) — дополнение к IDE [Arduino](https://ru.wikipedia.org/wiki/Arduino), позволяющее программировать ESP8266 так же легко как любые другие модули Ардуино. При этом доступна сетевая функциональность ESP8266. Компилятор GCC, загрузчик прошивки ESPTool. Подробное русскоязычное описание процесса установки и доступного API [здесь](http://esp8266.ru/arduino-ide-esp8266/), пример работы [здесь](http://samopal.pro/esp8266-4/).
* [«GNU toolchain for esp8266»](http://gnutoolchains.com/esp8266/). Имеет возможность интеграции в [Visual Studio](https://ru.wikipedia.org/wiki/Visual_Studio).
* [«ESP8266 GCC Toolchain»](https://github.com/esp8266/esp8266-wiki/wiki/Toolchain) Макса Филиппова.
* «Sming»[[9]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-9) — проект добавления [Arduino](https://ru.wikipedia.org/wiki/Arduino) совместимых библиотек поверх стандартных библиотек Espressif, но без среды Ардуино.

**Прошивки**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=5) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=5)]

Чтобы упростить использование микроконтроллера в типовых проектах возможно использование готовых бинарных файлов, пригодных к прямой заливке в ПЗУ модулей (так называемых [прошивок](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%BE%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)). Готовые прошивки можно разделить на несколько групп согласно концепции их использования:

* Прошивки для работы **под управлением внешнего контроллера**. В этих прошивках реализовано задание параметров работы через внешний контроллер [UART](https://ru.wikipedia.org/wiki/UART). К таким прошивкам относится:
	+ Прошивка с управлением [AT-командами](https://ru.wikipedia.org/wiki/AT-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B) из SDK Espressif[[4]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-SDK-4). Там же можно найти документы «4A-ESP8266\_\_AT Instruction Set» и «4B-ESP8266\_\_AT Command Examples». Русскоязычный справочник по AT командам [здесь](http://esp8266.ru/esp8266-at-commands-v022/). Также существует [утилита](http://esp8266.ru/esp8266-appstack-esp8266-config/), позволяющая сконфигурировать модуль без знания AT команд.
* Прошивки с **встроенными**[**интерпретаторами**](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80) разнообразных языков высокого уровня. Эти прошивки позволяют подгружать через UART и исполнять скрипты разработчика устройства.
	+ [NodeMCU](https://ru.wikipedia.org/w/index.php?title=NodeMCU&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/NodeMCU) — проект на основе скриптового языка [Lua](https://ru.wikipedia.org/wiki/Lua)[[10]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-10). Прошивка умеет исполнять Lua-скрипты как из UART (аналогично AT-командам) так и из внутренней flash памяти. Для загрузки скриптов в flash память поддерживается файловая система. Со стороны Wi-Fi имеются встроенные MQTT протокол и HTTP сервер. Встроена графическая оболочка, что позволяет подключать к ESP8266 графические индикаторы. Краткий русскоязычный обзор можно найти [здесь](http://esp8266.ru/esp8266-nodemcu/) и [здесь](https://web.archive.org/web/20160223060452/http%3A/smart-el.ru/?p=149). Русскоязычное описание API языка [здесь](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_ru). Официальная документация [здесь](http://nodemcu.readthedocs.org/en/dev/). Есть живые [русскоязычный](http://esp8266.ru/forum/forums/nodemcu/) и [англоязычный](http://www.esp8266.com/viewforum.php?f=17) форумы. В интернет-магазинах популярны недорогие модули с прошивкой NodeMCU.
	+ [Espruino](http://www.espruino.com/) — проект на основе скриптового языка [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). Имеется графическая оболочка (IDE) для работы с исходными текстами скриптов и закачки их в модули. Исходники, прошивки, документы и инструментарий [здесь](https://github.com/espruino). Русскоязычный обзор [здесь](http://tim4dev.com/2015/01/espruino-javascript-dlya-veshhej/). Англоязычный форум [здесь](http://www.esp8266.com/viewforum.php?f=44).
	+ [ESP8266 BASIC](http://www.esp8266basic.com/) — проект на основе скриптового языка [Basic](https://ru.wikipedia.org/wiki/Basic). Проект отличает возможность редактирования и запуска скриптов через браузер, через HTML страничку ESP8266. Также проект поддерживает свою графическую утилиту для прошивки флеш-памяти. Поддерживаются модули с 512 кБ, 1, 2 или 4 МБ памяти.
	+ [Smart.js](https://github.com/cesanta/smart.js) — проект на основе скриптового языка [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). Русскоязычный обзор [здесь](https://geektimes.ru/post/259260/).
	+ [NodeLUA](https://web.archive.org/web/20160503054627/https%3A/nodelua.org/) — проект на основе скриптового языка [Lua](https://ru.wikipedia.org/wiki/Lua).
	+ [ZBasic](http://esp8266.ru/forum/threads/zbasic-dlja-esp8266.595/) — проект на основе скриптового языка [Basic](https://ru.wikipedia.org/wiki/Basic).
	+ [esp-lisp](https://github.com/yesco/esp-lisp) — вялотекущий проект на основе скриптового языка [Лисп](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%81%D0%BF).
	+ [MicroPython](https://github.com/micropython/micropython/tree/master/ports/esp8266) — проект на основе скриптового языка [MicroPython](https://ru.wikipedia.org/wiki/MicroPython).
* Прошивки для **интернета вещей**. Этот класс прошивок позволяет с одной стороны подключить к ESP8266 набор датчиков и исполнительных устройств, а с другой стороны предоставляет необходимую сетевую функциональность для работы в инфраструктуре Internet of Things.
	+ [ESP Easy](https://ru.wikipedia.org/w/index.php?title=ESP_Easy&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/ESP_Easy)[[11]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-11) — прошивка для домашней автоматизации. Выполнена как скетч [Arduino](https://ru.wikipedia.org/wiki/Arduino). Поддерживает множество датчиков и исполнительных устройств, поддержка которых есть в Arduino (например, датчики температуры, влажности, освещённости, дисплеи, реле и т. п.). Есть клиент MQTT. Документация [здесь](http://www.esp8266.nu/index.php/Main_Page).
	+ [WiFi-IoT.ru](http://wifi-iot.ru/) — онлайн конструктор IoT-прошивок для ESP8266 от Максима Миклина для проекта домашней автоматизации [homes-smart.ru](http://homes-smart.ru/). Позволяет прямо на сайте создать прошивку ESP8266 для работы с одной из множества поддерживаемых IoT систем верхнего уровня. Вариант требуемой прошивки создаётся путём выбора опций конфигурации. Проект коммерческий, но есть бесплатные ограниченные возможности.
	+ [BOLT IOT Platform](https://ru.wikipedia.org/w/index.php?title=BOLT_IOT_Platform&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/BOLT_IOT_Platform)[[12]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-12) — полностью коммерческий индийский проект, доступный пока только на рынке Индии. Интересен тем что программируется в стиле HTML кода, который разработчик может набрать прямо на HTML страничке своего устройства. Есть приложения для Android и iOS.
* Прошивки для популярного применения **переходника UART-WiFi**.
	+ Проект библиотек HTTP сервера с файловой системой для ESP8266[[13]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-13). Реализована настройка WiFi параметров через HTML интерфейс. На основе библиотеки создан проект TCP2UART переходника[[14]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-14).
	+ [ESP8266-transparent-bridge](https://github.com/beckdac/ESP8266-transparent-bridge) — проект прозрачного переходника TCP-UART.

**Начальная загрузка и обновление прошивки**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=6) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=6)]

Источник исполняемой программы ESP8266 задаётся состоянием портов GPIO0, GPIO2 и GPIO15 в момент окончания сигнала Reset (то есть подачи питания). Наиболее интересны два режима: исполнение кода из UART (GPIO0 = 0, GPIO2 = 1, GPIO15 = 0) и из внешней ПЗУ (GPIO0 = 1, GPIO2 = 1, GPIO15 = 0). Режим исполнения кода из UART используется для перепрошивки флеш-памяти, а второй режим штатный рабочий.

Для управления процессом обновления прошивки имеется множество утилит:

* XTCOM — консольная утилита из Espressif SDK[[4]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-SDK-4). Подробное русскоязычное описание утилиты XTCOM [здесь](http://esp8266.ru/esp8266-obnovlenie-proshivki-xtcom_util/).
* [NodeMCU-Flasher](https://github.com/nodemcu/nodemcu-flasher) — оконная утилита под Win.
* [esptool\_ck](https://github.com/igrr/esptool-ck) — консольная утилита, написанная на Си. Есть сборки Win и Linux. Поддерживает также ESP32.
* [esp\_tool](https://github.com/DonKinzer/esp_tool/) — консольная утилита, написанная на C++. Есть сборки Win и Linux.
* [esptool.py](https://github.com/themadinventor/esptool) — консольная утилита, написанная на [Python](https://ru.wikipedia.org/wiki/Python).

Внешняя SPI flash должна иметь определённый заголовок для корректного исполнения кода. Структура заголовка указана в документации к утилите XTCOM. На русском есть [здесь](http://www.pvsm.ru/diy/88150). Загрузочные утилиты обычно умеют добавлять его к прошивке.

**Обновление через Wi-Fi**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=7) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=7)]

Предусмотрена возможность обновить прошивку работающего устройства через Wi-Fi. Для этого разделяют флеш-память программ на несколько частей. Одна отводится менеджеру прошивок, две другие под пользовательскую программу. Когда хотят обновить прошивку, новый образ загружают в свободную часть флеш-памяти. После тщательной проверки целостности вновь загруженного образа менеджер прошивок переключает флажок, после чего участок памяти со старой прошивкой освобождается, а исполнение кода идёт из нового участка. Соответственно в следующий раз обновление будет загружаться в свободный участок памяти. Подробности в документе «99C-ESP8266\_\_OTA\_Upgrade» из Espressif SDK[[4]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-SDK-4).

**Утилиты**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=8) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=8)]

* [ESPlorer](http://esp8266.ru/esplorer-ide-esp8266/) — IDE для ESP8266. Содержит редактор и средства связи с модулем. Позволяет подгружать скрипты в проекте NodeMCU.

**Сетевая инфраструктура**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=9) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=9)]

Типовое применение ESP8266 как аппаратной основы Internet of Things чаще всего подразумевает установку в домах или офисах. При этом сетевое подключение осуществляется к домашней/офисной локальной сети с выходом в интернет через [роутер](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80). Пользователь устройства может контролировать его с помощью планшета или компьютера через свою локальную сеть либо удалённо, через Интернет.

**Wi-Fi**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=10) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=10)]

ESP8266 может работать как в роли [точки доступа](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%87%D0%BA%D0%B0_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0), так и оконечной станции. При нормальной работе в локальной сети ESP8266 конфигурируется в режим оконечной станции. Для этого устройству необходимо задать [SSID](https://ru.wikipedia.org/wiki/SSID) Wi-Fi сети и, в закрытых сетях, пароль доступа. Для **первоначального конфигурирования** этих параметров удобен режим точки доступа. В режиме точки доступа устройство видно при стандартном поиске сетей в планшетах и компьютерах. Остаётся подключиться к устройству, открыть HTML-страничку конфигурирования и задать сетевые параметры, после чего устройство штатно подключится к локальной сети в режиме оконечной станции.

В случае исключительно местного использования возможно всегда оставлять устройство в режиме точки доступа, что снижает необходимые усилия пользователя по его настройке.

**Локальная сеть**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=11) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=11)]

После подключения к Wi-Fi сети устройство должно получить IP-параметры локальной сети. Эти параметры можно задать вручную вместе с параметрами Wi-Fi либо активизировать какие-либо [сервисы автоматического конфигурирования IP-параметров](https://ru.wikipedia.org/wiki/Zeroconf) (например, [DHCP](https://ru.wikipedia.org/wiki/DHCP)).

После настройки IP параметров обращение к серверу устройства в локальной сети обычно осуществляется по его IP-адресу, сетевому имени (в случае, если имена поддержаны какой-либо технологией, например, [NBNS](https://ru.wikipedia.org/wiki/NBNS)) или сервису (в случае, если поддержан автоматический [поиск сервисов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BD%D0%B0%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D0%BE%D0%B2), например, через протокол [SSDP](https://ru.wikipedia.org/wiki/Simple_Service_Discovery_Protocol)).

**Интернет**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=12) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=12)]

Зачастую доступ к устройству требуется из Интернета. Например пользователь с мобильного телефона удалённо проверяет состояние своего «[умного дома](https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%BC)», обращаясь напрямую к устройству. В этом случае устройство работает в режиме сервера, к которому обращается внешний клиент.

Как правило, устройство на основе ESP8266 находится в локальной сети офиса или дома. Выход в Интернет обеспечивает [роутер](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80), подключённый с одной стороны к локальной сети, а с другой - к сети провайдера интернета. Провайдер назначает роутеру свой статический или динамический IP-адрес и роутер осуществляет [трансляцию адресов](https://ru.wikipedia.org/wiki/NAT) локальной сети в сеть провайдера. По умолчанию правила этой трансляции обеспечивают свободную видимость интернет-адресов из локальной сети, но не позволяют обратиться к локальным адресам со стороны Интернета. Есть несколько способов обойти это ограничение.

**Конфигурирование NAT**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=13) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=13)]

Большинство современных роутеров позволяет задать дополнительные правила [трансляции сетевых адресов](https://ru.wikipedia.org/wiki/NAT) между локальной и глобальной сетями. Как правило, для этого используются технологии [Virtual server](https://ru.wikipedia.org/w/index.php?title=Virtual_server&action=edit&redlink=1) или [DMZ](https://ru.wikipedia.org/wiki/DMZ_%28%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8%29). Обе технологии позволяют обратиться к серверу в локальной сети из глобальной сети, зная лишь IP-адрес, выданный роутеру провайдером. В случае статического IP-адреса роутера это зачастую может быть удовлетворительным решением для ограниченного круга пользователей системы. Однако такой подход не всегда удобен: необходимо вручную конфигурировать роутер и выяснять IP-адрес роутера, который может регулярно меняться. Относительно легко решить проблему неизвестного IP-адреса можно с помощью механизма [DDNS](https://ru.wikipedia.org/wiki/DDNS).

**DDNS**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=14) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=14)]

Чтобы обратиться к серверу устройства, конечный пользователь должен знать IP-адрес, по которому находится устройство. Однако получить у провайдера Интернета для устройства статический IP-адрес не всегда возможно, да и пользоваться таким адресом неудобно. Для решения этой проблемы были созданы специальные интернет-сервисы под общим наименованием [**динамический DNS**](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_DNS). Эти сервисы работают как специальные серверы с фиксированными именами в интернете. Разработчик заводит на таком сервисе свой аккаунт с уникальным именем. Параметры этого аккаунта он прописывает в устройстве. Устройство в режиме клиента периодически обращается к серверу сервиса, сообщая ему имя своего аккаунта и свой текущий IP-адрес. Конечный пользователь в интернете обращается к этому же сервису и получает от него текущие IP-параметры устройства. В таком случае устройство в сети видно с доменным именем третьего уровня, например esp8266.ddns.org.

Основная проблема DDNS-сервисов - это гарантии существования конкретного сервиса. Как правило, гарантируется только коммерческий сервис, когда за его использование взимается плата.

**Внешние IoT-сервисы**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=15) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=15)]

Чтобы облегчить проблему доступности устройства в Интернете и сделать инсталляцию устройства лёгкой для пользователя, был разработан ряд решений. Механизм этих решений базируется на существовании в Интернете специального сервера, к которому может подключиться как IoT-устройство, так и планшет/компьютер пользователя. При этом устройство работает в режиме клиента, никаких специальных настроек роутера или особых навыков от инсталлятора и пользователя устройства не требуется. Обмен данными с устройством осуществляется при посредничестве этого специального сервиса, параметры которого в устройство должен заложить разработчик. Распространение использования таких сервисов сдерживается необходимостью длительно поддерживать свой сервис в Интернете или пользоваться чужими сервисами с непонятными перспективами длительного существования бесплатных возможностей или регулярной оплатой коммерческих вариантов.

**Internet of Things**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=16) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=16)]

Основное применение ESP8266 находит в управлении разнообразными бытовыми приборами через беспроводные сети. Концепцию такого управления часто называют «[Internet of Things](https://ru.wikipedia.org/wiki/Internet_of_Things)» (IoT, «интернет вещей»). Верхний уровень IoT представлен разнообразными приложениями под популярные платформы (Android, iOS, Windows, …). Эти приложения позволяют разработчику прибора адаптировать приложение под управление его прибором и передать пользователю готовое решение. Существует несколько популярных реализаций концепции IoT в плане обмена данными по сети:

* HTTP сервер на ESP8266. Контроль и управление устройством ведется через браузер. Тяжеловесное решение, подходит автономным устройствам автоматики.
* [AllJoyn](https://ru.wikipedia.org/w/index.php?title=AllJoyn&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/AllJoyn)[[15]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-15) — набирающий популярность открытый IoT протокол крупного альянса производителей цифровой техники «Allseen». Поддержка встроена в [Windows 10](https://ru.wikipedia.org/wiki/Windows_10). На русском можно почитать [здесь](https://habrahabr.ru/company/rainbow/blog/273859/).
* HTTP запросы с использованием протоколов типа [REST](https://ru.wikipedia.org/wiki/REST), [XML-RPC](https://ru.wikipedia.org/wiki/XML-RPC) ([SOAP](https://ru.wikipedia.org/wiki/SOAP)). Для этого на ESP8266 запускают упрощённый HTTP сервер, без HTML. Достоинство метода — отсутствие проблем с настройкой файрволов, HTTP обычно открыт всегда.
* [MQTT](https://ru.wikipedia.org/wiki/MQTT). Это простой протокол поверх TCP/IP. Очень популярное решение. Существует большое количество IoT приложений верхнего уровня для Android, iOS и других платформ, поддерживающих этот протокол.
* [SNMP](https://ru.wikipedia.org/wiki/SNMP). Расширяемый протокол управления сетевыми устройствами. Основной недостаток в том что в большинстве сетей файрволы блокируют прохождение SNMP.
* [ModBus](https://ru.wikipedia.org/wiki/ModBus) и другие протоколы промышленной автоматизации.

Интересные проекты ПО верхнего уровня с решениями на базе ESP8266:

* Majordomo — русскоязычный открытый проект домашней автоматизации[[16]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-16).
* Blynk — облачная платформа для IoT, которая имеет приложения для iOS и Android и поддерживает управление микроконтроллерами ESP8266, [Arduino](https://ru.wikipedia.org/wiki/Arduino), [Raspberry Pi](https://ru.wikipedia.org/wiki/Raspberry_Pi), SparkFun и д.р. через Интернет[[17]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-17)[[18]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-18).
* SUPLA — открытый проект систем автоматизации зданий, использующий ESP8266[[19]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-19).
* BortX — IOT платформа с открытым доступом для ESP8266[[20]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-20).

**Модули и отладочные платы**

[[править](https://ru.wikipedia.org/w/index.php?title=ESP8266&veaction=edit&section=17) | [править код](https://ru.wikipedia.org/w/index.php?title=ESP8266&action=edit&section=17)]

Первыми и наиболее популярными встраиваемыми модулями[[21]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-21) на базе ESP8266 были изделия китайской компании AI-Thinker[[22]](https://ru.wikipedia.org/wiki/ESP8266#cite_note-22). Как правило, эти модули поступают в продажу с прошивками, поддерживающими [AT-команды](https://ru.wikipedia.org/wiki/AT-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B). Однако у компании есть своя прошивка под IoT приложение, некоторые модули могут поставляться с ней. К сожалению, компания поддерживает только китайский язык, что затрудняет использование ее IoT прошивок и Андроид-приложений для любительской автоматизации.

Первоначально модули поставлялись с Flash памятью объёмом 512 кБ. Позже официальные прошивки подросли и перестали помещаться в полмегабайта. Поэтому сегодня большинство модулей поставляется с Flash памятью объёмом 4 МБ.