ИНТЕРНЕТ ВЕЩЕЙ

ПРОДОЛЖЕНИЕ «КОНСПЕКТА ХАКЕРА»

Амперка

Электронная версия книги:
iot-m.amperka.ru

СОДЕРЖАНИЕ

Включать приборы дистанционно

Управлять приборами со смартфона

Управлять светом

IOT, «INTERNET OF THINGS» — ИНТЕРНЕТ ВЕЩЕЙ

Это громкое словосочетание означает концепцию связи большого количества устройств (вещей) в общую сеть. Устройства общаются между собой через Интернет: передают друг другу информацию, а затем обрабатывают её. Не просто так, а принося пользу людям. Например, собирают данные о погоде со всех концов Земли, управляют офисными зданиями или сообщают пути объезда, если впереди на дороге образовалась пробка.

Рой устройств в единой сети создаёт полную картину происходящего вокруг, повышает комфорт и позволяет улучшить качество жизни людей.

Собирать данные о здоровье

Включать музыку по расписанию

Наблюдать за приборами по графикам

Создавать системы оповещений

Активировать видеослежение по движению

Просыпаться с умным будильником

«Умный дом» — понятие, тесно связанное с интернетом вещей. Множество устройств, включённых в общую сеть, автоматизируют домашние рутинные задачи и позволяют удалённо наблюдать за состоянием дома. Например, можно включить чайник незадолго до прихода домой. Да и управлять светом в квартире со смартфона — это очень весело!

Эксперименты из этого набора помогут тебе сделать первые шаги к созданию своего умного дома. Подумай о том, какие процессы ты хотел бы автоматизировать, вооружись знаниями и действуй!

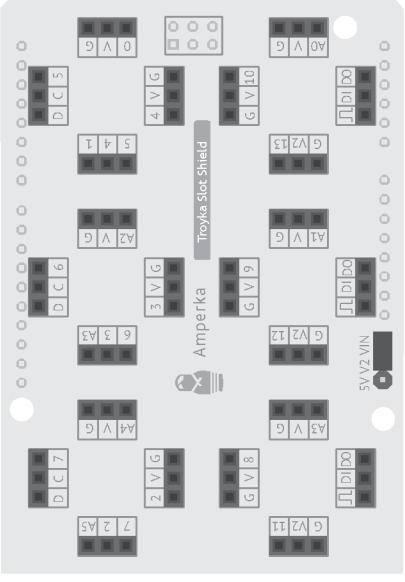
Собирать данные о погоде

Следить за объектами

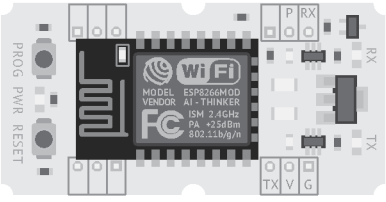
Измерять температуру

Собирать данные в единую базу

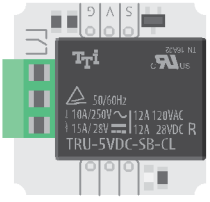
ЭЛЕМЕНТЫ В НАБОРЕ



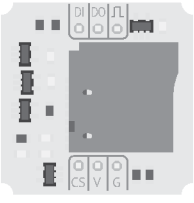
Troyka Slot Shield
Плата расширения для компактного подключения модулей



Wi-Fi модуль
Позволяет передавать данные по Wi-Fi-сети



Мини-реле
Управляет высоковольтными приборами

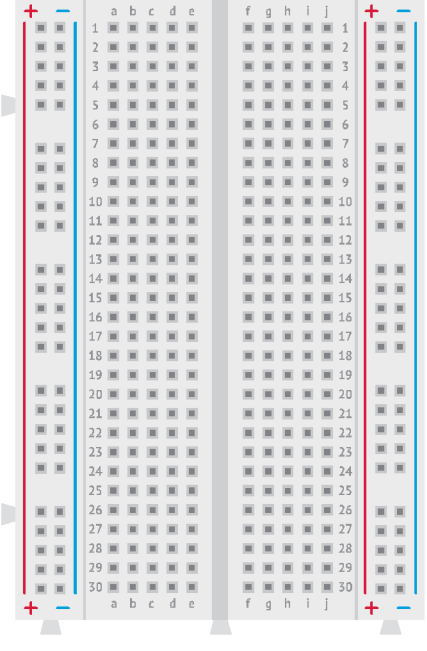


SD-картридер
Читает и записывает файлы на карту microSD

Карта microSD
Необходима для хранения файлов

ЕЩЁ ПОНАДОБЯТСЯ

Для проектов IoT тебе потребуются платы и компоненты из набора «Матрёшка Y» или «Матрёшка Z».

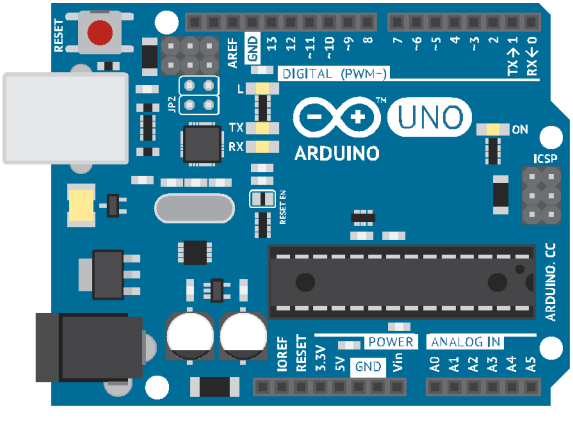


Макетная плата Breadboard Half

Резисторы
на 10 кОм
и на 220 Ом

Тактовая
кнопка

Кабель USB B



Arduino Uno

Фоторезистор

Термистор

Светодиод

Трёхцветный
(RGB) светодиод

ИНТЕРФЕЙСЫ И ПРОТОКОЛЫ

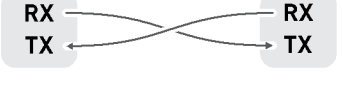
ИНТЕРФЕЙСЫ

Интерфейсы устанавливают физический способ передачи сигналов от устройства к устройству.

У людей тоже есть свои интерфейсы. Они передают друг другу сообщения устно или письменно. Электроника вместо голоса и письма использует электрические провода и радиоволны. Очень важно передавать данные на одной скорости, иначе устройства не поймут сообщения друг друга.

ИНТЕРФЕЙС UART (SERIAL)

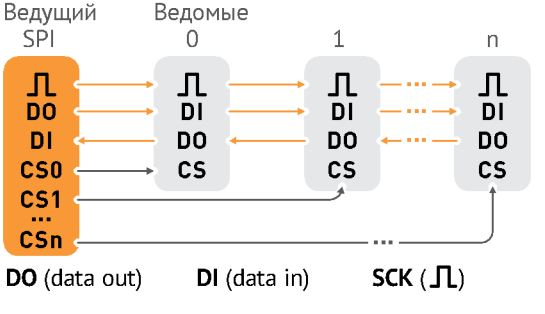
UART (Universal Asynchronous Receiver-Transmitter, *Универсальный асинхронный приёмопередачик*) использует для передачи данных два провода: по одному передаёт данные, по другому — принимает. UART задаёт скорость передачи в бодах (1 бод = 1 бит в секунду). Она принимает конкретные значения, например 9 600 бод, 115 200 бод и другие. Оба устройства обязаны передавать данные на одной и той же скорости. Часто этот интерфейс ещё называют Serial (*последовательный*). Он передаёт данные последовательно, бит за битом.



Пин RX принимает данные, а пин TX — передаёт. Скорость в бодах иногда называют *битрейтом*.

ИНТЕРФЕЙС SPI

Интерфейс SPI (Serial Peripheral Interface), в отличие от UART, позволяет соединить в сеть больше двух устройств. Одно из них становится ведущим (Master, *мастером*), остальные — ведомыми (Slave). Ведущее устройство по очереди передаёт данные ведомым по линии MOSI (master output slave input). Очерёдность задаётся линиями CS (Chip Select, *выбор ведомого*). Ведомые устройства передают свои данные по линии MISO (master input slave output), но только с разрешения мастера (линией CS). Скорость передачи задаётся линией SCK.



ПРОТОКОЛЫ

Протоколы устанавливают правила передачи данных между устройствами. Они используют интерфейсы как «транспорт» для данных.

Языки, на которых общаются люди, тоже можно назвать протоколами. «Меня зовут Амперка» и «My name is Amperka» — одна и та же информация, но передана она разными протоколами. Люди могут общаться, только если знают общий протокол и умеют его использовать. Устройствам для общения тоже нужно знать общий протокол.

ПРОТОКОЛЫ TCP/IP

TCP/IP — это набор протоколов (названный по двум важнейшим — TCP и IP). Этот набор обеспечивает работу сети Интернет.

TCP (сокращение от английского Transmission Control Protocol, *протокол управления передачей*) — протокол, обеспечивающий передачу данных. Задачей этого протокола является прямая связь между конечными пунктами — клиентом и сервером. Ещё одна функция этого протокола — проверка данных на целостность.

IP (Internet Protocol) обеспечивает доставку данных по определённым адресам — он определяет, куда именно нужно направить данные.

ПРОТОКОЛ HTTP

Протокол HTTP обеспечивает передачу html-страниц и медиафайлов. С ним работают все браузеры.

Передача данных выглядит как простой текст из множества строк.

СИСТЕМА DNS

Систему DNS (англ. Domain Name System — система доменных имён) можно представить себе как адресную книжку вида «IP-адрес — буквенный адрес». Это позволяет не запоминать длинные и непонятные числа, чтобы попасть на нужный сайт. Пример: запомнить адрес сайта «amperka.ru» гораздо проще, чем его IP-адрес «178.79.159.36».

Буквенные адреса называются URL (англ. Uniform Resource Locator, Единый указатель ресурса). URL состоит из двух частей — «имени» сайта (до точки) и доменной зоны (после точки). Доменные зоны объединяют сайты по их языку, назначению и другим признакам.

ПРОТОКОЛ HTTPS

Протокол HTTPS — это тот же HTTP, но с шифрованием (S — security). Оно защищает данные от перехвата злоумышленниками. Шифрование требует сравнительно большого количества вычислительных ресурсов.

Это далеко не все существующие интерфейсы и протоколы, но в рамках экспериментов ты будешь иметь дело только с этими.

СТРУКТУРА ЛОКАЛЬНОЙ И ГЛОБАЛЬНОЙ СЕТЕЙ

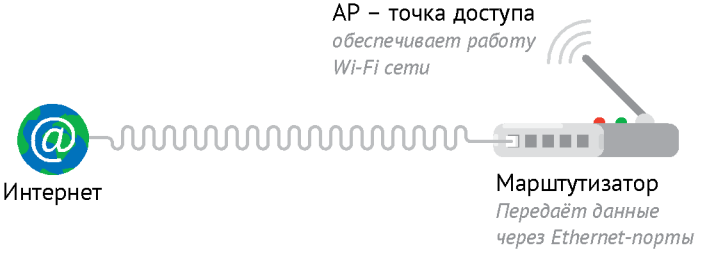
Технически интернет — это сеть из устройств, общающихся между собой с помощью проводов или радиоволн. Устройства запрашивают друг у друга данные и отдают их. Того, кто запрашивает, называют *клиентом*. Того, кто отдаёт, — *сервером*.

Устройства из этого набора выходят в интернет при помощи Wi-Fi. Сеть Wi-Fi создаётся роутером, к нему подключаются клиенты.

Роутер позволяет им выйти в интернет, только если сам имеет к нему доступ.

МАРШРУТИЗАТОР И ТОЧКА ДОСТУПА

Wi-Fi роутер состоит из двух устройств. Первое из них — маршрутизатор, он обеспечивает передачу информации между разными сетями по протоколу TCP/IP. Второе — беспроводная точка доступа (сокращённо AP — от англ. *Access Point*). Она создаёт и обеспечивает работу Wi-Fi сети.

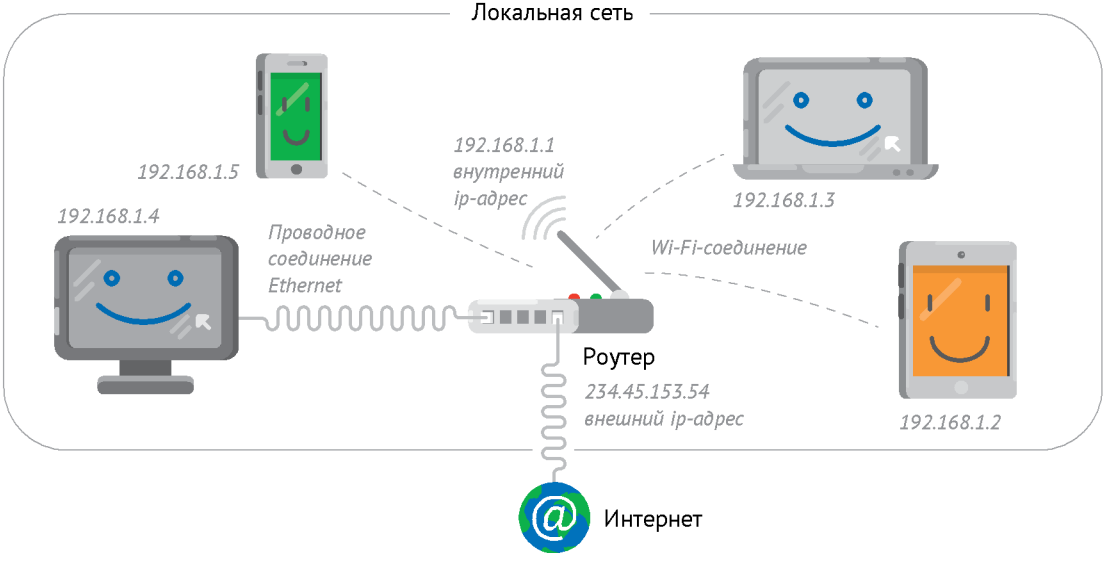


Для подключения к роутеру по Wi-Fi необходимо знать SSID (имя точки доступа) и пароль сети.

ЛОКАЛЬНАЯ СЕТЬ

Клиентов, подключённых к сети через один маршрутизатор, называют объединёнными в локальную сеть.

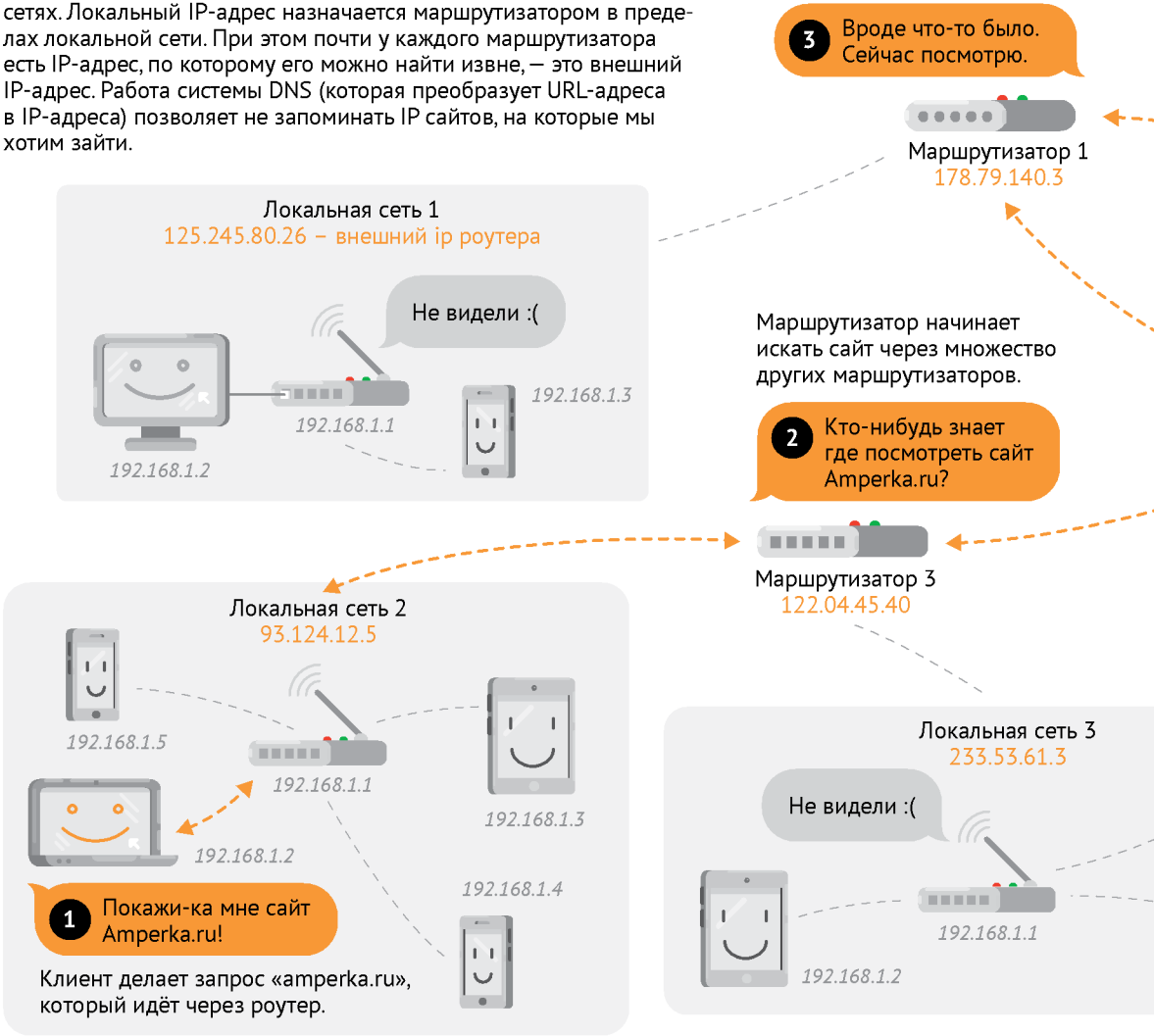
Каждое устройство в сети имеет IP-адрес. По этому адресу маршрутизатор понимает к какому устройству ему следует обратиться. Локальные IP-адреса имеют формат 192.168.xxx.xxx.

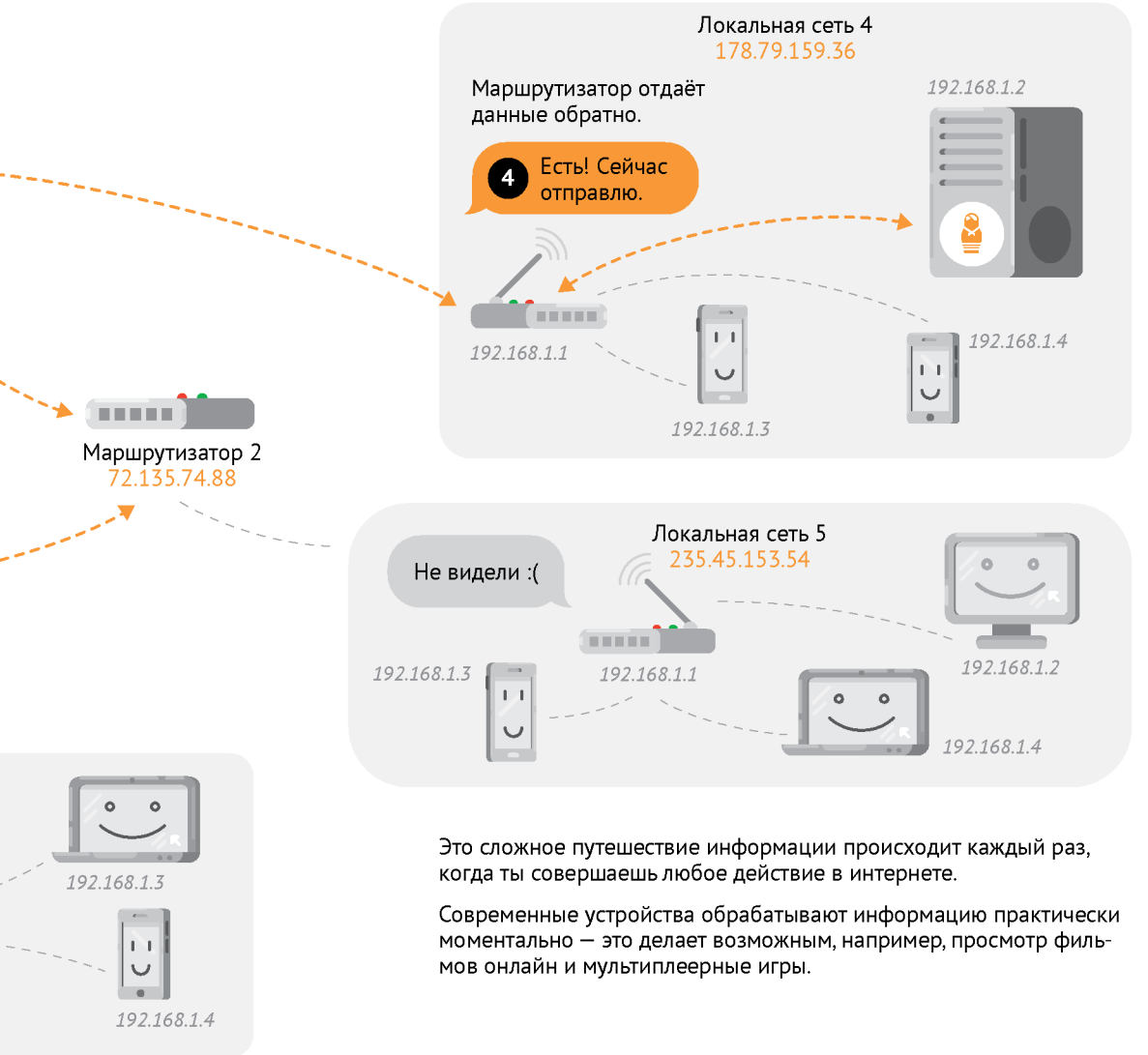


Локальные сети тоже объединены между собой маршрутизаторами. Пример: локальные сети каждой квартиры в подъезде объединены маршрутизатором, расположенным на последнем этаже дома. Масса локальных сетей и создаёт глобальную сеть, а проще — интернет.

IP-АДРЕСА

Существует разница между IP-адресами в локальной и глобальной сетях. Локальный IP-адрес назначается маршрутизатором в пределах локальной сети. При этом почти у каждого маршрутизатора есть IP-адрес, по которому его можно найти извне, — это внешний IP-адрес. Работа системы DNS (которая преобразует URL-адреса в IP-адреса) позволяет не запоминать IP сайтов, на которые мы хотим зайти.





Это сложное путешествие информации происходит каждый раз, когда ты совершаешь любое действие в интернете.

Современные устройства обрабатывают информацию практически моментально — это делает возможным, например, просмотр фильмов онлайн и мультиплеерные игры.

ПРОТОКОЛ HTTP

HTTP — это повсеместно используемый протокол для передачи веб-страниц и медиафайлов. Каждое твоё действие с любым сайтом в браузере происходит с помощью этого протокола.

По протоколу HTTP устройства общаются прерывно — запрос-ответ. Клиент посылает запрос. Протокол IP направляет его к нужному серверу через кучу маршрутизаторов. Открывается TCP-соединение между клиентом и сервером, и по нему, байт за байтом, запрос приходит по нужному адресу. После обработки запроса сервер отправляет ответ обратно клиенту, и TCP-соединение обрывается.

Данные по протоколу HTTP передаются в виде текстовых строк. В начале текста записаны *заголовки* (headers), затем идёт *тело* (body). Заголовки — это особые строки, которые определяют свойства тела. Например, они сообщают размер тела в байтах, тип данных (текст, объект, файл) и служебную информацию.

Наиболее распространены два метода запроса: GET и POST. Они отличаются по структуре.

GET-ЗАПРОС

GET-запрос устроен просто. Его можно послать прямо из адресной строки браузера. В GET-запросе существует только заголовок. Тела запроса в нём нет. Например, GET-запрос браузера к сайту Амперки выглядит примерно так:

1 `GET` — метод запроса,
`/product/matryoshka-z` — запрашиваемая страница,
`HTTP/1.1` — версия протокола.

```
GET /product/matryoshka-z HTTP/1.1;
Host: amperka.ru;
Upgrade-Insecure-Requests: 1;
User-Agent: Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
Accept-Encoding: gzip, deflate;
Accept-Language: en-GB,en;q=0.8,en-US;q=0.6,ru;q=0.4;
```

2 Заголовок `Host` означает имя сайта, на который отправляется запрос.

3 Запрос к серверу на установку безопасного соединения (HTTPS).

4 Информация о браузере клиента.

5 Типы контента, понятные клиенту.

6 Способы кодировки, которые клиент готов обработать в ответе.

7 Языки, с которыми может работать клиент.

Ответ сервера на запрос:

```
HTTP/1.1 200 OK
Date: Mon, 10 Apr 2017 11:43:05 GMT
Content-Language: ru
Content-Type: text/html
Content-Length: 1234

далее следует тело ответа
```

1 Первая строчка в ответе сервера означает его статус. Каждый статус закодирован числом. Статус 200 говорит, что всё в порядке, сервер обработал запрос и отдал корректный ответ.

2 Поле `Date` передаёт время сервера. Оно может отличаться от твоего местного времени, если сервер находится в другом часовом поясе.

3 Поля, начинающиеся со слова `Content`, сообщают свойства тела ответа. Например, длину в байтах, кодировку, формат данных и некоторые другие.

4 Пустая строка в ответе обязательна. Она сообщает, что заголовки закончились и дальше пойдёт тело ответа.

5 В теле ответа содержатся запрашиваемые данные, например HTML-страница сайта amperka.ru.

В GET-запросе также можно передавать дополнительную информацию прямо из строки браузера. Выглядит это так:

amperka.ru?login=Amperka&password=12345

Прямо при обращении к серверу передаём ему информацию о логине и пароле.

Сначала мы пишем имя сайта, к которому обращаемся. После URL находится служебный символ «?». Он сообщает серверу, что мы хотим передать в запросе дополнительные параметры.

Дополнительных параметров может быть больше одного. Они разделяются между собой служебным символом «&».

Именно из-за их использования в структуре запроса в URL-адресах нельзя использовать символы «?» и «&».

Дополнительные параметры передаются в первой строке запроса.

```
GET /?login=Amperka&password=12345 HTTP/1.1
...
```

POST-ЗАПРОС

POST-запрос также позволяет сообщать или запрашивать дополнительную информацию у сервера, но устроен он гораздо сложнее. В первую очередь работа с этим запросом происходит не через адресную строку, а при нажатии кнопки на сайте (например, при заполнении полей логина и пароля). При работе POST параметры передаются в фоновом режиме.

Существует специальный параметр `Content-Type`, определяющий вид информации, который необходимо передать. Например, POST можно использовать для загрузки файлов большого объёма.

Вот пример запроса, выполненного методом POST. В нём уже есть не только заголовок, но и тело запроса.

```
POST / HTTP/1.0
Host: www.amperka.ru
Referer: http://www.amperka.ru/index.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 28

login=Amperka&password=12345
```

1 `POST` — метод запроса.

2 Ссылка на страницу, с которой работает пользователь.

3 Тип содержимого. Например, форма регистрации на сайте.

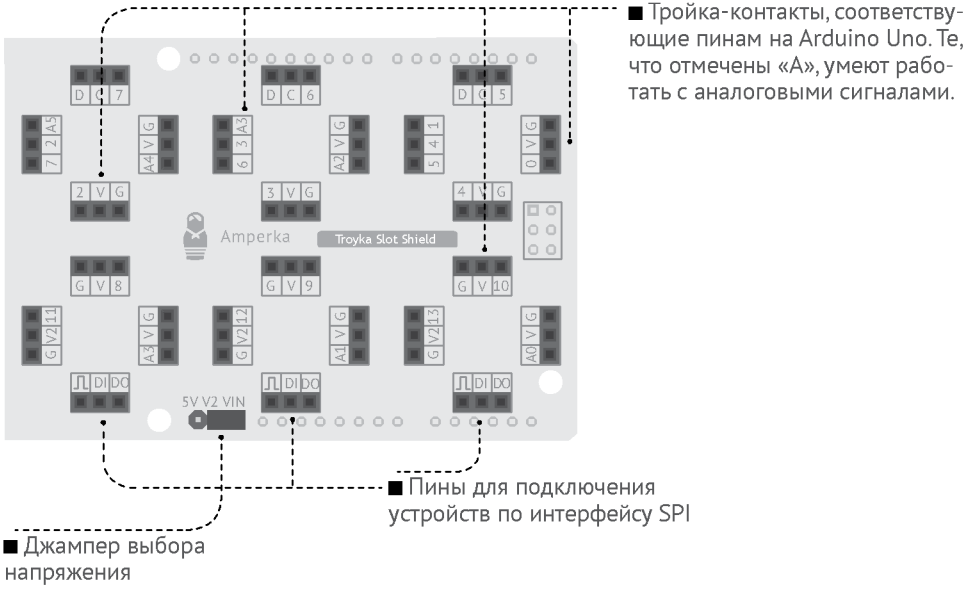
4 Длина тела запроса в байтах.

5 Пустая строка.

6 Тело запроса.

TROYKA SLOT SHIELD

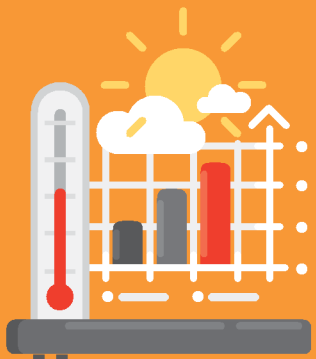
Troyka Slot Shield — это плата расширения для быстрой сборки компактных устройств из Troyka-модулей без проводов и паяльника. На плате расположены шесть слотов. Каждый слот состоит из четырёх групп контактов, в которые можно подключить один Troyka-модуль.

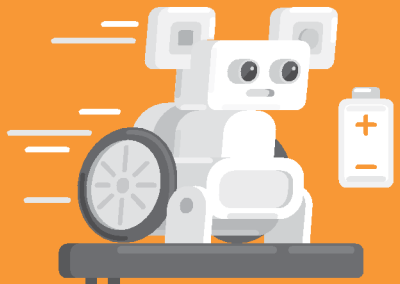


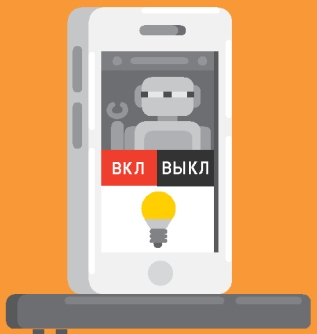
В ПЛАТУ МОЖНО ПОДКЛЮЧИТЬ:

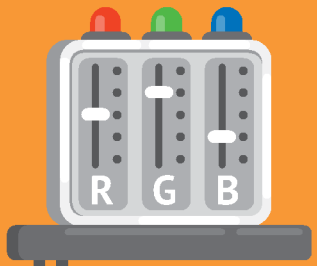
- шесть Troyka-модулей;
- пять модулей, использующих аналоговые входы/выходы;
- три модуля, работающих по протоколу SPI (DI/DO/SCK).

Для подключения таких Troyka-модулей требуется два трёхконтактных разъёма. Один разъём используется для подачи напряжения и коммуникации с платой управления через пин CS. Другой — для подключения пинов SCK, MISO и MOSI.







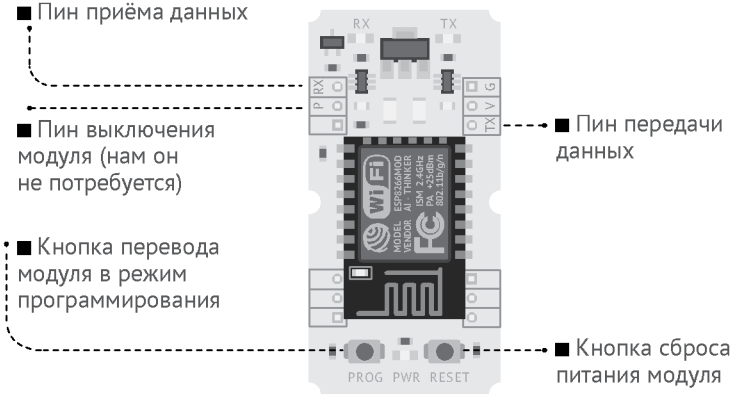


№ 1

НА СТАРТ, ВНИМАНИЕ, WI-FI!

Научимся подключать свои устройства к Wi-Fi сети. Для доступа к Wi-Fi воспользуемся специальным модулем. Он общается с Arduino по протоколу UART.

TROYKA-МОДУЛЬ WI-FI

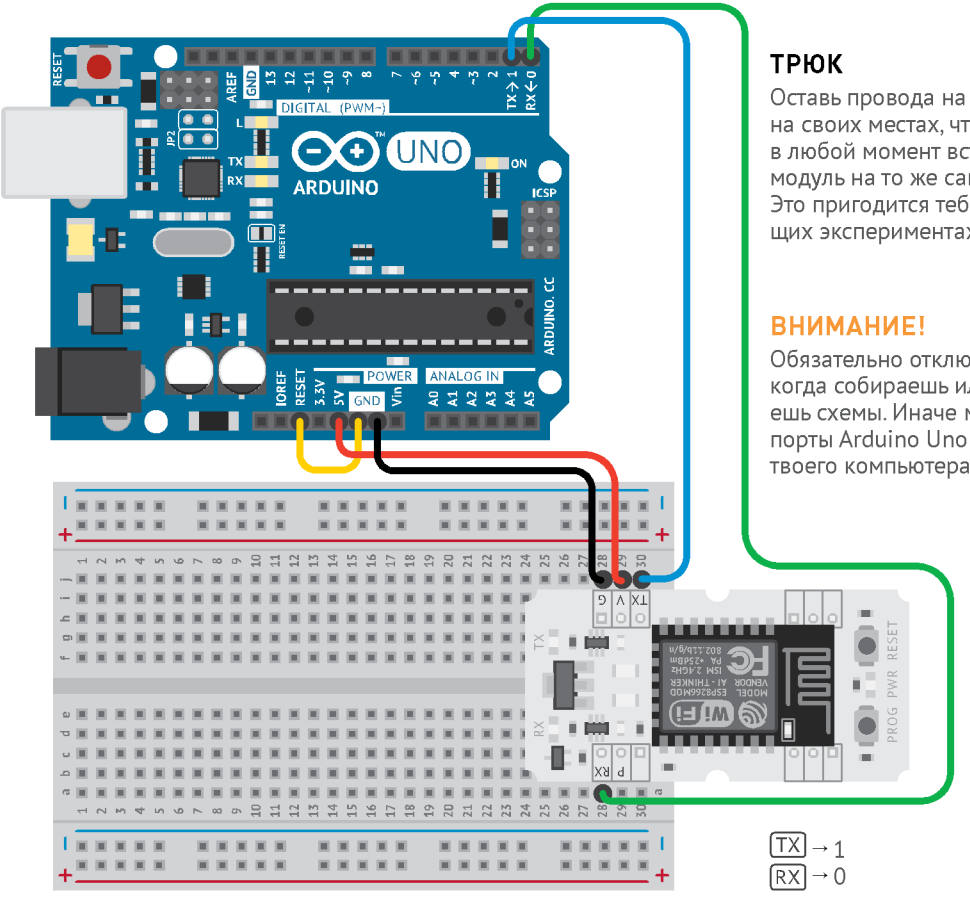


Стандартным способом связи между Arduino и Wi-Fi модулем являются AT-команды Это специальный набор команд, начинающихся с букв «AT» (от англ. ATtention — «Внимание»). Ты можешь поуправлять модулем вручную, отправляя команды в Serial-интерфейс (Serial Monitor в Arduino IDE). Это может пригодиться для поиска ошибок и отладки твоих программ.

СПРАВКА

Набор AT-команд был разработан в 1977 году и изначально предназначался для модемов, однако до сих пор широко используется.

1 Установи модуль Wi-Fi на Breadboard. Подсоедини проводами к Arduino UNO в режиме USB-UART-преобразователя. Не забудь замкнуть пины RESET и GND на Arduino UNO.



ТРЮК

Оставь провода на Breadboard'е на своих местах, чтобы ты мог в любой момент вставить Wi-Fi модуль на то же самое место. Это пригодится тебе в следующих экспериментах.

ВНИМАНИЕ!

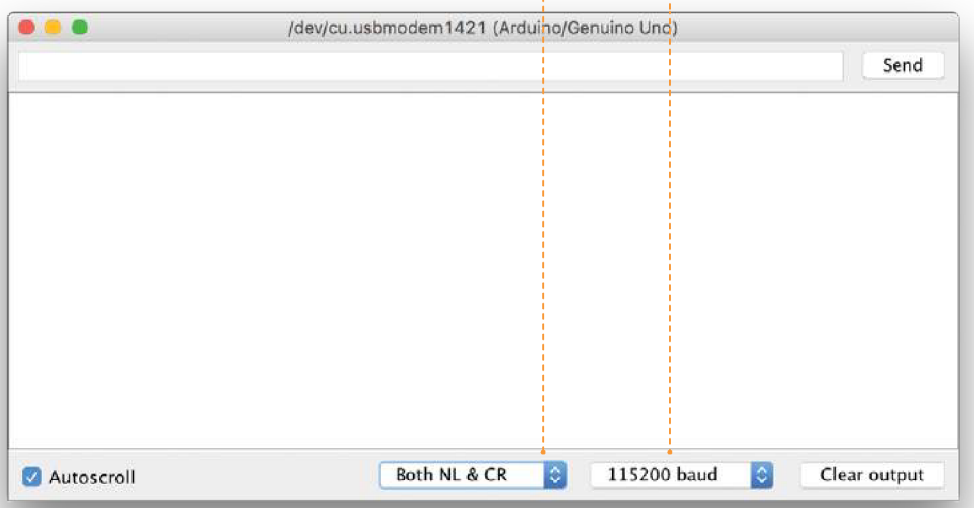
Обязательно отключай питание, когда собираешь или разбираешь схемы. Иначе можно сжечь порты Arduino Uno и даже USB твоего компьютера.

TX → 1
RX → 0

V → 5V
G → GND

RESET на Arduino замкнут с GND

2 Открой окно Serial Monitor в Arduino IDE (*Инструменты → Монитор порта*). Установи скорость обмена данными на 115 200 бод. Выстави параметр «Both NL & CR», это поможет тебе корректно отправлять AT-команды и видеть ответы на них.



3 Набери в командной строке большими буквами в латинской раскладке «AT» и нажми Enter. Это — базовая команда, которая покажет тебе, работает ли модуль и можешь ли ты корректно получить с него данные. Все следующие команды будут начинаться с «AT».

Модуль ответит «OK». Если ответа нет или появляются непонятные символы — проверь правильность подключения и настройки скорости обмена данными (например, попробуй выставить скорость 9 600 бод).

4 Теперь сделаем кое-что интересное. Этот модуль Wi-Fi имеет три режима:

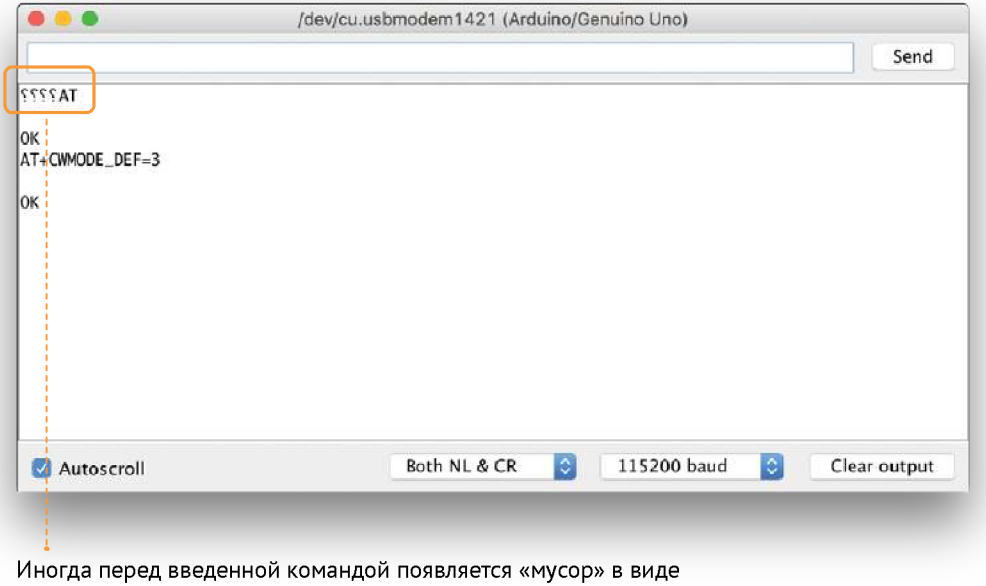
- Режим точки доступа (тогда к нему смогут подключаться клиенты как к Wi-Fi роутеру).
- Режим клиента (для подключения к точкам доступа).
- Смешанный режим, когда модуль может быть и точкой доступа и клиентом.

Для начала переведи модуль в смешанный режим командой `AT+CWMODE_DEF=3`.

Модуль должен ответить «OK».

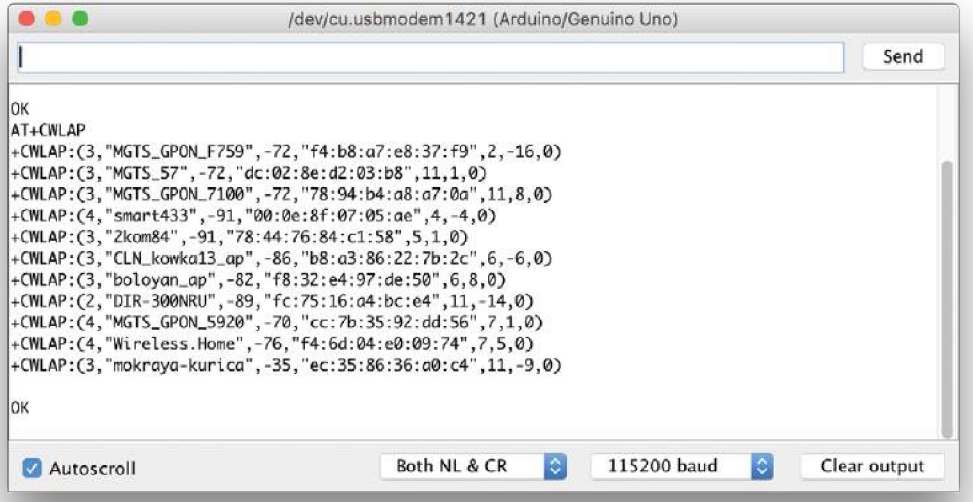
ВНИМАНИЕ!

Окончание _DEF означает, что эти настройки сохранятся в памяти модуля и он будет использовать их при следующем включении.



Иногда перед введенной командой появляется «мусор» в виде знаков вопроса или непонятных символов. Не обращай на это внимания.

5 Набери в консоли AT+CWLAP .



Эта команда отдаёт в Serial Monitor список всех точек доступа, к которым можно подключиться. Её ответ состоит из строк с параметрами, разделёнными запятой:

- Метод шифрования точки доступа — 0: открытая, 1: WEP, 2: WPA_PSK, 3: WPA2_PSK, 4: WPA_WPA2_PSK.
- SSID (имя) точки доступа (в кавычках).
- Уровень сигнала. Чем больше числовое значение по модулю, тем лучше сигнал.
- Дополнительная служебная информация (сейчас она не понадобится).

6 Командой AT+CWJAP="SSID","password" выполняется подключение к сети. У этой команды есть аргументы — SSID и пароль от точки доступа. В команде нет пробелов — проверь, что ты случайно не поставил его после запятой.

На эту команду Serial Monitor ответит тебе:

```
/dev/cu.usbmodem1421 (Arduino/Genuino Uno)

+CWLAP:(3,"mokraya-kurica",-51,"ec:55:86:56:d0:c4",11,-9,0)
+CWLAP:(0,"DIR-300NRU.2",-89,"fc:75:16:a4:bc:e5",11,-12,0)
+CWLAP:(3,"2kom84",-90,"78:44:76:84:c1:58",5,3,0)
+CWLAP:(3,"MGTS_GPON_B5DC",-86,"70:9f:2d:cc:e5:d6",6,-21,0)
+CWLAP:(3,"RT-WiFi_67",-78,"00:1f:ce:f2:a9:13",6,23,0)
+CWLAP:(4,"MGTS_GPON_5920",-71,"cc:7b:35:92:dd:56",7,1,0)
+CWLAP:(4,"Wireless.Home",-70,"f4:6d:04:e0:09:74",7,5,0)
+CWLAP:(3,"MGTS_57",-71,"dc:02:8e:d2:03:b8",11,1,0)

OK
AT+CWJAP="Wireless.Home","23972828"
WIFI DISCONNECT
WIFI CONNECTED
WIFI GOT IP

OK

Autoscroll        Both NL & CR        115200 baud
```

1 Убеждаемся в том, что предыдущее соединение разорвано.

2 Подключаемся к точке доступа.

3 Получаем IP-адрес от точки доступа.

Если ты хочешь узнать IP- и MAC-адреса своего модуля, используй команду AT+CIFSR.

```
/dev/cu.usbmodem1421 (Arduino/Genuino Uno)        Send

OK

ERROR
AT+CIFSR
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"5e:cf:7f:d3:3e:7f"
+CIFSR:STAIP,"192.168.2.30"
+CIFSR:STAMAC,"5c:cf:7f:d3:3e:7f"

OK
```

7 Теперь создай сервер на Troyka-модуле Wi-Fi!

Для этого разреши несколько одновременных соединений командой `AT+CIPMUX=1`.

Команда для старта сервера — `AT+CIPSERVER=1,80. 1` — запуск сервера, 80 после запятой означает номер порта, по которому сервер будет слушать запросы.

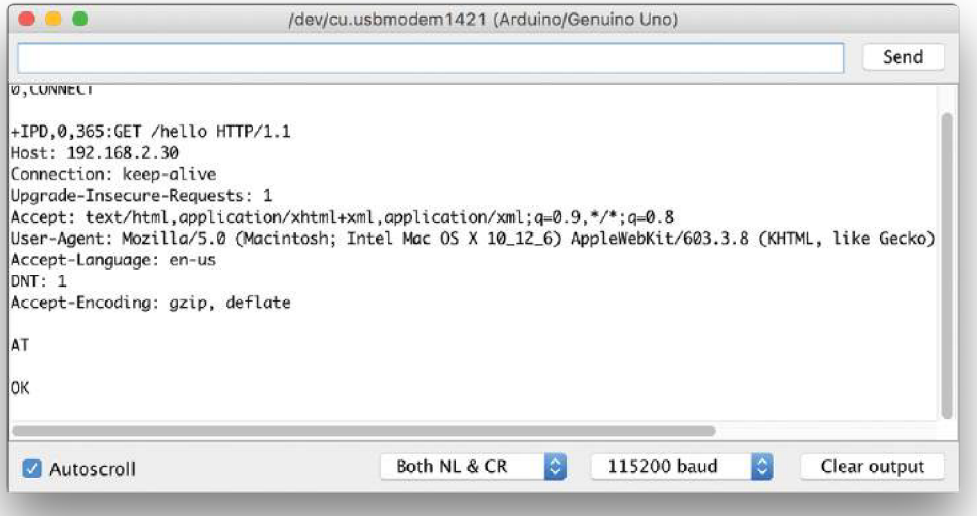
Если в Serial Monitor отобразилась надпись «OK» — значит, сервер стартовал и у тебя всё получилось!

Поздравляем, ты поднял свой первый сервер!

8 После выполнения этой команды проверь, не изменился ли IP-адрес твоего устройства уже известной тебе командой. Затем введи в адресную строку браузера: `ip_адрес_устройства/hello.`

Номер порта при запросе можно не указывать — по умолчанию все браузеры используют 80-й порт.

Serial Monitor отобразит запрос, который поступит от твоего браузера. Запрос будет содержать в себе большое количество информации. Раздел «Протокол HTTP» поможет тебе в ней разобраться.

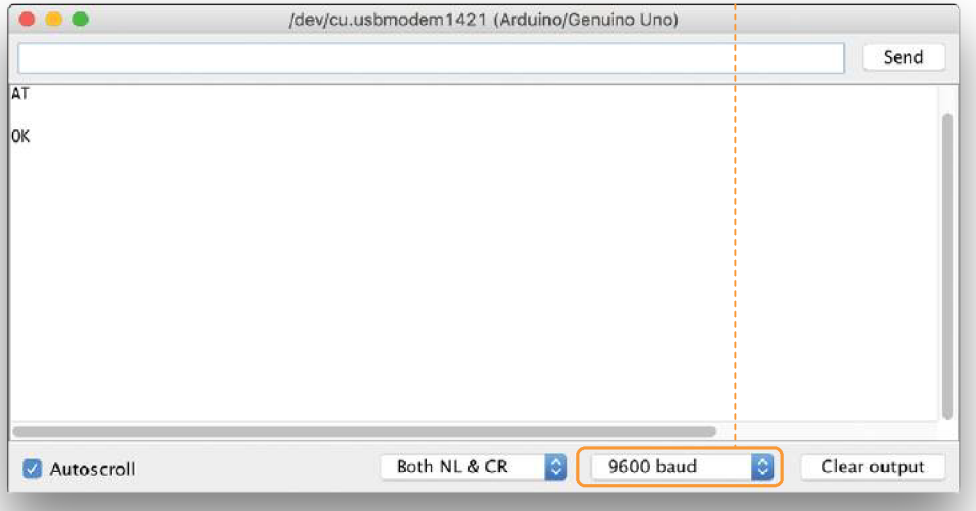


9 Теперь останови сервер командой `AT+CIPSERVER=0,80`. Пока что он не нужен.

10 Последнее, что нужно сделать, — изменить скорость общения Troyka-модуля Wi-Fi с Arduino Uno на 9 600 бод. Это потребуется для следующих экспериментов. Воспользуйся командой `AT+UART_DEF=9600,8,1,0,0`.

После этой операции не забудь переключить в Serial Monitor скорость обмена данными на 9 600 бод.

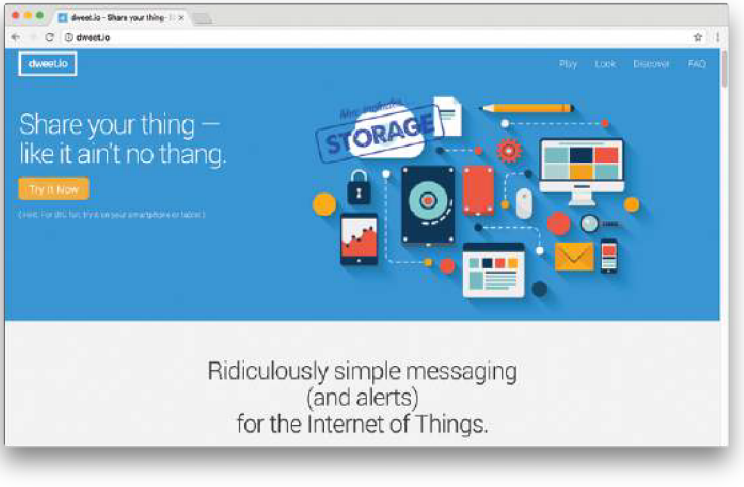
Отправь модулю базовую команду AT. Убедись, что он отвечает OK. Все остальные команды из этого эксперимента также будут работать и при этой скорости.



№ 2 УДАЛЁННЫЙ ТЕРМОМЕТР

Соберём устройство для наблюдения за температурой через интернет. Сервис dweet.io умеет строить красивый график по принимаемым данным. Будем отправлять ему данные о температуре.

Каждую секунду сервис получает информацию от тысяч устройств. Чтобы различать устройства между собой, они должны сообщить свой уникальный ключ, по которому можно однозначно определить отправителя. Ключ нужно придумать самому. Он должен состоять только из латинских букв и цифр, без пробелов. Создай в качестве такого ключа строку, состоящую из твоего имени и даты рождения, например: «Amperka03042011» (Амперка, 03 апреля 2011).

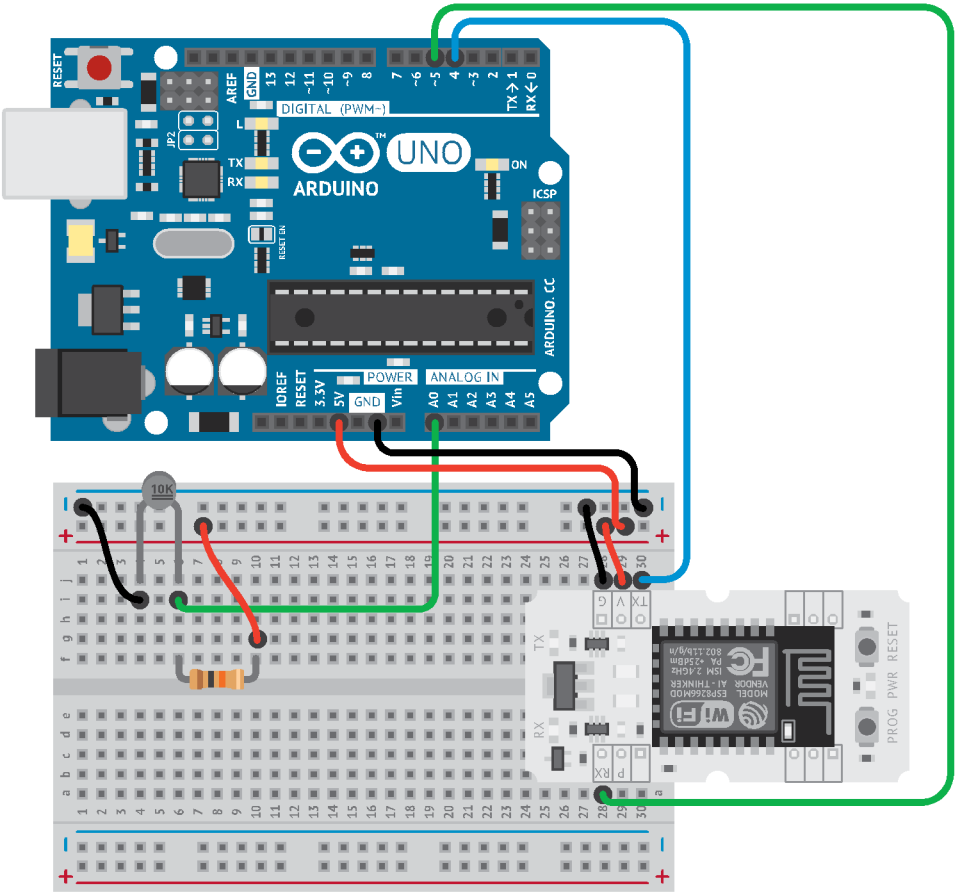


Подключи к Arduino UNO Wi-Fi модуль и датчик температуры. Wi-Fi модуль будет общаться с Arduino по программному UART (Software Serial).

Не забудь использовать делитель напряжения — резистор номиналом 10 кОм.

ВНИМАНИЕ!

Внимание! Сначала обязательно пройди эксперимент «На старт, внимание, Wi-Fi!», если вдруг ты его пропустил!



Wi-Fi модуль:
TX → P4
RX → P5

Термистор → A0

Резистор на 10 кОм

ВНИМАНИЕ!

Не набирай код программы вручную, скопируй его с сайта iot-m.amperka.ru.

1 Подключаем необходимые библиотеки: `ESP8266.h` для работы с Wi-Fi модулем и `SoftwareSerial.h` для использования программного UART. Библиотека `math.h` потребуется для расчета логарифма. Кавычки вида <> означают, что мы используем библиотеку, встроенную в Arduino IDE. Кавычки вида " " — что мы используем внешнюю библиотеку.

2 Задаём пин подключения термистора, указываем SSID и пароль Wi-Fi сети, с которой будем выходить в интернет.

3 Говорим, что Arduino Uno будет работать с Wi-Fi модулем на программном UART на пинах 4 RX и 5 TX. Задаём `"твой_ключ"` для dweet.io.

4 Запускаем аппаратный UART на скорости 9600 бод. Он потребуется для вывода информации в Serial Monitor.

```
1   #include "ESP8266.h"
2   #include <SoftwareSerial.h>
3   #include <math.h>
4
5   #define SSID     "имя_твоего_Wi-Fi"
6   #define PASSWORD "пароль_твоего_Wi-Fi"
7   #define TEMP_PIN A0
8
9   SoftwareSerial mySerial(4, 5);
10  ESP8266 wifi(mySerial);
11  String name = "твой_ключ";
12
13  void setup(void) {
14    Serial.begin(9600);
15    if (wifi.joinAP(SSID, PASSWORD)) {
16      Serial.println("https://dweet.io/follow/" + name);
17    } else {
18      Serial.println("Wi-Fi connection error");
19    }
20  }
21
22  void loop(void) {
23    float v_temp = 1023.0 / analogRead(TEMP_PIN) - 1.0;
24    float temp = 1.0 / ( -log(v_temp) / 3977.0
25                         + 1.0 / 295.0 ) - 273.0;
26    if (wifi.createTCP("www.dweet.io", 80)) {
27      String data = "GET /dweet/for/" + name + "?";
28      data += "temp=" + String(temp) + " HTTP/1.1\r\n";
29      data += "Host: dweet.io\r\n\r\n";
30      wifi.send(data.c_str(), data.length());
31      wifi.releaseTCP();
32    } else {
33      Serial.println("create TCP error");
34    }
35    delay(1000);
36  }
```

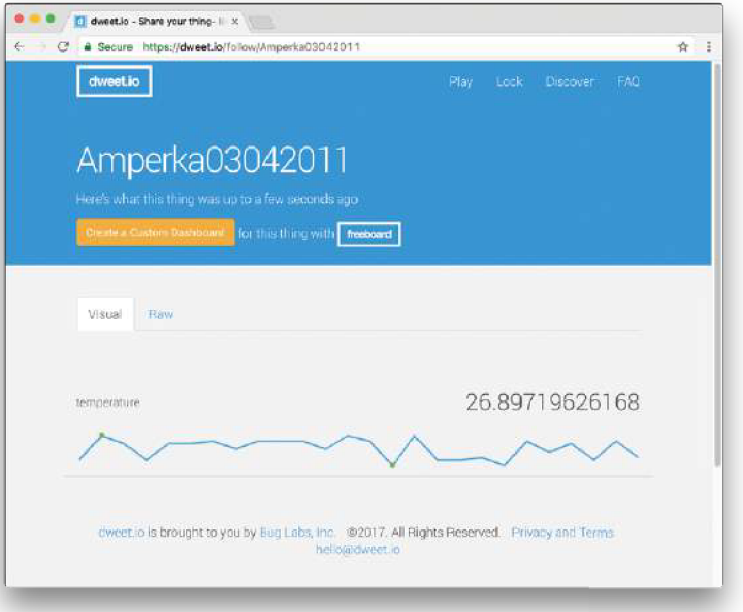
5 Подключаемся к Wi-Fi сети. Если подключение пройдёт успешно, в Serial Monitor появится ссылка на страницу с графиком температуры. В случае ошибки подключения в Serial Monitor появится сообщение «Wi-Fi connection error».

ВНИМАНИЕ!

Если в Serial Monitor появляются ошибки подключения к сети, внимательно проверь SSID, пароль сети и все провода. Помни: RX модуля нужно подключать в TX Arduino, а TX — в RX.

В меню *Инструменты* → *Плата* выбери плату «Arduino/Genuino Uno». Укажи порт подключения твоей Arduino.

Загрузи код в Arduino Uno, открой Serial Monitor и дождись ссылки (около 10 секунд). Скопируй ссылку в адресную строку браузера и нажми Enter. В браузере откроется страница сервиса dweet.io с графиком температуры, обновляющимся раз в секунду.



Теперь можешь отключить Arduino от компьютера и перенести термометр в любое место в доме, где есть сигнал Wi-Fi-роутера. Подключи прибор к источнику питания (например, к зарядке мобильника) — и Arduino снова начнёт посылать данные в dweet.io!

6 Считываем показания термистора как в эксперименте №15 «Конспекта хакера».

7 Открываем TCP-соединение к серверу dweet.io. Если соединение установится, выполним запрос к dweet.io. Если нет — в Serial Monitor появится сообщение «create TCP error».

8 «Собираем» HTTP-запрос и внутрь вставляем значение датчика температуры.

9 Отправляем запрос на сервер. Закрываем TCP-соединение.

10 Сервис dweet.io позволяет отправлять данные не чаще 1 раза в секунду, поэтому нет смысла делать задержку между отправками меньше 1 000 миллисекунд.

№ 3

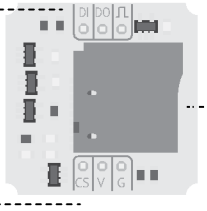
СИСТЕМА РЕГИСТРАЦИИ ДАННЫХ

Научимся снимать показания с датчиков температуры и освещённости и записывать их в файл на microSD-карточке. Будем использовать формат .csv, понятный для Microsoft Excel и подобных программ. Так мы сможем с лёгкостью строить графики и следить, как меняются температура и освещённость в течение больших периодов времени.

TROYKA-МОДУЛЬ SD КАРТРИДЕР

Модуль SD картридер позволяет читать файлы с MicroSD-карты и записывать их на неё. Картридер работает по интерфейсу SPI.

■ Пины
MOSI (DI — Digital Input, цифровой вход)
MISO (DO — Digital Output, цифровой выход)
SCK (Л — Clock, тактирование)



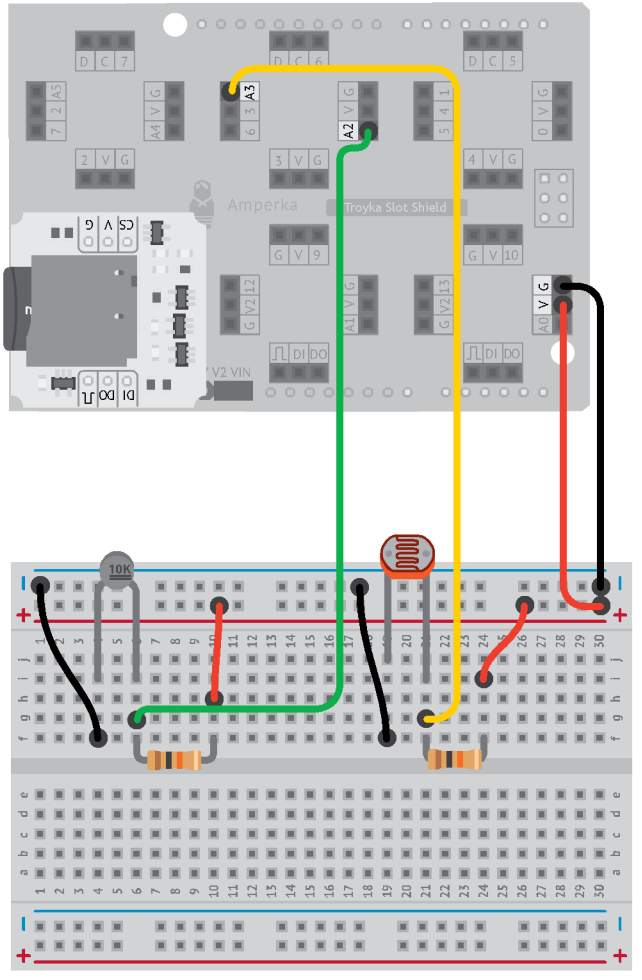
Вставь в картридер microSD карту из набора

■ Пин CS — Chip Select (выбор ведомого)

ВНИМАНИЕ!

Arduino может работать только с картами памяти, отформатированными в FAT32.

Собери схему с аналоговыми датчиками: термистором и фоторезистором. Похожие ты уже собирал в экспериментах 4, 5 и 16 «Конспекта хакера».



SD картридер
CS → P8
DO → DI
DI → DO
SCK → SCK

Резисторы на 10 кОм

```
1   #include <SPI.h>
2   #include <SD.h>
3   #include <math.h>
4
5   #define LIGHT_PIN A3
6   #define TEMP_PIN A2
7   #define CS 8
8
9   void setup() {
10    Serial.begin(9600);
11
12
13    if (!SD.begin(CS)) {
14      Serial.println("initialization failed!");
15      return;
16    }
17  }
18
19  void loop() {
20    float r_light = 10.0
21              / (1023.0 / analogRead(LIGHT_PIN) - 1.0);
22    float light = 10.0 * pow(14.0 / r_light, 1.6);
23    float v_temp = 1023.0 / analogRead(TEMP_PIN) - 1.0;
24    float temp = 1.0 / ( -log(v_temp) / 3977.0
25                          + 1.0 / 295.0 ) - 273.0;
26    String data = String(millis() / 1000) + ";"
27                + String(light) + ";" + String(temp);
28    data.replace(".", ",");
29    Serial.println(data);
30
31    File logFile = SD.open("log.csv", FILE_WRITE);
32    logFile.println(data);
33    logFile.close();
34
35    delay(1000);
36  }
```

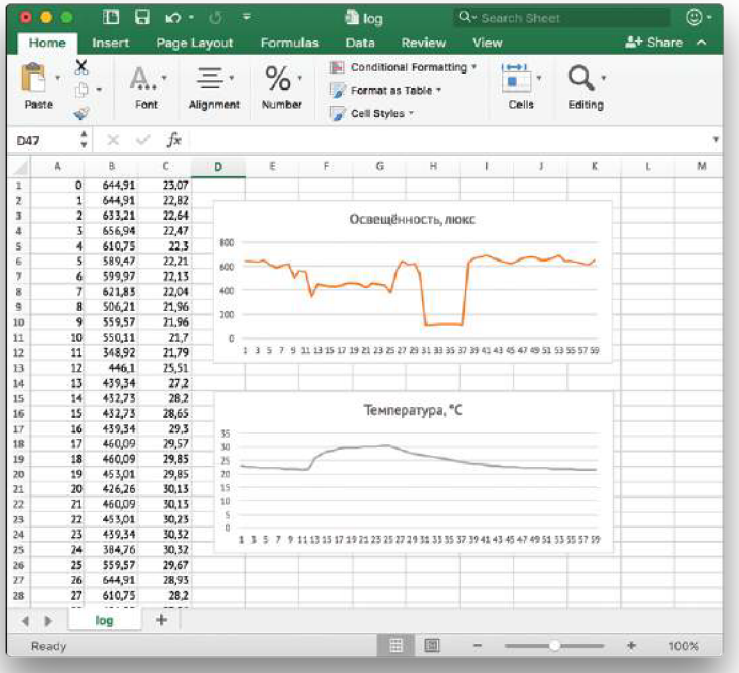
1 Задаём пин Chip Select для SD-картридера.

2 Инициализируем подключение к картридеру. Если что-то пошло не так — в Serial Monitor выводим сообщение об ошибке.

3 Считываем данные датчика освещения. Используем хитрую формулу с функцией pow, которая позволяет возвести число в дробную степень. Значение выводим в люксах (lx).

4 Собираем данные в единую строку с помощью функции `String`. Функция `millis()` возвращает время с момента включения Arduino в миллисекундах. Формат `.csv` (понятный для Microsoft Excel и подобных ему программ) требует разделения значений в строке точкой с запятой. Строку записываем в переменную `data`.

Собери данные. Отключи Arduino от питания. Вынь SD-карту из картридера и открой её на компьютере.



5 Дробные числа в Arduino записываются через точку, а русскоязычные версии программ Excel требуют разделять дробные части с помощью запятой. Заменяем все точки в строке на запятые командой `replace`. Если у тебя англоязычная версия — просто удали эту строку кода.

6 Открываем на SD-карте файл для записи данных. Аргумент `FILE_WRITE` указывает, что мы будем записывать данные в этот файл. При открытии файла для записи команда open проверяет, есть ли в корне SD-карты файл «log.csv». Если такого файла нет — он создаётся автоматически.

7 Записываем в файл строку `data`. Закрываем файл.

8 Ставим задержку в миллисекундах. Например, если мы хотим снимать показания датчиков один раз в 10 минут, задержка должна быть в 1000 × 60 × 10 = 600 000 миллисекунд.

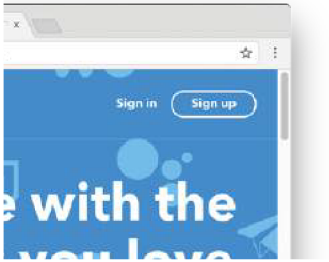
№ 4 НАПОМИНАЛЬНИК

Соберём систему напоминаний. На твой e-mail может приходить письмо, содержащее время нажатия на кнопку. Для этого воспользуемся сторонним сервисом.

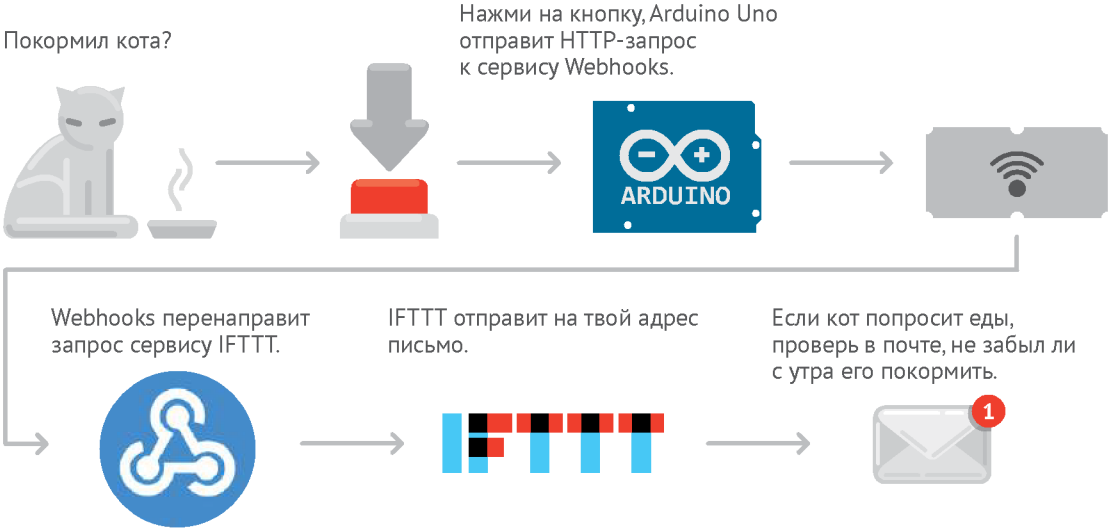
Для отправки e-mail используется особый протокол: SMTP. Он не очень удобен для работы напрямую с Arduino. Гораздо проще использовать готовые сервисы. Один из таких — IFTTT. Тебе понадобится зарегистрироваться в нём.

IFTTT (IF This Then That — если это, сделай то). Сервис позволяет подключать множество компонентов друг к другу по принципу «если произошло событие А, сделай действие Б». Комбинации действий и условий называются *апплетами* (applet, application — приложение, -let — уменьшительный суффикс).

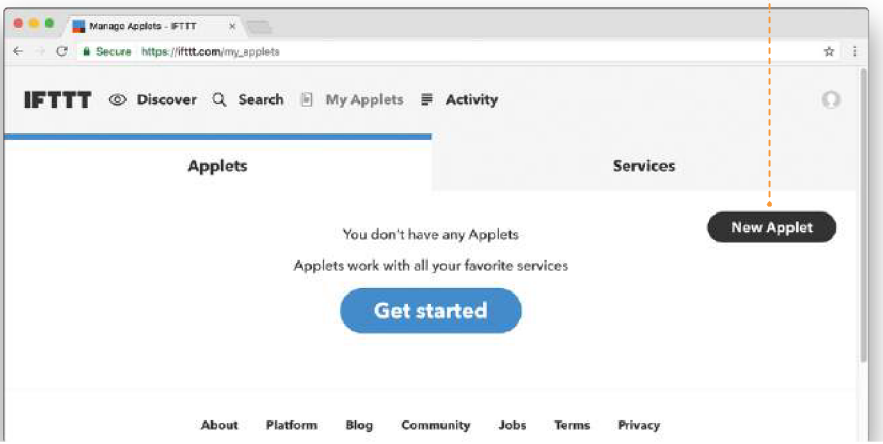
Компонент Webhooks сервиса IFTTT умеет принимать простые HTTP-запросы, а компонент Email — отправлять письма.



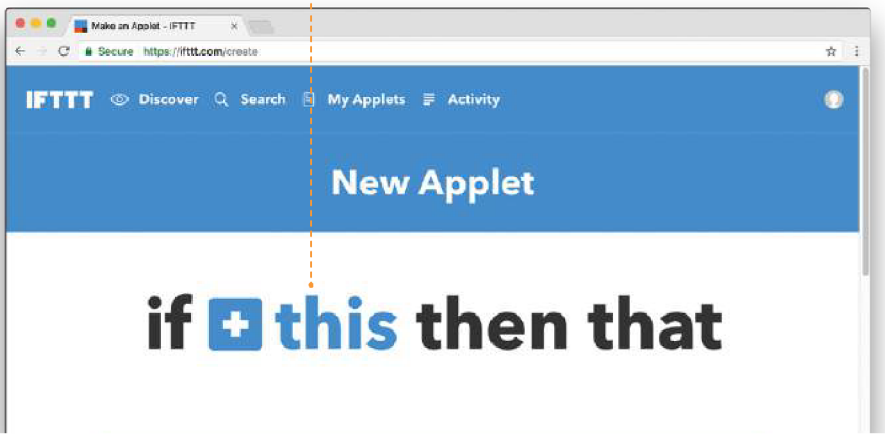
Заведи себе учётную запись на сайте ifttt.com



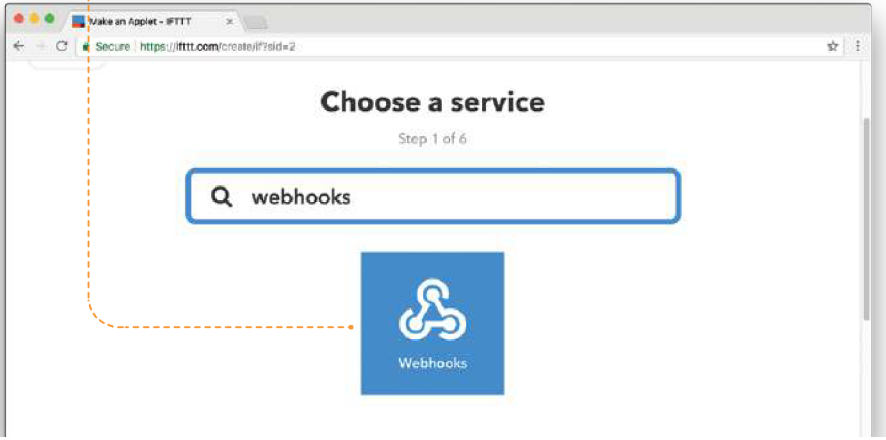
1 Зайди на ifttt.com. Нажми вверху кнопку «My Applets», а затем «New Applet».



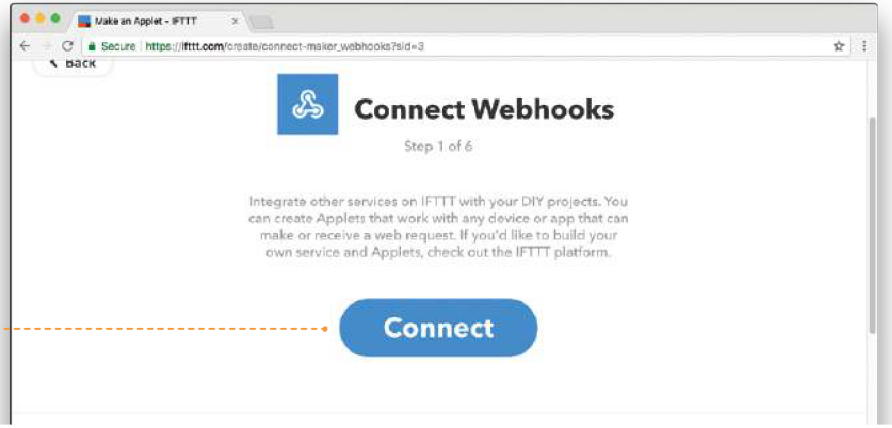
2 Нажми кнопку «this».



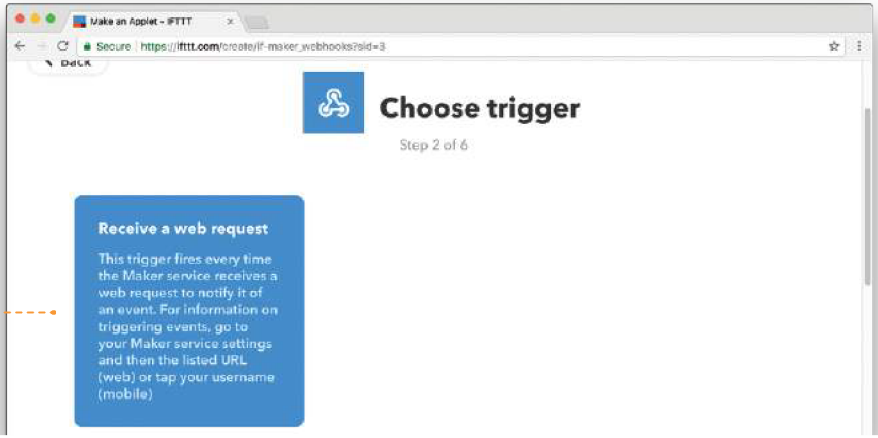
3 Набери в поисковой строке «Webhooks». В появившемся списке выбери «Webhooks».



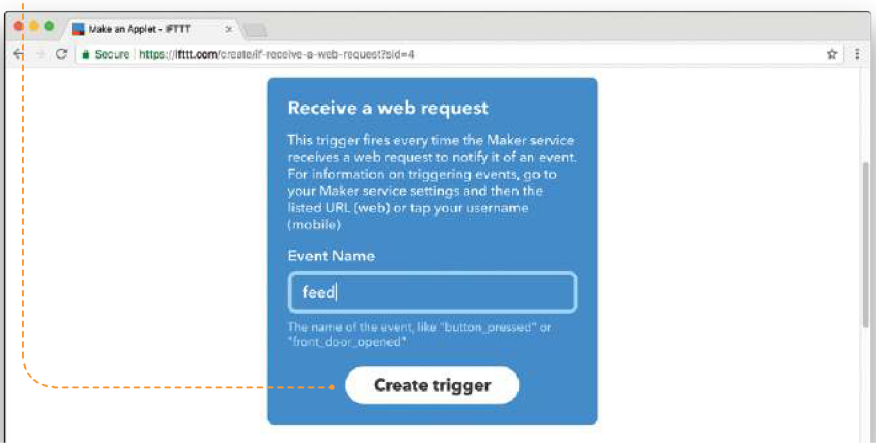
4 Нажми «Connect».



5 Выбери действие «Receive a web request».

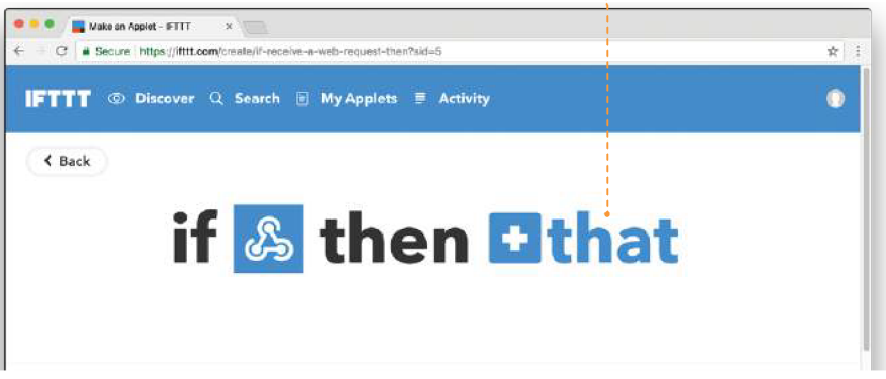


6 Поле «Event Name» назови «feed» и нажми кнопку «Create trigger».

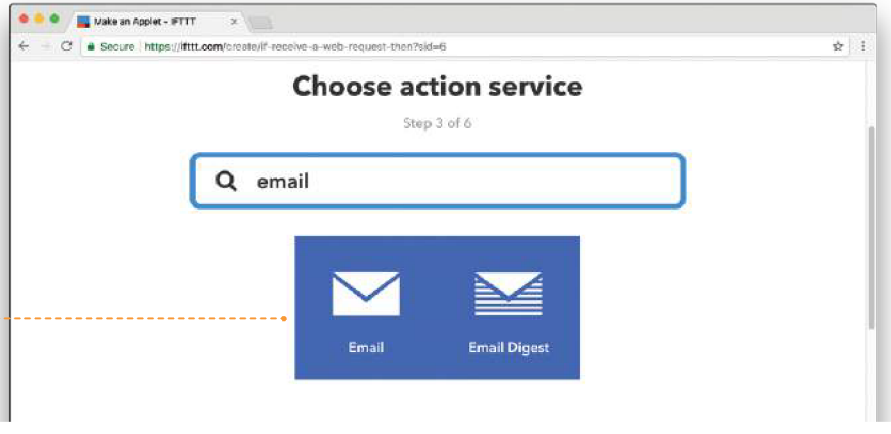


Ты создал условие (триггер). Теперь компонент Webhooks готов принимать HTTP-запросы от Arduino. Это похоже на работу сервиса dweet.io, но Webhooks не строит графики, а передаёт информацию дальше другим компонентам. Теперь нужно задать действие при срабатывании твоего триггера.

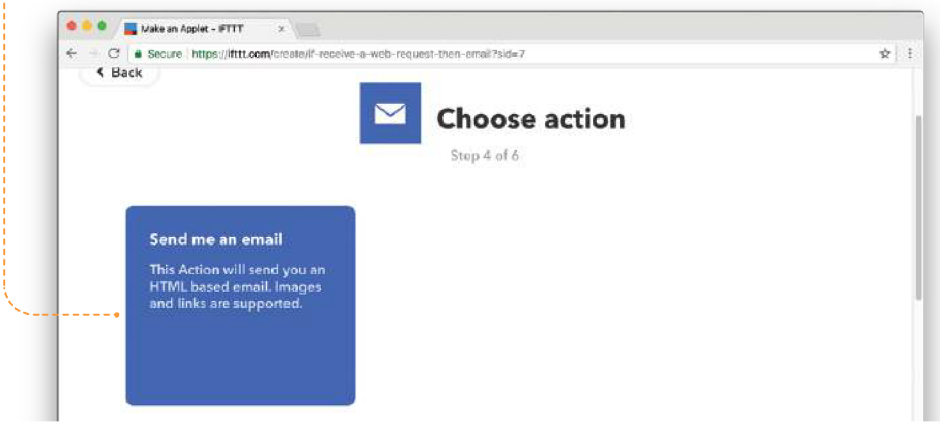
7 Нажми кнопку «that».



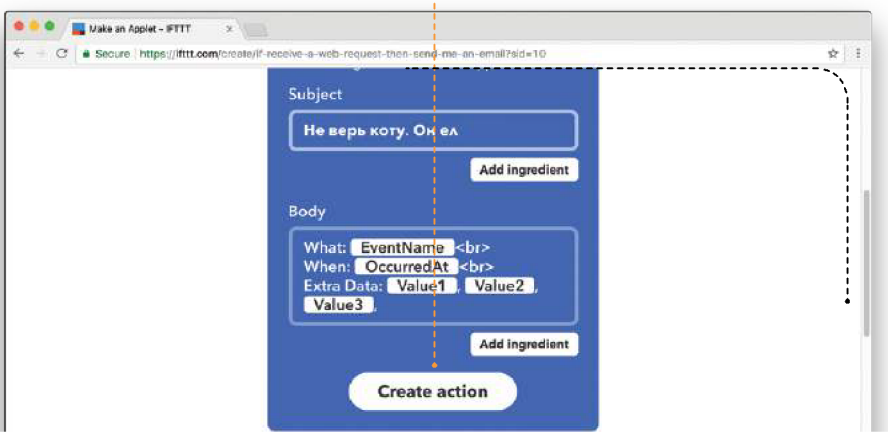
8 Набери в поисковой строке «email» и выбери появившийся компонент.



9 Выбери действие «Send me an email».

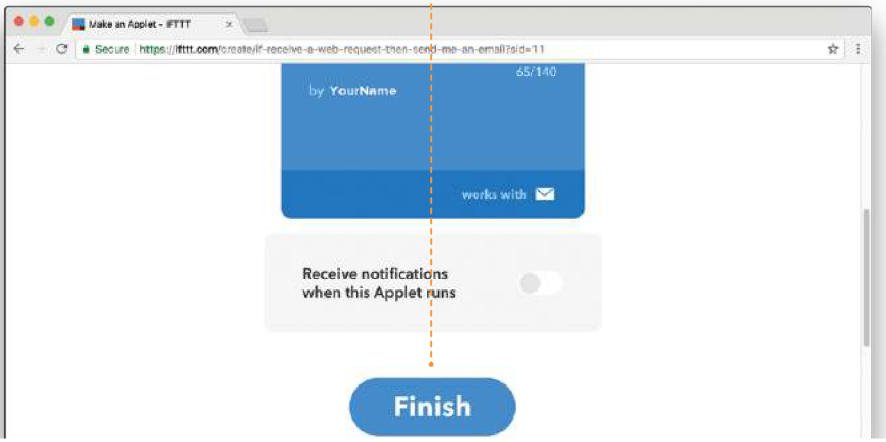


10 Задай образец письма. Придумай заголовок (Subject), а тело письма (Body) оставь нетронутым. Затем нажми «Create action».

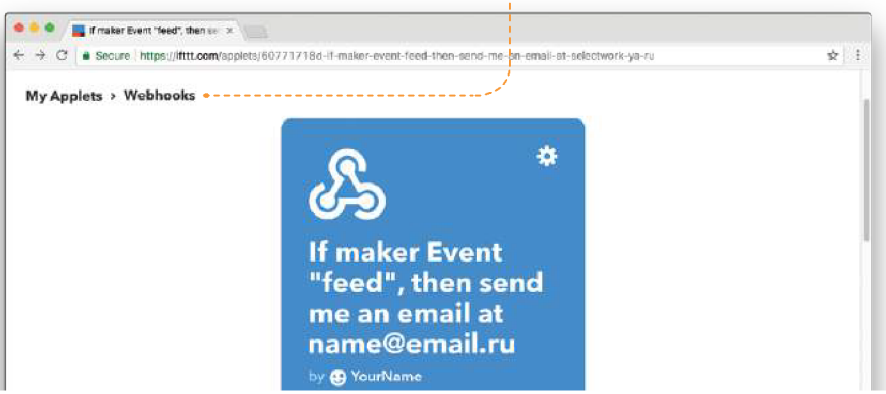


Поля Value1, Value2 и Value3 позволяют передавать в письме дополнительные данные. Отправляя с Arduino запрос компоненту Webhooks, можно указать значения этих полей.

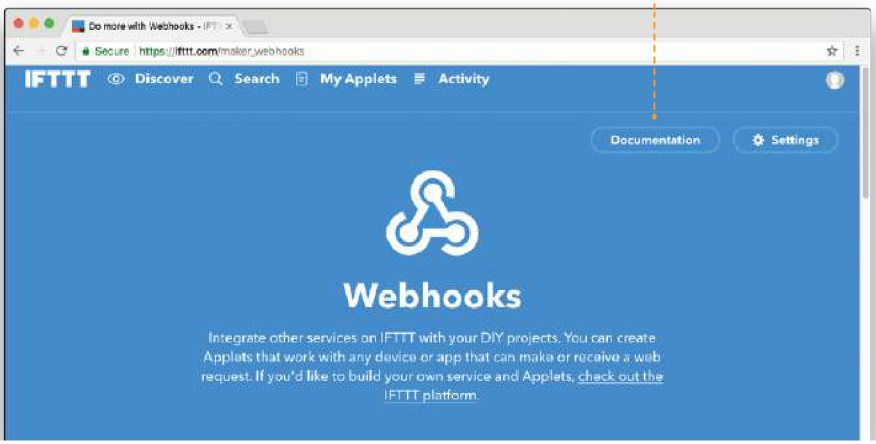
11 Наконец, жми «finish».



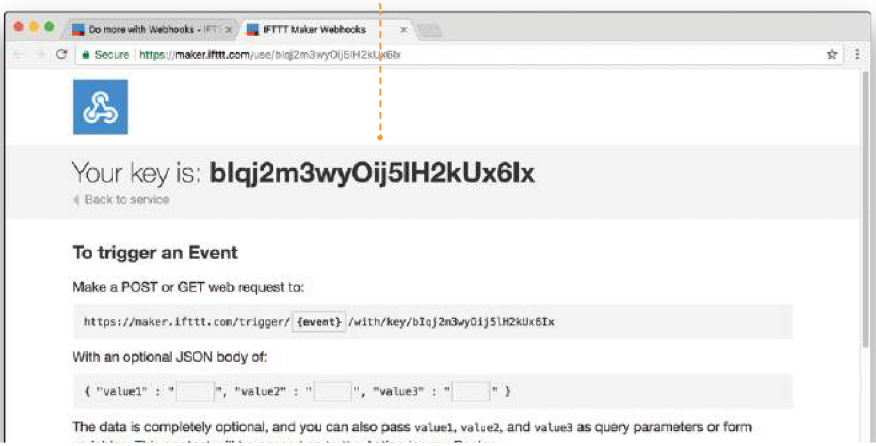
12 Webhooks использует уникальный ключ для идентификации устройства и выдаёт его самостоятельно. Чтобы узнать этот ключ, перейди во вкладку «My applets», нажми на иконку апплета и выбери вкладку «Maker Webhooks». Нажми на «Webhooks».



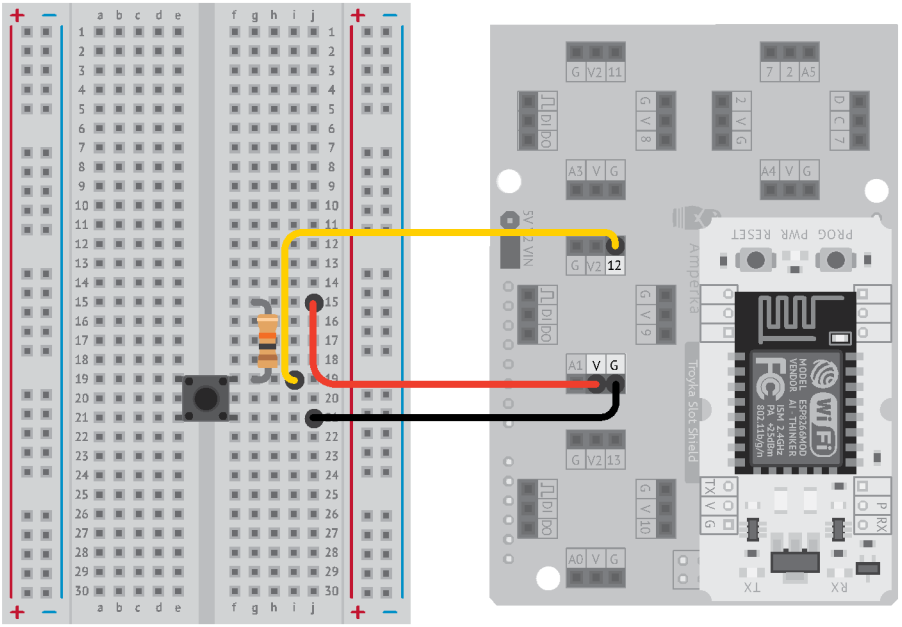
13 Нажми на кнопку «Documentation».



14 Webhooks сообщит тебе ключ для идентификации — его потребуется скопировать в код.



15 Установи кнопку на Breadboard и Wi-Fi модуль на Slot Shield.




```
#include "ESP8266.h"
#include <SoftwareSerial.h>

#define SSID        "имя_твоего_Wi-Fi"
#define PASSWORD    "пароль_твоего_Wi-Fi"
#define BTN_PIN     12

SoftwareSerial mySerial(4, 5);
ESP8266 wifi(mySerial);
boolean buttonWasUp = true;
String maker_ID = "id_компонента_webhooks";

void setup(void) {
  Serial.begin(9600);
  if (wifi.joinAP(SSID, PASSWORD)) {
    Serial.println("I'm ready! Press the button.");
  } else {
    Serial.println("Wi-Fi connection error");
  }
}
```

1 Запоминаем, была ли кнопка отпущена.

2 Заводим переменную типа `String`, в которой храним ID для компонента Webhooks.

```
21  void loop(void) {
22    boolean buttonIsUp = digitalRead(BTN_PIN);
23    if (buttonWasUp && !buttonIsUp) {
24      delay(10);
25      buttonIsUp = digitalRead(BTN_PIN);
26      if (!buttonIsUp) {
27        sendEmail();
28        Serial.println("Notification has been sent.");
29      }
30    }
31    buttonWasUp = buttonIsUp;
32  }
33
34  boolean sendEmail() {
35    if (wifi.createTCP("maker.ifttt.com", 80)) {
36      String value1 = "Hello!";
37      String request = "GET /trigger/feed/with/key/"
38               + maker_ID + "?value1=" + value1
39                          + " HTTP/1.1\r\n";
40      request += "Host: maker.ifttt.com\r\n\r\n";
41      wifi.send(request.c_str(), request.length());
42      wifi.releaseTCP();
43    } else {
44      Serial.println("create tcp error");
45    }
46  }
```

3 Обрабатываем нажатие кнопки точно таким же способом, как и в эксперименте 10 «Конспекта хакера».

4 Если программа зафиксирует клик — запустится функция `sendEmail`, а в Serial Monitor появится сообщение «Notification has been sent».

5 Функция `sendEmail` отправляет запрос компоненту Webhooks.

6 Поля `Value1`, `Value2` и `Value3` позволяют передавать в запросе дополнительные данные. Это могут быть числа или строки. В целом, использовать их не обязательно. Используем только параметр `Value1` со значением «`Hello!`».

16 Загрузи скетч в Arduino и открой Serial Monitor. Дождись сообщения, что Arduino подключилась к Wi-Fi.

```
I'm ready! Press the button.
```

17 Нажми и удерживай кнопку. В Serial Monitor увидишь надпись:

```
Notification has been sent.
```

В течение нескольких минут тебе на почту придёт сообщение с «Не верь коту. Он ел.» и дополнительным полем «`Hello!`»

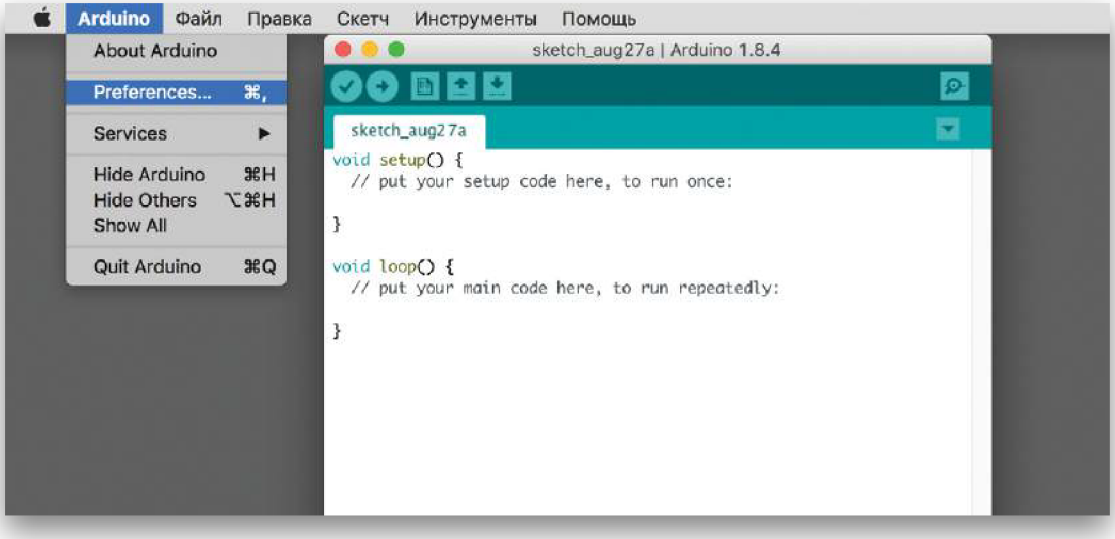
Теперь, когда ты с утра покормил кота, нажми на кнопку. Если кот днём попросит еды, покажи ему письмо — пусть не клянчит.

НАСТРОЙКА WI-FI МОДУЛЯ

Wi-Fi Troyka-модуль — очень умный модуль. Под металлической крышкой прячется целый микроконтроллер, который можно программировать через Arduino IDE. Он называется ESP8266, и он несколько мощнее того, что находится в сердце Arduino Uno. Например, его мощности хватит для работы с протоколом HTTPS, а это нужно для эксперимента с Telegram. Чтобы программировать Wi-Fi модуль в среде Arduino, её нужно настроить.

НАСТРОЙКА ARDUINO IDE

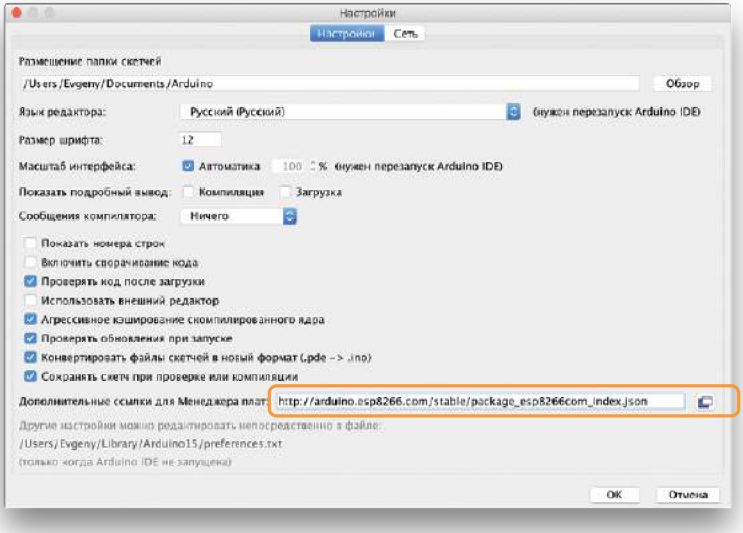
1 Открой в меню «Файл» пункт «Настройки».



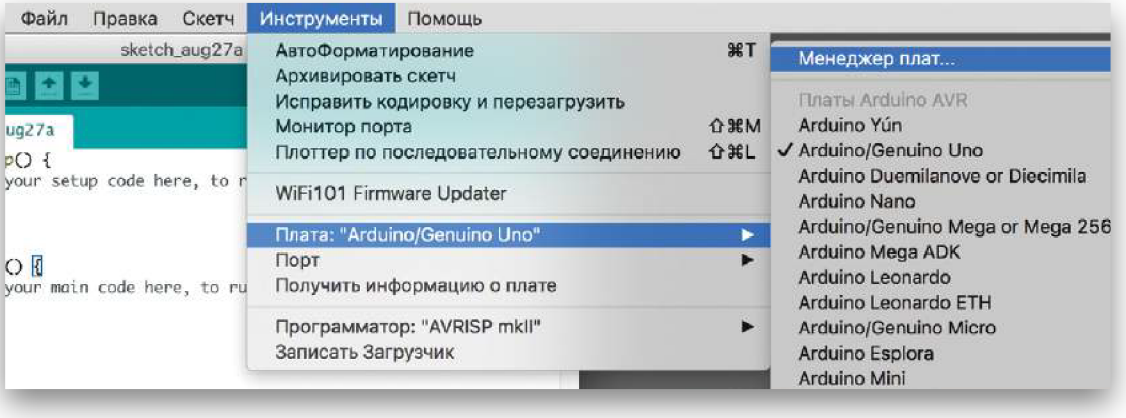
2 В окне настройки введи дополнительную ссылку для Менеджера плат:
http://arduino.esp8266.com/stable/package_esp8266com_index.json

ВНИМАНИЕ!

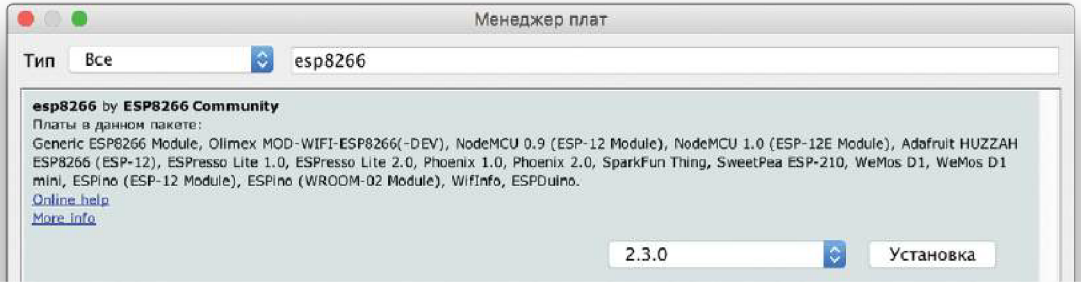
Ссылку можно скопировать на iot-m.amperka.ru → *«Программирование Wi-Fi модуля».*



3 Открой менеджер плат.

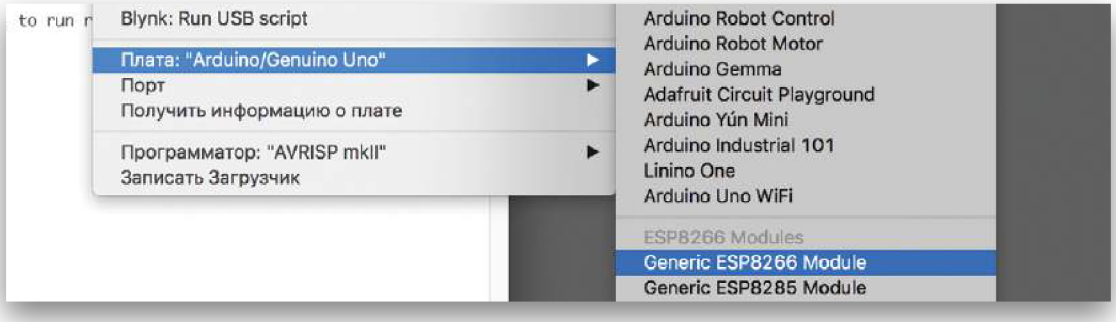


4 В поле для ввода набери «esp8266». Кликни по пункту «esp8266 by ESP8266 Community», выбери последнюю доступную версию и нажми «Установка».



ПРОГРАММИРОВАНИЕ МОДУЛЯ WI-FI

5 В Arduino IDE зайди в меню *«Инструменты»* → *«Плата»*. Выбери «Generic ESP8266 Module».

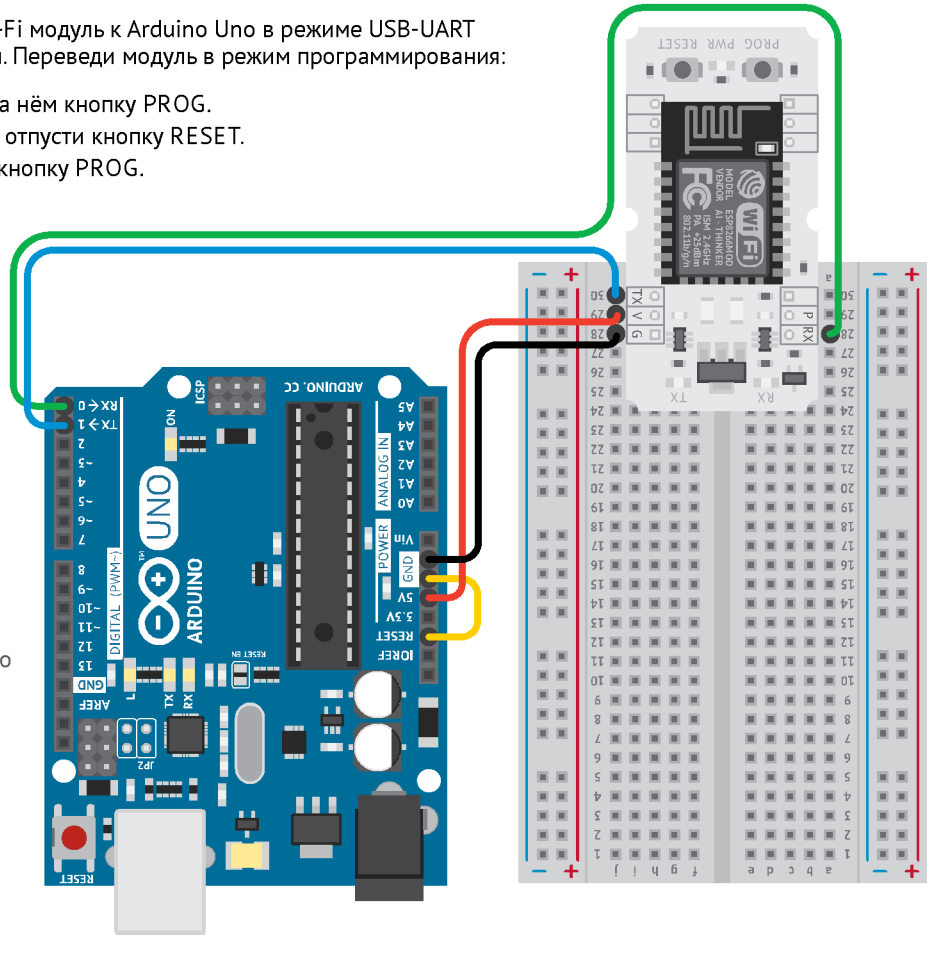


Остальные настройки должны выглядеть так:

- Плата: "Generic ESP8266 Module"
- Flash mode: "DIO"
- Flash Frequency: "40 MHz"
- CPU Frequency: "80 MHz"
- Flash Size: "512K (64K SPIFFS)"
- Debug port: "Disabled"
- Debug Level: "Ничего"

6 Подключи Wi-Fi модуль к Arduino Uno в режиме USB-UART преобразователя. Переведи модуль в режим программирования:

6.1 Нажми на нём кнопку PROG.
6.2 Нажми и отпусти кнопку RESET.
6.3 Отпусти кнопку PROG.



TX → 1
RX → 0

V → 5V
G → GND

RESET на Arduino замкнут с GND

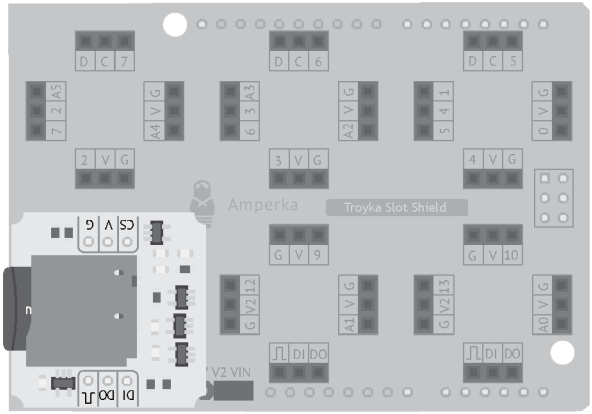
7 Всё готово, можно загружать код. Открой *«Примеры»* → *«Basics»* → *«Blink»*. Нажми кнопку «Загрузить». Светодиоды RX и TX будут быстро пульсировать, передавая код прошивки. Не пугайся, если код будет загружаться дольше обычного, — это связано с особенностями среды Arduino IDE и микроконтроллера ESP8266. После загрузки светодиод TX на модуле начнёт медленно мигать — это и делает Blink.

№ 5

БРАУЗЕРНЫЙ DENDY

Сделаем из Arduino сервер с браузерной игрой. Arduino будет читать с флеш-карты файл и передавать его в Wi-Fi модуль, а модуль обработает файл и покажет игру в браузере!

1 Для начала установи Slot Shield на Arduino Uno. Размести на нём модуль SD-картридера:



SD картридер
CS → P8
DO → DI
DI → DO
SCK → SCK

2 Запрограммируй Arduino, чтобы проверить, правильно ли она передаёт данные. Не забудь указать плату Arduino Uno в Arduino IDE (*«Инструменты» → «Плата» → «Arduino/Genuino Uno»*).

1 Запускаем Serial-соединение на скорости 115 200 бод и подключаем к пину 8 Chip Select модуля картридера. На скорости 115 200 бод файлы будут передаваться гораздо быстрее, чем на 9 600.

2 С помощью функции `Serial.find()` «слушаем» Serial-соединение, пока в нём не появится строка «race.htm». Как только эта строка там появится — открываем на SD-карте одноимённый файл функцией `SD.open()`.

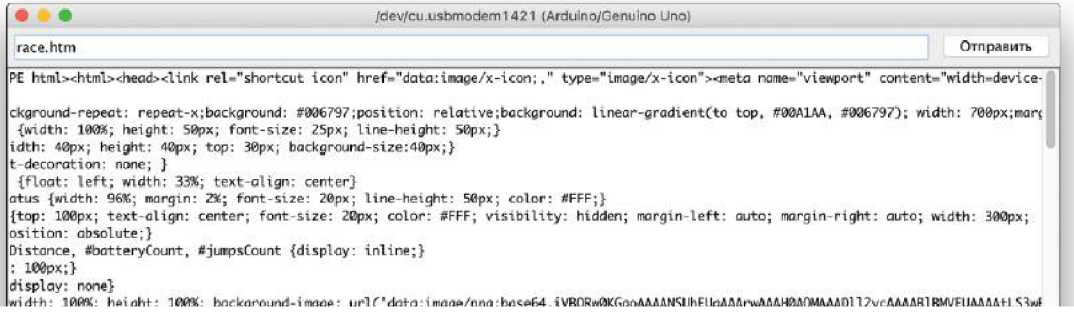
Код для Arduino Uno

```
#include <SPI.h>
#include <SD.h>

void setup() {
  Serial.begin(115200);
  SD.begin(8);
}

void loop() {
  if (Serial.find("race.htm")) {
    File myFile = SD.open("race.htm");
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    myFile.close();
  }
}
```

3 Считываем содержимое файла по одному байту и отправляем каждый байт в Serial-соединение. Как только файл прочитан полностью — закрываем его.

Нужно убедиться, что файл правильно читается. Загрузи код в Arduino Uno, открой Serial Monitor и установи скорость обмена данными в 115 200 бод. В нижнем левом выпадающем списке выстави параметр «Both NL & CR».

3 Набери в Serial Monitor команду «race.htm». В ответ Arduino отправит содержимое http-страницы с игрой в текстовом виде.



Пришла пора перепрограммировать Wi-Fi модуль под твои задачи! Сними Slot Shield с Arduino и временно отложи его в сторону.

4 Настрой Wi-Fi модуль для перепрошивки (схема сборки стр. 57): в Arduino IDE выбери в меню *«Инструменты»* → *«Плата»* → *«Generic ESP8266 Module»* и переведи Wi-Fi модуль в режим программирования:

- 4.1 Нажми на нём кнопку PROG.
- 4.2 Нажми и отпусти кнопку RESET.
- 4.3 Отпусти кнопку PROG.

5 Напиши код для ESP8266.

1 В переменной page будем хранить html-страницу с игрой.

2 Заводим переменную для работы модуля в качестве сервера. Число `80` означает, что сервер будет работать на 80-м порту.

3 Напишем функцию для обработки запроса клиентов. Назовём её `handleRoot()`.

4 Команда `server.send()` будет отправлять ответ на запросы клиентов, сформированный из служебной информации и самих данных. В качестве служебной информации разместим код `200` (он означает, что сервер правильно распознал запрос и корректно на него среагировал) и тип данных — в нашем случае это «`text/html`». Третий аргумент — тело HTTP-ответа.

Код для Wi-Fi модуля

```
1   #include <ESP8266WiFi.h>
2   #include <WiFiClient.h>
3   #include <ESP8266WebServer.h>
4
5   String page = "";
6   ESP8266WebServer server(80);
7
8   void handleRoot() {
9     server.send(200, "text/html", page);
10  }
11
12  void setup(void) {
13    Serial.begin(115200);
14    WiFi.begin("имя_твоего_Wi-Fi", "пароль_твоего_Wi-Fi");
15
16    while (WiFi.status() != WL_CONNECTED) {
17      delay(500);
18    }
19
20    Serial.println();
21    Serial.println(WiFi.localIP());
22
23    server.on("/race.htm", handleRoot);
24    server.begin();
25
26    Serial.println("race.htm");
27  }
28
29  void loop(void) {
30    if (Serial.available()) {
31      page += (char)Serial.read();
32    }
33    server.handleClient();
34  }
```

5 Ожидаем подключения к сети, проверяем каждые 500 миллисекунд. Как только соединение будет установлено — в Serial Monitor отобразится IP-адрес сервера с игрой.

6 Когда клиент запросит страницу `"/race.htm"`, вызовем функцию обработки `handleRoot()`. Запускаем сервер.

7 Отправляем в Serial-соединение команду `"race.htm"`. В ответ на неё Arduino начнёт посылать содержимое страницы по одному байту.

8 Собираем из байтов всю страницу. Каждый пришедший в Serial байт добавляем в строку `page` при помощи команды `+= (char)Serial.read()`.

9 Вызываем функцию `server.handleClient()`. В каждом цикле функции `loop` она проверяет, есть ли новые запросы. Если есть — отправляет их на обработку.

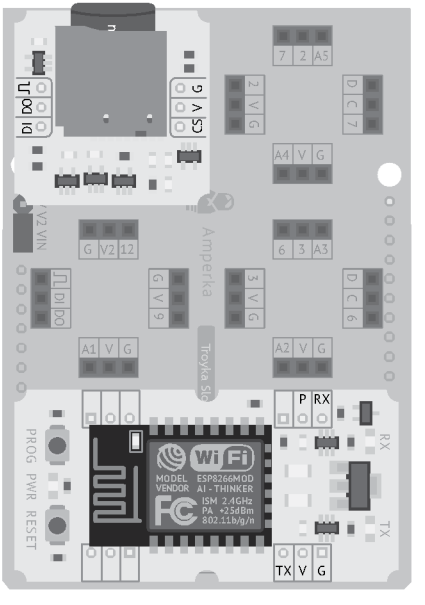
6 Загрузи код в Wi-Fi модуль. Открой Serial Monitor и дождись строки с IP-адресом. Сохрани его в блокноте, чтобы не забыть. Ниже ты увидишь строку «race.htm» — результат работы функции `Serial.println("race.htm");`. Это значит, что ESP8266 ждёт байты html-страницы.

7 Теперь собери свой сервер в единое устройство. Установи Slot Shield с модулем картридера обратно на Arduino Uno и добавь к ним прошитый модуль Wi-Fi:

SD картридер
CS → P8

DO → DI
DI → DO
SCK → SCK

Wi-Fi модуль:
TX → P0 RX
RX → P1 TX



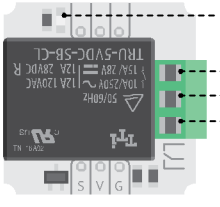
8 Готово! Обнови страницу в браузере — запусти игру и ставь рекорды!

№ 6 УМНЫЙ ДОМ

Продолжим использовать мощности Wi-Fi модуля для обработки информации. Соберём выключатель света, управляемый из web-интерфейса.

TROYKA-МОДУЛЬ МИНИ-РЕЛЕ

Мини-реле замыкает и размыкает электрические цепи подобно выключателю света на стене. Правда, выключатель управляется вручную, а реле управляется программой с Arduino. Реле способно работать под напряжением 220 вольт.



- Красный светодиод

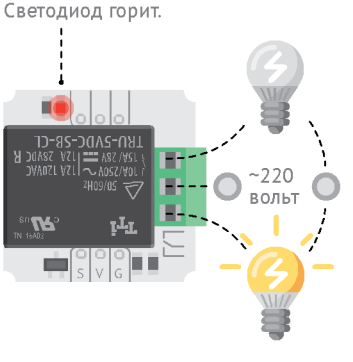
Контакты для подключения приборов с высоким напряжением:

- Нормально замкнутый
- Общий
- Нормально разомкнутый

Реле имеет 3 контакта на клеммах. В положении «выключено» (его ещё называют «нормальным» положением) реле замыкает правую пару контактов и размыкает левую. В состоянии «включено» всё наоборот: реле размыкает правую и замыкает левую пары контактов.

На пин S подаётся логический 0. Светодиод не горит.

На пин S подаётся логическая 1. Светодиод горит.



ВНИМАНИЕ!

РАБОТА С ВЫСОКИМ НАПРЯЖЕНИЕМ ОПАСНА ДЛЯ ЗДОРОВЬЯ И ЖИЗНИ.

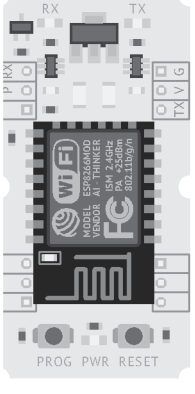
Если тебе ни разу не приходилось работать с напряжением 220 вольт, оставь зелёные клеммы неподключёнными.

Вместо этого используй светодиод на реле. Если горит — реле включено. Если нет — выключено.

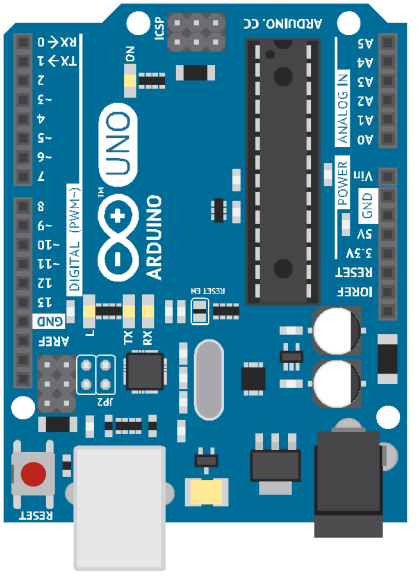
Это сделано для удобства и безопасности. Если на управляющих пинах (S-V-G) внезапно пропадёт питание, реле отключится и перейдёт в «нормальное» положение.

Представь, что в твоей комнате свет подключён к реле и Arduino Uno. Внезапно тёмным вечером Arduino Uno отключилась. Тебе хотелось бы, чтобы свет погас? Если ты хочешь, чтобы он остался гореть, нужно подключать лампочки к нормально замкнутой паре контактов. Если ты хочешь, чтобы свет потух, — к нормально разомкнутой. По правилам техники безопасности свет должен потухнуть.

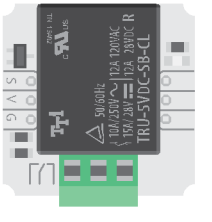
Включать и выключать реле ты будешь прямо из браузера. Wi-Fi модуль снова выступит сервером, работающим по протоколу HTTP. А Arduino Uno будет передавать в модуль html-страницу с «выключателем» и управлять реле.



Обрабатывает команды: принимает их из веб-интерфейса и отправляет в Serial.

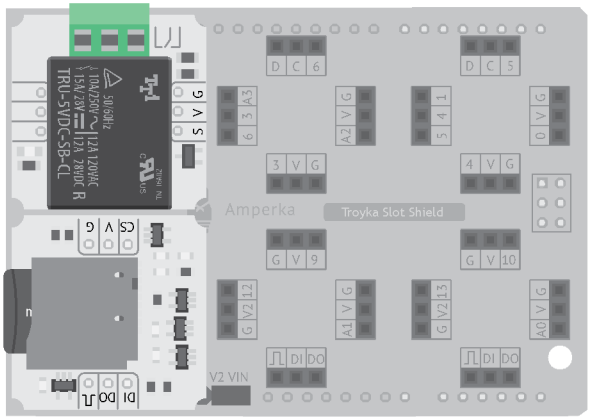


Постоянно проверяет Serial-соединение на наличие команд, и в случае их поступления включает или выключает реле.



Замыкает и размыкает силовые цепи.

1 Установи на Arduino Slot Shield с модулями картридера и мини-реле:



Мини-реле → A4

SD картридер
CS → P8

DO → DI
DI → DO
SCK → SCK

Для начала стоит проверить, что Arduino правильно обрабатывает поступающие команды.

2 Загрузи код в Arduino Uno.

1 Встроенная функция `readStringUntil` будет читать все данные, пришедшие в UART до специального символа — `'\n'` (перенос строки). Он появляется в паре с `'\r'` (возврат каретки) при передаче данных функцией `Serial.println()`. Эти символы удобно передавать для разделения команд, но не очень удобно обрабатывать. Удаляем их функцией `trim()`.

Код для Arduino Uno

```
#include <SPI.h>
#include <SD.h>

#define RELAY A4

File myFile;

void setup() {
  Serial.begin(115200);
  SD.begin(8);
  pinMode(RELAY, OUTPUT);
}

void loop() {

  String command = Serial.readStringUntil('\n');
  command.trim();

  if (command == "On") {
    digitalWrite(RELAY, HIGH);
  }
  if (command == "Off") {
    digitalWrite(RELAY, LOW);
  }
  if (command == "home.htm") {
    myFile = SD.open("home.htm");
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    myFile.close();
  }
}
```

2 Проверяем, подходит ли команда под одну из ожидаемых: `"On"`, `"Off"` или `"home.htm"`. Если подходит, выполняем действие.

3 Программа должна реагировать на три команды, попробуй отправить их в Serial Monitor:

- On — включить реле. Ты увидишь загоревшийся на реле светодиод.
- Off — выключить реле.
- home.htm — передать по UART содержимое файла «home.htm» в виде текста.

4 Теперь запрограммируй Wi-Fi модуль.

Код для Wi-Fi модуля

```
1  #include <ESP8266WiFi.h>
2  #include <WiFiClient.h>
3  #include <ESP8266WebServer.h>
4
5  String page = "";
6  ESP8266WebServer server(80);
7
8  void handleRoot() {
9    server.send(200, "text/html", page);
10 }
11
12 void handleOn() {
13   Serial.println("On");
14   server.send(200, "text/plain", "On");
15 }
16
17 void handleOff() {
18   Serial.println("Off");
19   server.send(200, "text/plain", "Off");
20 }
21 void setup(void) {
22   Serial.begin(115200);
23   WiFi.begin("имя_твоего_Wi-Fi", "пароль_твоего_Wi-Fi");
24
25   while (WiFi.status() != WL_CONNECTED) {
26     delay(500);
27   }
28   Serial.println();
29   Serial.println(WiFi.localIP());
30
31   server.on("/home.htm", handleRoot);
32   server.on("/turnOn", handleOn);
33   server.on("/turnOff", handleOff);
34
35   server.begin();
36   Serial.println("home.htm");
37 }
38
39 void loop() {
40   if (Serial.available()) {
41     page += (char)Serial.read();
42   }
43   server.handleClient();
44 }
```

1 Заведём функции `handleOn()` и `handleOff()`. Они будут обрабатывать нажатия кнопок включения и выключения света в браузере.

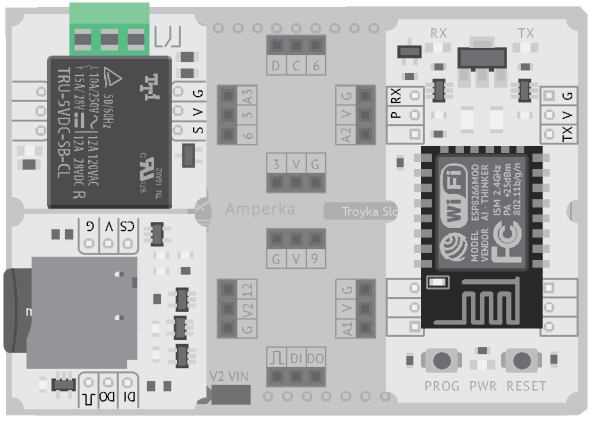
2 Если на сервер приходит запрос на включение или выключение реле, вызываем соответствующую функцию.

5 Проверь, как работает сервер. Открой Serial Monitor, дождись строки с IP-адресом модуля. Скопируй его в адресную строку браузера и допиши в конце «/turnOn».

192.168.88.192/turnOn

6 В Serial Monitor появится строка «On». Это значит, что Wi-Fi модуль отправил команду на включение реле. Попробуй команду «turnOff» — появится строка «Off».

7 Теперь собери устройство целиком!



Wi-Fi модуль
TX → P0
RX → P1

DO → DI
DI → DO
SCK → SCK

SD картридер
CS → P8

Мини-реле → A4

8 Подключи сервер к питанию. Набери в браузере IP-адрес сервера и добавь к нему «/home.htm». Ты увидишь интерфейс панели управления реле.

192.168.88.192/home.htm

9 Переключай реле кнопками «Включить» и «Выключить». При переключении реле должно издавать щелчки. Это звук работы металлической пластины внутри чёрного корпуса, которая замыкает то одну, то другую пару контактов. Ура, теперь ты можешь собрать умный светильник!

№ 7

TELEGRAM BOT

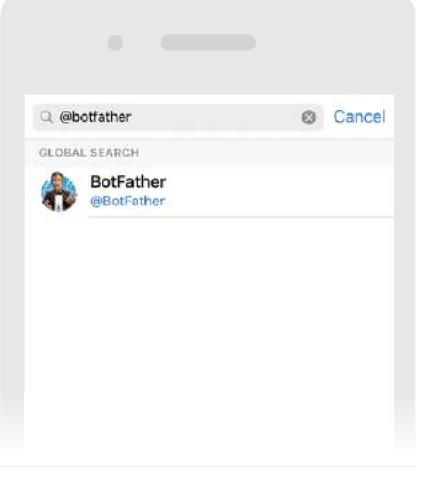
Будем управлять своими устройствами прямо из мессенджера Telegram. Для этого заведём собственного бота и научим Wi-Fi модуль с ним работать.

Если ты ещё не используешь Telegram, зайди на сайт telegram.org, установи приложение на свой смартфон и создай аккаунт.

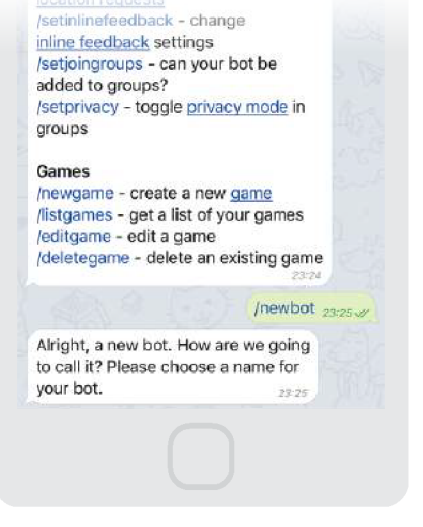
Telegram позволяет создавать собственных чат-ботов. Чат-бот — это робот, умеющий вести переписку. Робот умеет создавать специальные кнопки управления в чате и реагировать на их нажатие. Такие кнопки называются клавиатурами. Можно, например, сделать бота, управляющего светом по команде с клавиатуры.

Telegram работает только по протоколу HTTPS.

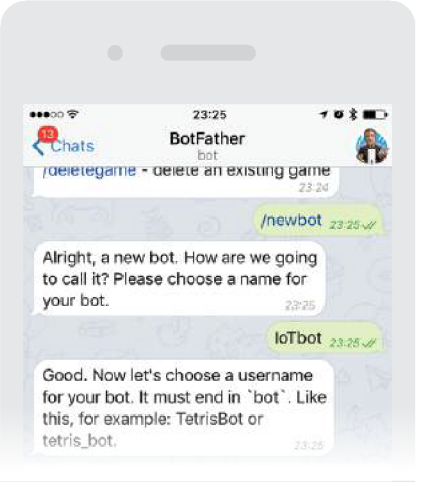
1 Открой мессенджер. В поиске найди главного бота — «@botfather». Заведи с ним диалог и нажми кнопку «START».



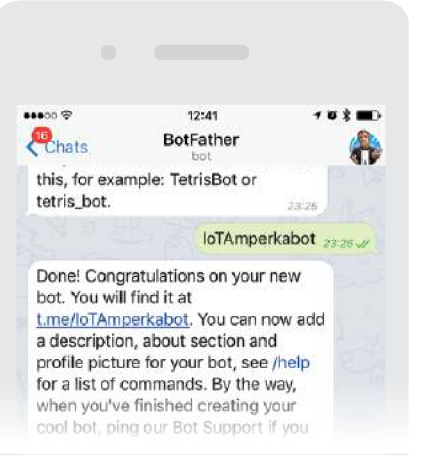
2 Выбери пункт «/newbot».



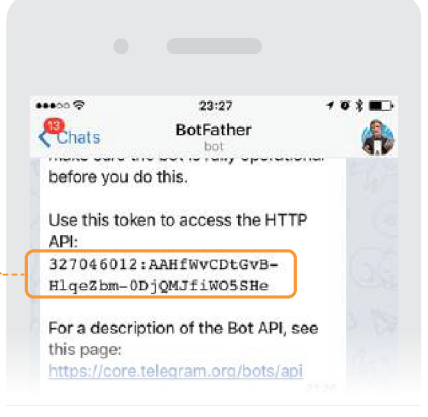
3 Придумай имя своему боту.



4 Теперь придумай уникальный логин бота. Он обязательно должен заканчиваться на «bot».



5 Сделано! Теперь скопируй специальный токен, который тебе выдал @BotFather. Он потребуется для авторизации. *Токен* — это уникальный ключ для доступа к сервису. Его должен знать только ты — не сообщай его никому!



Мощности Arduino Uno не хватает для работы с протоколом HTTPS. Поэтому нужно снова использовать Wi-Fi модуль, прошитый специально для твоих задач. Схема остаётся прежней — USB-UART из главы «Важное про Serial».

6 Запрограммируй Wi-Fi модуль (схема сборки стр. 57).

Код для Wi-Fi модуля

```
1   #include <ESP8266WiFi.h>
2   #include <WiFiClientSecure.h>
3   #include <UniversalTelegramBot.h>
4
5   char ssid[] = "имя_твоего wi-fi";
6   char password[] = "пароль_wi-fi";
7
8   #define BOTtoken "твой_токен"
9
10  WiFiClientSecure client;
11  UniversalTelegramBot bot(BOTtoken, client);
12
13  String keyboardJson = "[[\"/ledon\", \"/ledoff\"]]";
14  void setup() {
15    Serial.begin(9600);
16    WiFi.begin(ssid, password);
17    while (WiFi.status() != WL_CONNECTED) {
18      delay(500);
19    }
20  }
21
22  void loop() {
23    int numNewMessages = bot.getUpdates(
24              bot.last_message_received + 1);
25    handleNewMessages(numNewMessages);
26  }
27
28  void handleNewMessages(int numNewMessages) {
29    for (int i = 0; i < numNewMessages; i++) {
30      String chat_id = String(bot.messages[i].chat_id);
31      String text = bot.messages[i].text;
32      if (text == "/ledon") {
33        Serial.println("turnOn");
34        bot.sendMessage(chat_id, "Relay is ON", "");
35      }
36      if (text == "/ledoff") {
37        Serial.println("turnOff");
38        bot.sendMessage(chat_id, "Relay is OFF", "");
39      }
40      if (text == "/start") {
41        bot.sendMessageWithReplyKeyboard(chat_id,
42            "Choose from one of the following options",
43                    "", keyboardJson, true);
44      }
45    }
46  }
```

1 Заводим переменную для токена, который тебе выдал @BotFather.

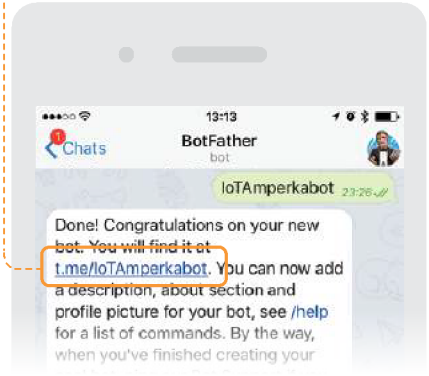
2 Задаём переменную для шифрованного соединения и инициализируем Telegram-бота с выделенным токеном.

3 Создаём клавиатуру для бота.

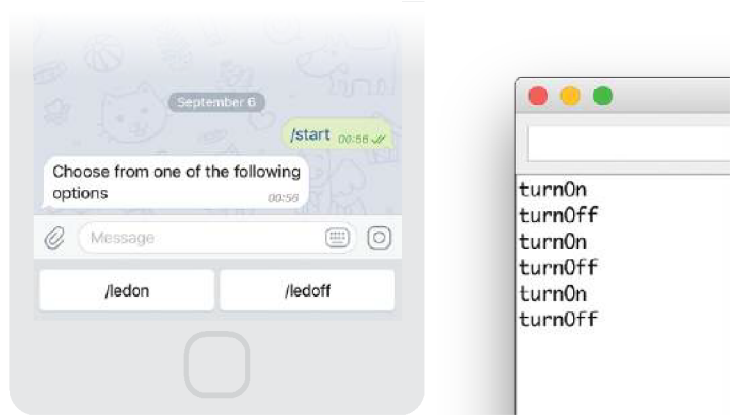
4 В цикле постоянно опрашиваем сервис на наличие новых сообщений для бота. В случае получения сообщения запускаем функцию-обработчик.

5 Функция-обработчик работает с сообщениями, которые поступают от Telegram, и отправляет сообщения о статусе выполнения команд.

7 Кликни на ссылку, которую тебе дал @BotFather, чтобы начать разговор с ботом.



8 Добавь своего бота в контакты и нажми кнопку «START». Ты увидишь клавиатуру с надписями «/ledon» и «/ledoff». Открой Serial Monitor и понажимай на эти кнопки. В Serial Monitor будут выводиться соответствующие сообщения.



Это и есть команды, которые будет слушать Arduino. Теперь научи Arduino их обрабатывать!

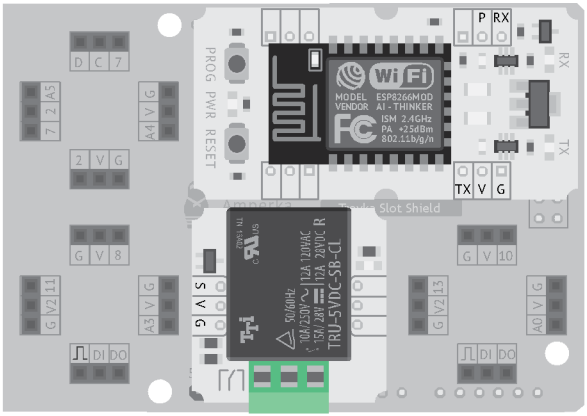
6 Проверяем каждое сообщение, пришедшее с сервиса, и запоминаем ID чата, из которого оно пришло.

7 Записываем в переменную типа `String` текст пришедшего сообщения.

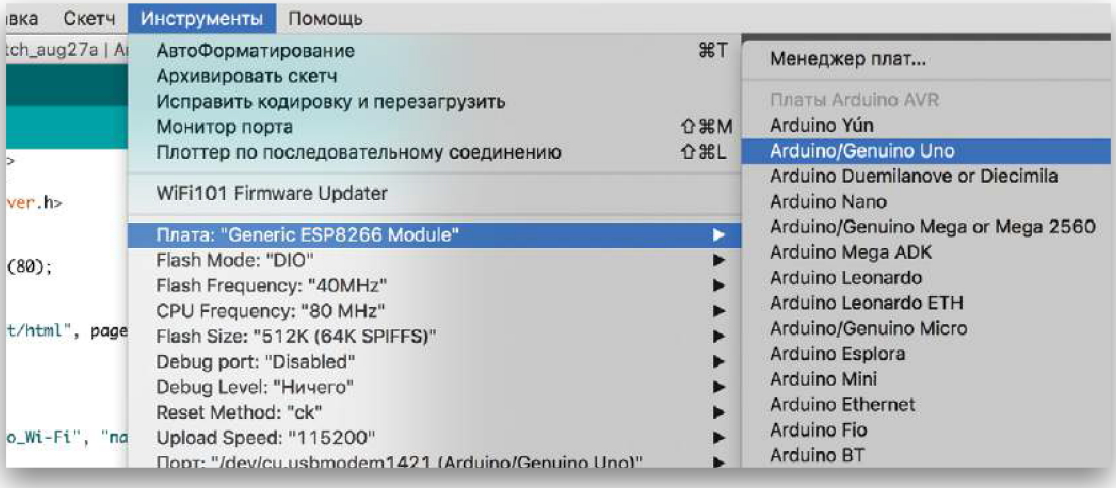
8 Если пришло сообщение с текстом `"/ledon"` — передаём в Arduino команду `"turnOn"`, а в ответном сообщении в Telegram сообщаем отправителю об успешном выполнении команды. Аналогичным образом обрабатываем команду `"/ledoff"`.

9 Если пришло сообщение с текстом `"/start"` — передаём отправителю клавиатуру с командами.

9 Отключи своё устройство от компьютера и пересобери его, добавив к Arduino Slot Shield и модуль мини-реле.



10 Подключи Arduino к компьютеру (не забудь сменить плату в Arduino IDE!) и загрузи в неё код:



```
#include <SoftwareSerial.h>

#define REL_PIN 12

SoftwareSerial esp(4, 5);

void setup() {
  esp.begin(9600);
  pinMode(REL_PIN, OUTPUT);
}

void loop() {
  String command = esp.readStringUntil('\n');
  command.trim();

  if (command == "turnOn") {
    digitalWrite(REL_PIN, HIGH);
  }

  if (command == "turnOff") {
    digitalWrite(REL_PIN, LOW);
  }
}
```

Код для Arduino Uno

1 «Слушаем» программный UART. Если по нему приходит команда `turnOn` — включаем реле. Если `turnOff` — выключаем реле.

11 Нажимай на кнопки в интерфейсе, смотри, как включается и выключается светодиод на реле.

ВНИМАНИЕ!

Для шифрования данных требуется много вычислительных ресурсов, поэтому задержка отклика устройства на сообщение в Telegram заметна глазу.

№ 8 BLYNK

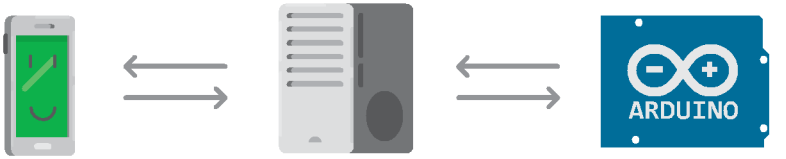
Познакомимся с сервисом Blynk и поуправляем RGB-светодиодом со смартфона.

Blynk — сервис для программирования приборов и удобного доступа к ним с панели управления на экране смартфона. Встроенный конструктор кода для различных задач и наглядный интерфейс панели управления сильно облегчают создание IoT-приборов. Сервис совместим с кучей устройств — от Wi-Fi Troyka-модуля до последней версии компьютера Raspberry Pi.

Для начала работы с Blynk прибор нужно прошить кодом с твоей программой и клиентской частью Blynk.

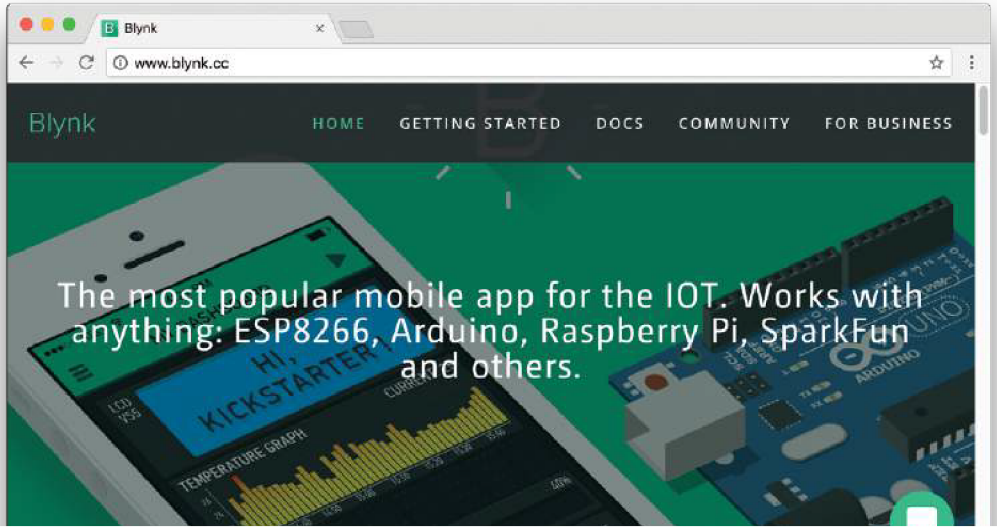
Затем понадобится смартфон с приложением Blynk — с него ты будешь контролировать свои умные устройства.

Оба устройства являются клиентами, связующим звеном между которыми выступает сервер Blynk.

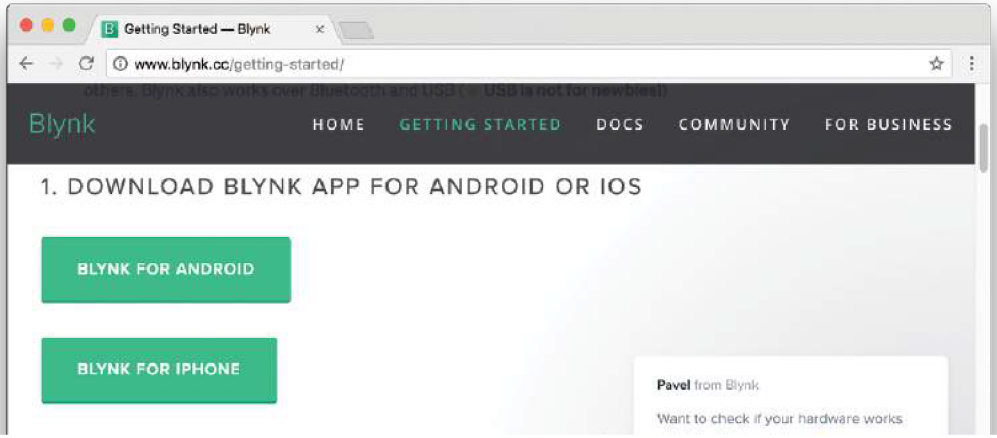


Смартфон с приложением Blynk — Сервер Blynk — IoT-прибор с прошивкой Blynk

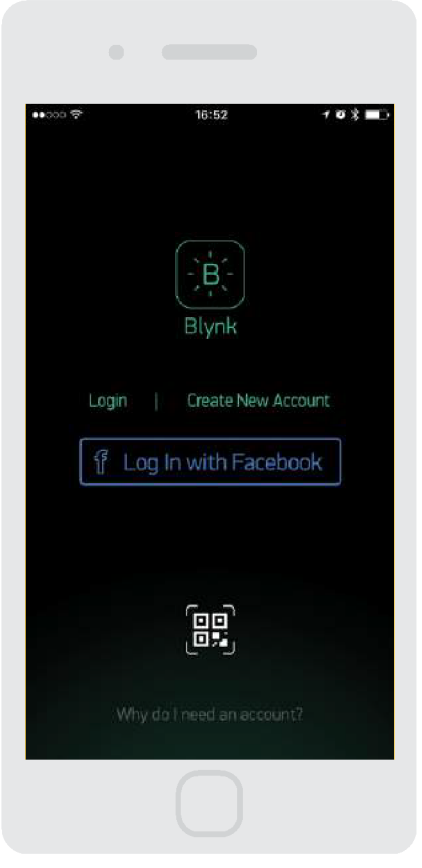
1 Зайди на blynk.cc и выбери пункт GETTING STARTED.



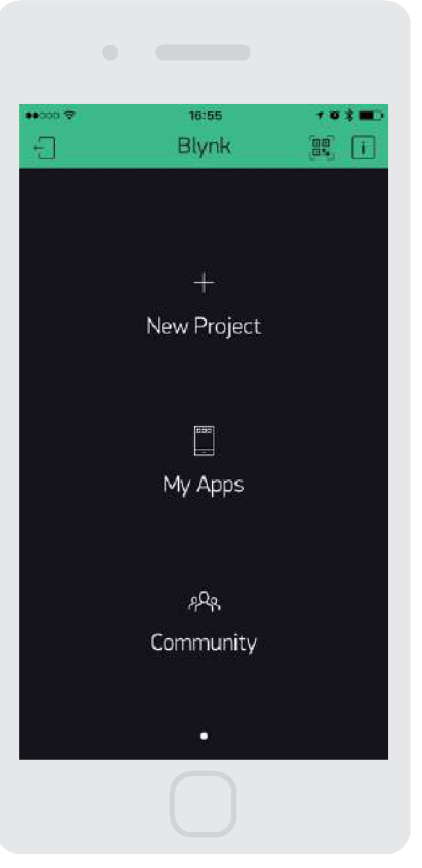
2 Установи на смартфон приложение Blynk.



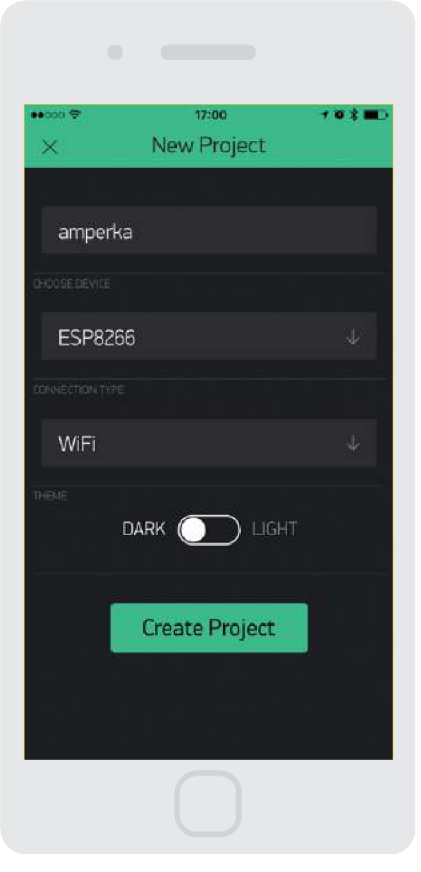
3 Открой приложение на смартфоне. Заведи учётную запись.



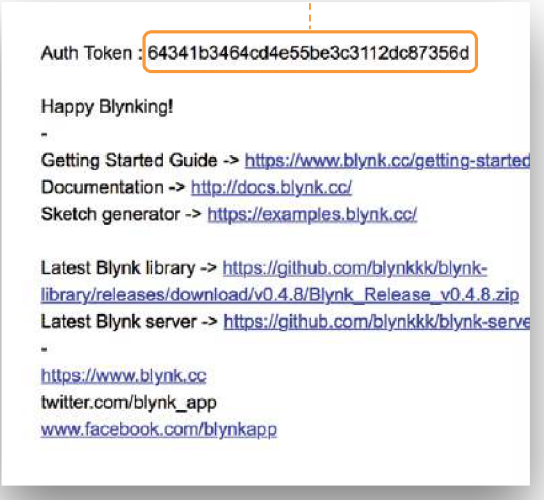
4 Нажми кнопку «New Project».



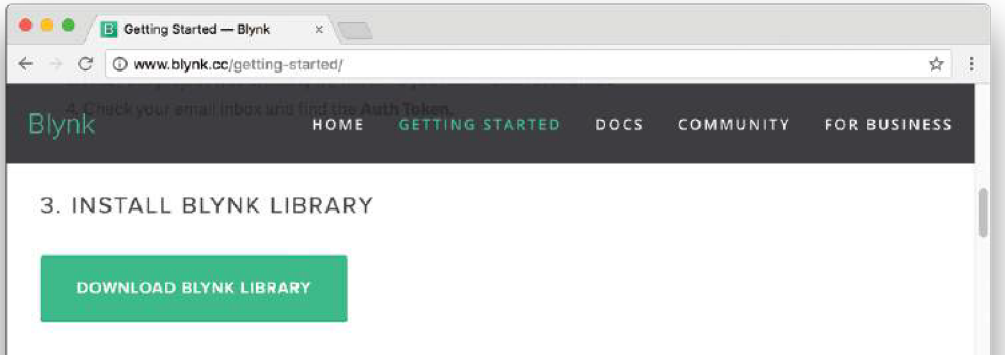
5 Введи название проекта и выбери плату, с которой будешь работать: ESP8266. Тип соединения — Wi-Fi.



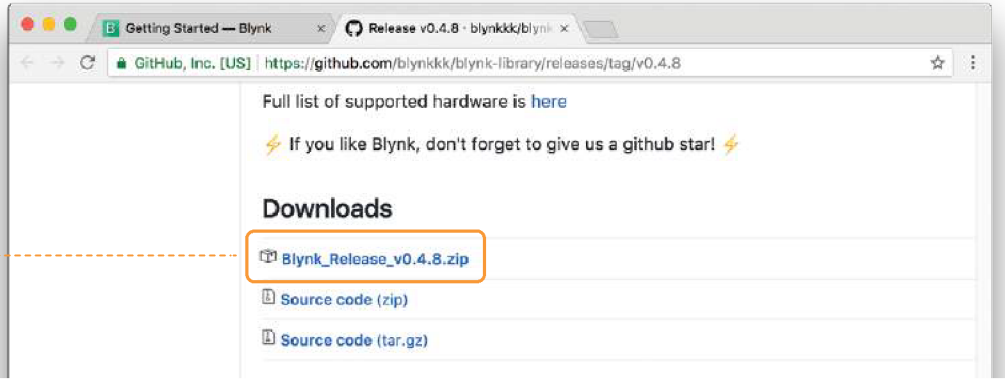
6 Как только ты создашь проект, на указанную при регистрации почту придёт сообщение с уникальным токеном идентификации. Он потребуется в коде программы.



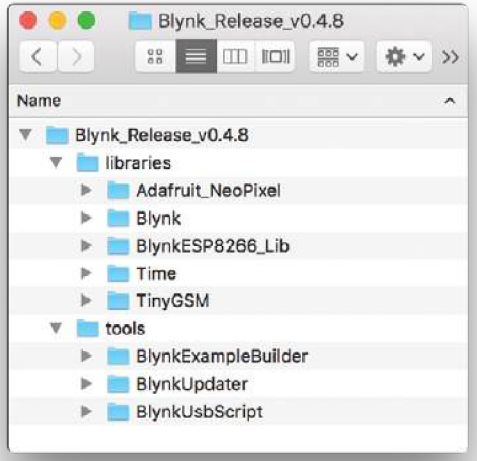
7 Теперь нужно установить библиотеки в Arduino IDE. Их нельзя автоматически установить из архива, поэтому придётся сделать это вручную. Ниже на странице «GETTING STARTED» нажми на кнопку «Download Blynk Library».



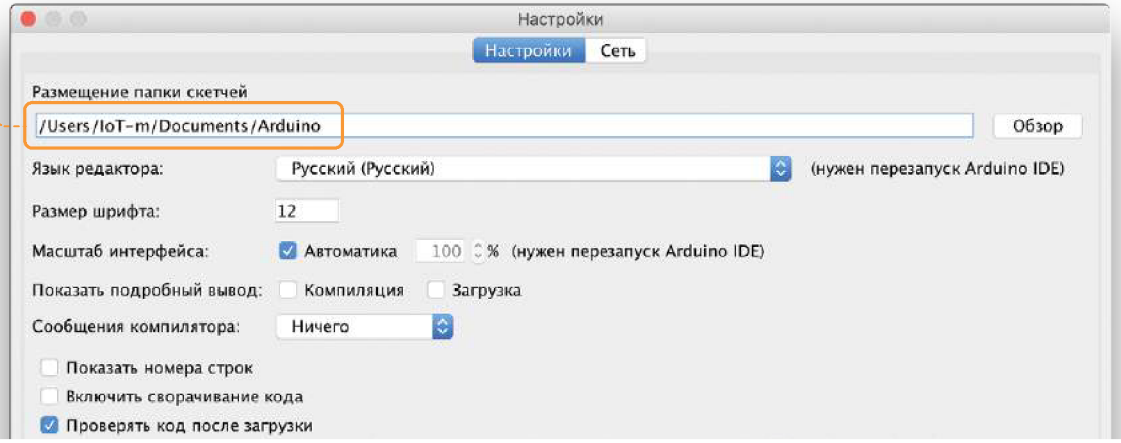
8 По ссылке в самом низу страницы находятся последние версии библиотек в .zip файле. Скачай этот архив.



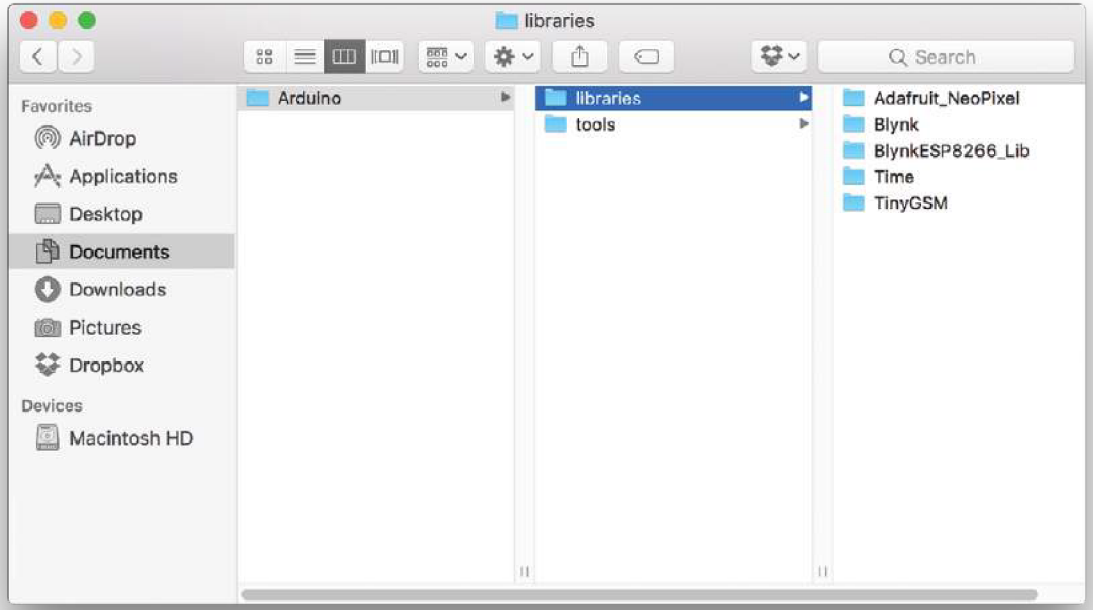
9 Разархивируй его. Ты увидишь, что архив содержит несколько папок.



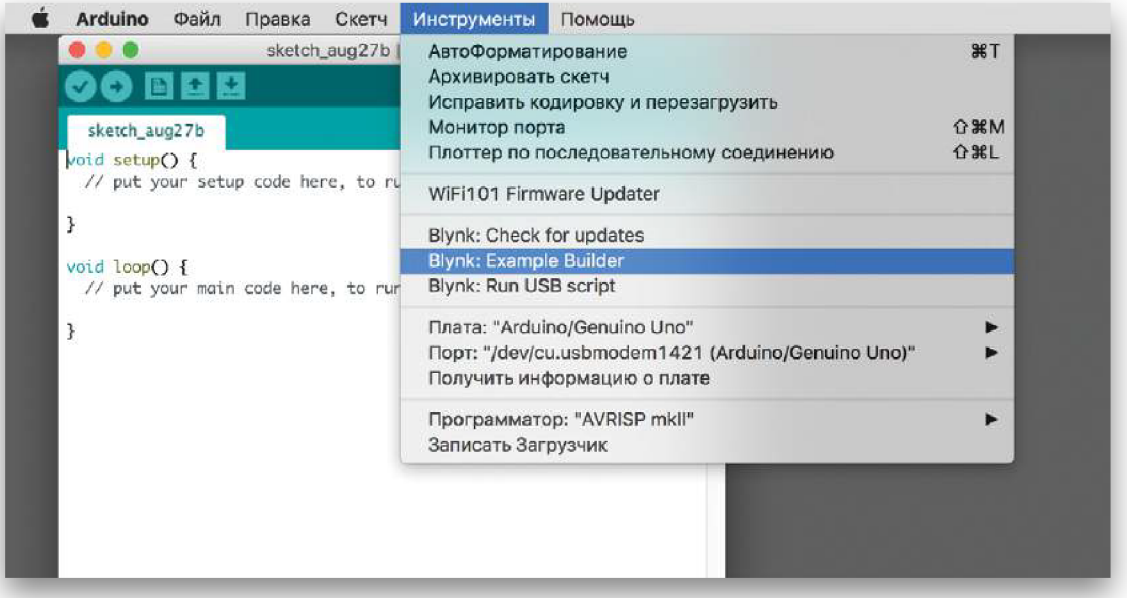
10 Найди свою папку со скетчами. Чтобы узнать её расположение в Windows, перейди в меню *Файл* → *Настройки*. В macOS – *Arduino* → *Настройки*.



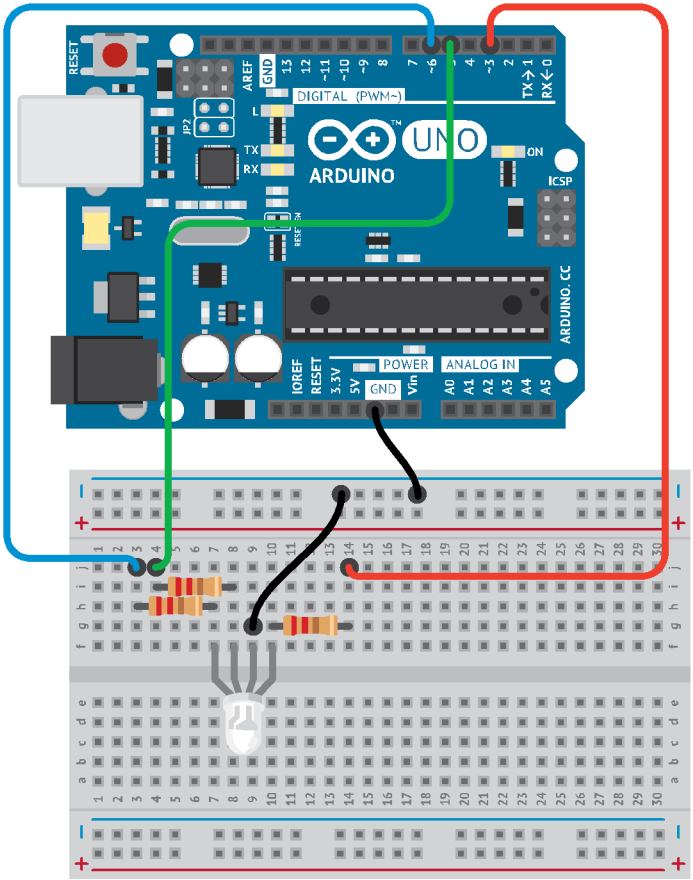
11 Скопируй содержимое папок из архива в папку со скетчами. Проследи, чтобы всё содержимое папки «libraries» было перенесено в «libraries», а «tools» — в «tools»:



12 Теперь Arduino IDE настроена и готова для работы с Blynk! Перезапусти Arduino IDE, чтобы среда подключила новые библиотеки и инструменты. Ты увидишь новые пункты в меню «Инструменты».



13 Теперь собери схему с RGB-светодиодом. Нужно проверить, как Arduino будет обрабатывать приходящие в Serial команды.

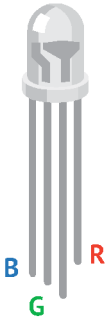


1 Если в Serial пришла непустая строка команды, пытаемся её обработать.

Резисторы — 220 Ом

RGB светодиод — это три светодиода с общим катодом.

Аноды светодиода — в пины 3(R), 5(G), 6(B)



14 Загрузи код в Arduino UNO:

Код для Arduino Uno

```
int ledR = 3;
int ledG = 5;
int ledB = 6;

void setup() {
  Serial.begin(115200);
}

void loop() {

  String command = Serial.readStringUntil('\n');

  if (command.length() > 0) {
    int led;
    if (command[0] == 'r') {
      led = ledR;
    }
    if (command[0] == 'g') {
      led = ledG;
    }
    if (command[0] == 'b') {
      led = ledB;
    }
    String brightness = command.substring(1);
    analogWrite(led, brightness.toInt());
  }
}
```

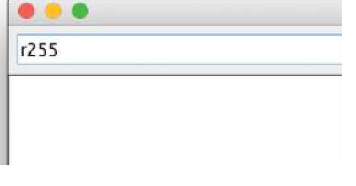
2 В переменную `led` будем записывать пин, для которого пришла команда. Сначала посмотрим на первый (нулевой) символ команды. Если первый символ строки `'r'`, значит эта команда для красного цвета. Таким же способом учим Arduino двум другим цветам.

3 В переменную `brightness` записываем строку команды `command` за исключением первого символа. В ней мы ожидаем увидеть числовое значение яркости. Это значение должно находиться в диапазоне 0...255 (255 — максимальная яркость).

4 Функцией `analogWrite()` включаем цвет с заданной яркостью `brightness`. Мы уже пользовались ей в эксперименте № 11 «Конспекта хакера». Функция `.toInt()` переведёт строку `brightness` к числовому виду.

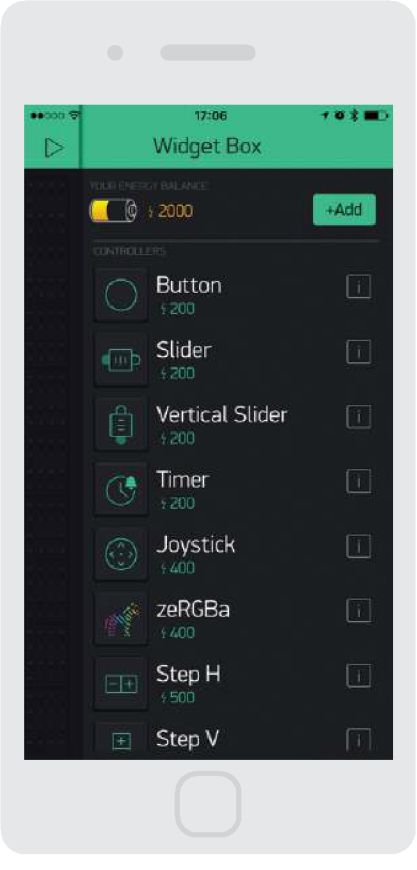
15 Теперь можно проверить команды. Открой Serial Monitor и выбери скорость соединения 115 200 бод. Отправь команду «r255». Светодиод должен загореться красным. Отправь «g255». Светодиод загорится жёлто-зелёным. И по команде «b255» светодиод должен стать белым. Можно поиграть с цифровыми значениями яркости, получая разные цвета.

Иногда при максимальных значениях всех трёх цветов светодиод светит не чистым белым светом. Это связано с разным уровнем падения напряжения на каждом из трёх светодиодов внутри общего корпуса. Можно попробовать добиться белого цвета, экспериментируя с резисторами разных номиналов.



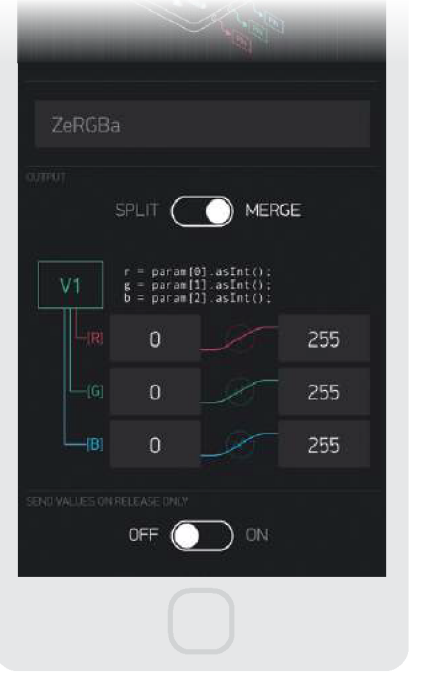
16 Вернись к своему смартфону. Открой приложение Blynk и проект, который ты создал.

17 Проведи пальцем по экрану влево. Ты увидишь панель виджетов.



18 Выбери виджет «zeRGBa». Он появится на рабочем поле проекта.

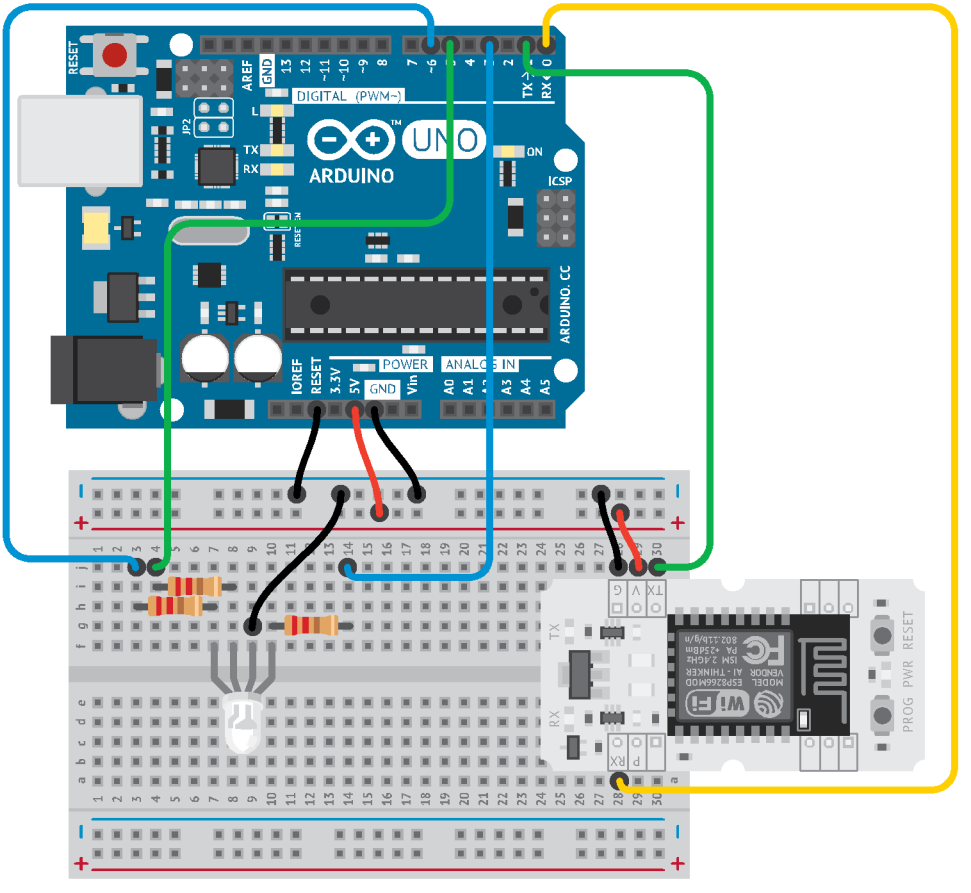
19 Щёлкни на виджете, чтобы увидеть его настройки. OUTPUT переведи в MERGE, установи диапазон значений от 0 до 255, а SEND ON RELEASE переведи в OFF.



Для доступа к физическим пинам устройства через элементы управления в приложении Blynk используются «виртуальные пины».

В коде, которым прошивается Arduino, можно назначить номера виртуальных пинов для любых доступных пинов устройства. А в самом приложении при выборе инструмента управления или наблюдения нужно указать номер виртуального пина. Виртуальный пин можно указать и для аппаратного UART — для этого в коде будет использоваться функция-обработчик.

20 Подключи модуль к Breadboard'у. Ты можешь перепрограммировать его, не разбирая схемы, — соедини модуль с Arduino в режиме USB-UART.



21 Загрузи на Wi-Fi модуль управляющую прошивку Blynk:

Код для Wi-Fi модуля

```
1   #include <ESP8266WiFi.h>
2   #include <BlynkSimpleEsp8266.h>
3
4   char auth[] = "токен_blynk";
5   char ssid[] = "имя_твоего wi-fi";
6   char pass[] = "пароль_wi-fi";
7
8   void setup() {
9     Serial.begin(115200);
10    Blynk.begin(auth, ssid, pass);
11  }
12
13  void loop() {
14    Blynk.run();
15  }
16
17  BLYNK_WRITE(V1) {
18    int redValue = param[0].asInt();
19    int greenValue = param[1].asInt();
20    int blueValue = param[2].asInt();
21
22    Serial.print("r");
23    Serial.println(redValue);
24    Serial.print("g");
25    Serial.println(greenValue);
26    Serial.print("b");
27    Serial.println(blueValue);
28  }
```

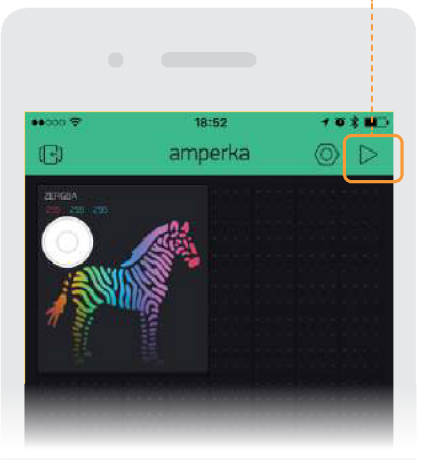
1 Запускаем Blynk с нашими настройками, указываем токен из письма.

2 Циклично запрашиваем данные у Blynk. Если от сервиса пришли новые данные для виртуального пина `V1`, вызываем функцию-обработчик `BLYNK_WRITE`.

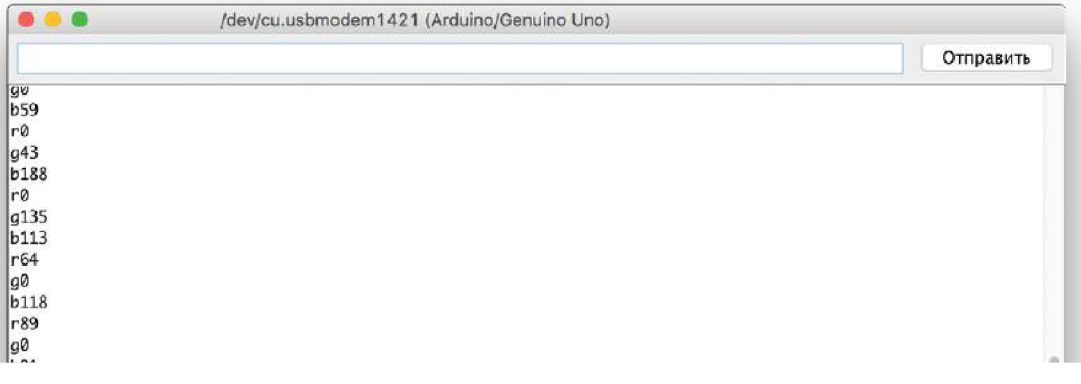
3 Данные для пинов содержатся в переменной `param`. Blynk отправляет сразу 3 значения, к каждому из них мы можем обратиться по индексу. Функция `asInt()` преобразует входные данные в числовое значение.

4 Записываем в Serial три строки. По одной на каждый цвет: первый символ строки указывает цвет, а за ним следует число от 0 до 255.

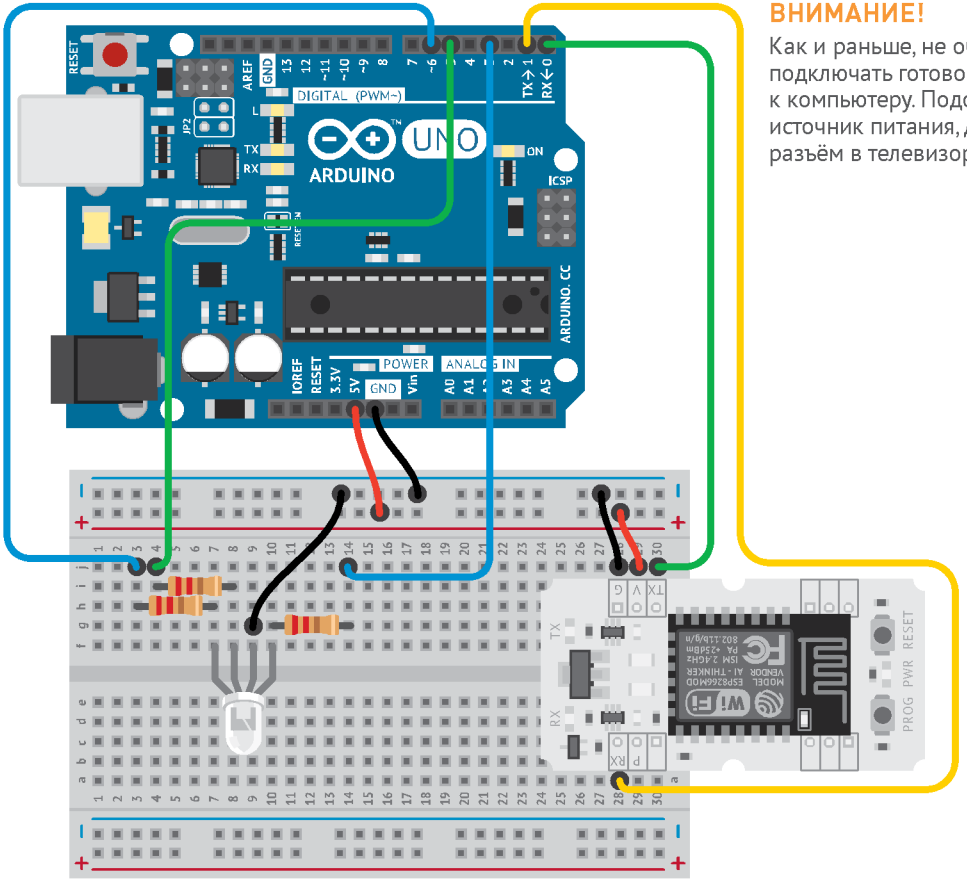
22 Проверь работу программы. Сразу после загрузки кода на Wi-Fi модуль открой проект с zeRGBa на смартфоне. Нажми на кнопку с треугольником в правом верхнем углу экрана — это запустит работу проекта.



23 Открой Serial Monitor. Убедись, что скорость выставлена на 115 200 бод, и начни водить пальцем по зебре. Ты увидишь поток данных, которые пойдут через Serial. Данные должны приходить именно в том формате, для которого ты подготовил программу на Arduino, — убедись в этом перед финальным запуском.



24 Теперь собери финальное устройство. Подключи правильно RX и TX Wi-Fi модуля и сними перемычку с RESET и GND на Arduino.



ВНИМАНИЕ!

Как и раньше, не обязательно подключать готовое устройство к компьютеру. Подойдёт любой источник питания, даже USB-разъём в телевизоре.

Ура, ты прошёл весь набор! Теперь ты можешь создавать серьёзные устройства. Используй изученные сервисы для своих крутых проектов. Полив растений, кормушка для домашних животных и птиц, система умного дома — управление устройствами теперь доступно с твоего смартфона в любое время даже на другом конце Земли!

ВОССТАНОВЛЕНИЕ AT-ПРОШИВКИ

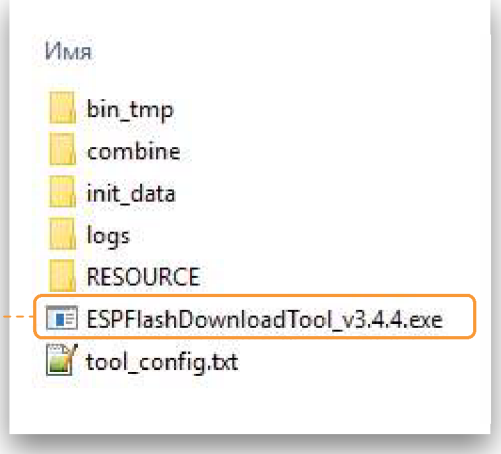
После прохождения экспериментов с перепрошивкой тебе наверняка потребуется восстановить стандартный функционал модуля. На iot-m.amperka.ru → *«Восстановление прошивки Wi-Fi модуля»* ты найдёшь архив, содержащий в себе все необходимые файлы. Подключи Wi-Fi модуль к Arduino так же, как и для перепрошивки.

ИНСТРУКЦИЯ ДЛЯ WINDOWS

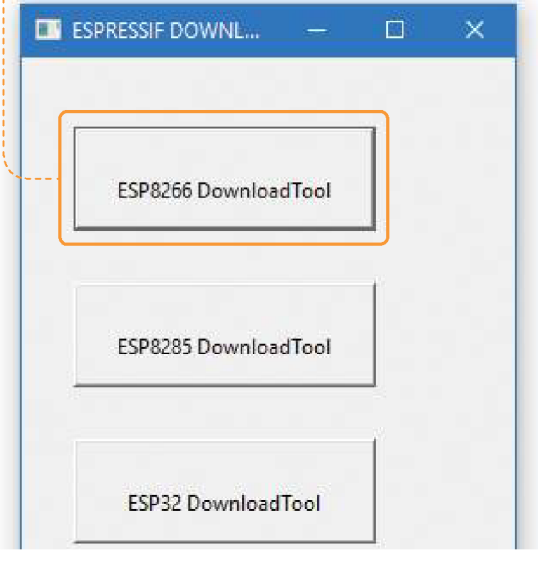
1. Скачай архив ESPTOOL_WIN.

2. Распакуй архив с программой для установки. При этом конечный путь к программе и файлу прошивки не должен содержать кириллических символов.

3. Запусти файл ESPFlashDownloadTool_v3.4.4



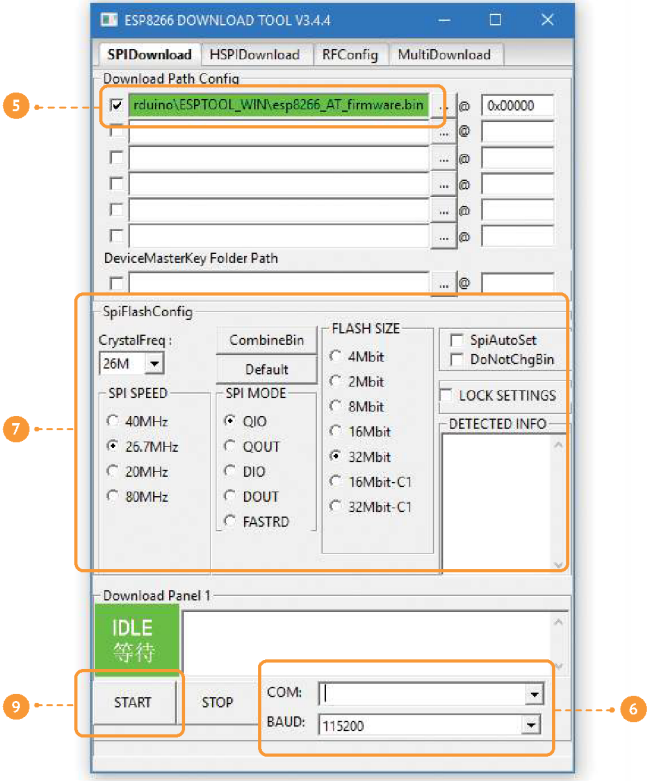
4. В появившемся окне выбери «ESP8266 DownloadTool».



5 Укажи путь к файлу AT-прошивки «esp8266_AT_firmware.bin», отметь выбранный файл галочкой в чекбоксе напротив и укажи адрес 0x00, по которому будет записан файл в память модуля.

6 Выбери COM порт, к которому подключён твой модуль, и установи Baud rate 115 200.

7 Настрой оставшиеся поля и чекбоксы:



8 Переведи модуль в режим программирования.

9 Теперь, когда модуль готов к прошивке, нажми кнопку «START» и дождись окончания процесса. Ура, модуль снова готов к работе!

ИНСТРУКЦИЯ ДЛЯ MACOS

Сначала необходимо установить инструменты для перепрошивки микроконтроллера ESP8266.

1. Скачай архив ESPTOOL_MAC.

2. Распакуй архив с программой для установки. При этом конечный путь к программе и файлу прошивки не должен содержать кириллических символов.

3. Открой терминал. Ты найдёшь его в *Finder* → *Программы* → *Служебные программы (Applications* → *Utilities)* → *Terminal*.

4. В терминале набери cd, нажми пробел и перенеси в окно терминала иконку папки с содержимым архива. Нажми Enter.

5 Набери в терминале `sudo python setup.py install`.
Нажми Enter.

```
esptool — -bash — 80×24
Last login: Tue Sep  5 01:52:22 on ttys000
Evgenys-MacBook-Pro:~ Evgeny$ cd /Users/Evgeny/Downloads/esptool
Evgenys-MacBook-Pro:esptool Evgeny$ sudo python setup.py install
```

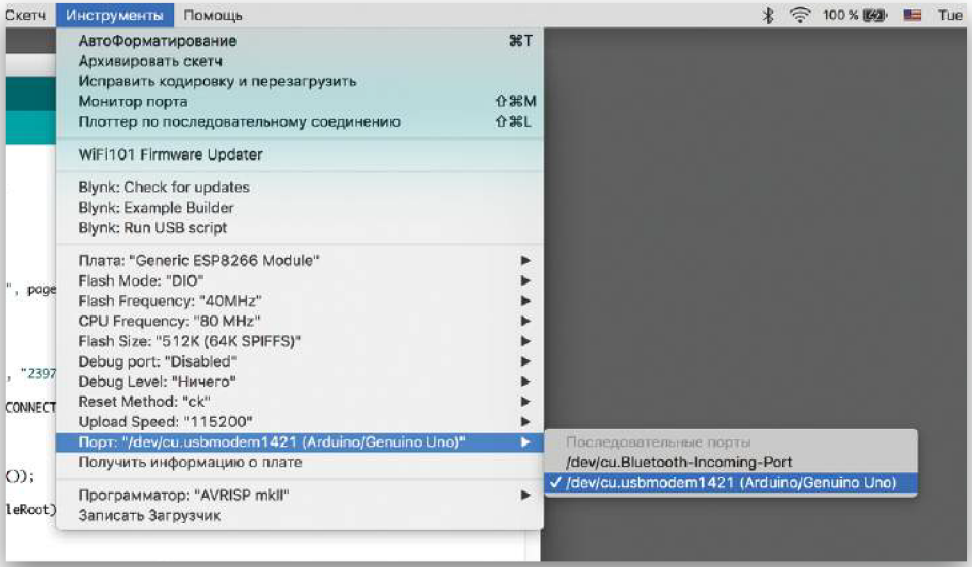
Терминал запросит пароль — введи его и нажми Enter.

```
esptool — sudo — 80×24
Last login: Tue Sep  5 01:52:22 on ttys000
Evgenys-MacBook-Pro:~ Evgeny$ cd /Users/Evgeny/Downloads/esptool
Evgenys-MacBook-Pro:esptool Evgeny$ sudo python setup.py install
Password:
```

```
esptool — -bash — 80×24

Installed /Library/Python/2.7/site-packages/esptool-2.0.1-py2.7.egg
Processing dependencies for esptool==2.0.1
Searching for ecdsa==0.13
Best match: ecdsa 0.13
Processing ecdsa-0.13-py2.7.egg
ecdsa 0.13 is already the active version in easy-install.pth

Using /Library/Python/2.7/site-packages/ecdsa-0.13-py2.7.egg
Searching for pyaes==1.6.0
Best match: pyaes 1.6.0
Processing pyaes-1.6.0-py2.7.egg
pyaes 1.6.0 is already the active version in easy-install.pth

Using /Library/Python/2.7/site-packages/pyaes-1.6.0-py2.7.egg
Searching for pyserial==3.4
```

Инструменты установлены. Осталось перепрошить модуль.

1. Узнай в Arduino IDE, к какому порту подключена твоя плата.



2 Набери в терминале команду для перепрошивки:

```
python esptool.py --port имя_порта write_flash 0x00000 esp8266_AT_firmware.bin
```

Вместо «имя_порта» подставь порт из Arduino IDE (не забывая про прямой слэш). Введи модуль в режим перепрошивки. Нажми Enter в терминале.

esptool — -bash — 80×24

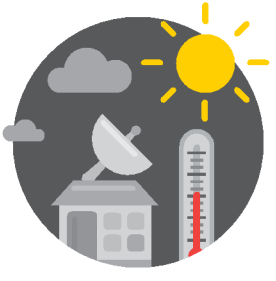
```
pyserial 3.4 is already the active version in easy-install.pth
Installing miniterm.py script to /usr/local/bin

Using /Library/Python/2.7/site-packages/pyserial-3.4-py2.7.egg
Finished processing dependencies for esptool==2.0.1
Evgenys-MacBook-Pro:esptool Evgeny$ python esptool.py --port /dev/cu.usbmodem142
1 write_flash 0x00000 esp8266_AT_firmware.bin
esptool.py v2.0.1
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 4190208 bytes to 307194...
Wrote 4190208 bytes (307194 compressed) at 0x00000000 in 26.5 seconds (effective
 1263.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting...
Evgenys-MacBook-Pro:esptool Evgeny$
```

3 После успешной прошивки модуль перезагрузится. Ты сможешь проверить его через Arduino IDE стандартными AT-командами, как в эксперименте «На старт, внимание, Wi-Fi!».

ИДЕИ ПРОЕКТОВ

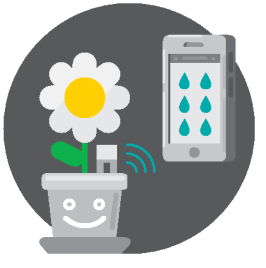
Прошёл набор «IoT»? Молодец!

Вот тебе идеи для приборов, которые пригодятся в быту. Применяй знания, полученные из набора, комбинируй различные модули, придумывай и твори с Амперкой!



МЕТЕОСТАНЦИЯ

Создай свою собственную метеостанцию, используя Troyka-модули барометра и цифрового датчика температуры и влажности. Добавь аккумулятор Power Shield — и получи автономное устройство, которое можно оставить на длительное время за окном. Не забудь поместить метеостанцию в герметичный корпус, чтобы она работала даже в очень плохую погоду.

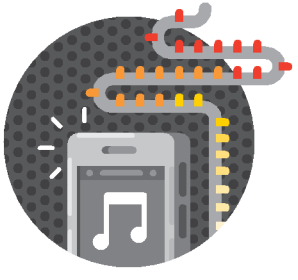


АВТОПОЛИВ

При помощи датчиков влажности почвы ты сможешь всегда быть уверенным, что твои редкие растения не засохнут от недостатка воды. А даты поливов ты сможешь регистрировать с помощью сервиса IFTTT.

ПОЖАРНАЯ СИГНАЛИЗАЦИЯ

Датчик горючих и угарного газов MQ-9 даст уверенность в безопасности твоего жилья. В случае обнаружения в воздухе присутствия опасных газов он пришлёт сообщение тебе прямо в Telegram!



СВЕТОМУЗЫКА

Цветная адресуемая светодиодная лента WS2811 позволит создать освещение в комнате, цвет которого ты сможешь контролировать со смартфона, или даже сделать его реагирующим на музыку, которая играет в комнате.

АНТИВОР

Датчик шума и ультразвуковой дальномер позволят построить систему сигнализации в комнате — ты всегда будешь знать, если в ней появятся непрошеные гости. А модуль MP3-плеера в комбинации с модулем аудиовыхода сможет твоим голосом вежливо попросить их покинуть чужую территорию.

АВТОНОМНЫЙ ГАДЖЕТ

GPRS Shield, совместимый с Arduino, позволит тебе создавать автономные приборы — ведь теперь они будут выходить в сеть не через Wi-Fi роутер, а напрямую с помощью сотовой сети. Не забудь приобрести SIM-карту с безлимитным трафиком!

ВСЕ ЭТИ ДАТЧИКИ И ПРИБОРЫ ТЫ СМОЖЕШЬ НАЙТИ НА AMPERKA.RU. ОСТАВАЙСЯ С НАМИ — ТЕБЯ ЖДЁТ ЕЩЁ БОЛЬШЕ КРУТЫХ ПРОЕКТОВ!

СОДЕРЖАНИЕ

Амперка

Мы в компании Амперка надеемся, что наш набор понравился.

Если у тебя есть вопросы, на них ответят на форуме: forum.amperka.ru

Обращайся к видеоканалу за порцией вдохновения: youtube.com/AmperkaRU

Ищи подробные руководства и инструкции на wiki.amperka.ru

Заходи в магазин amperka.ru за новыми модулями

Электронная версия книги: iot-m.amperka.ru

vk.com/amperkaru

facebook.com/amperka.ru

instagram.com/amperkaru

twitter.com/amperka

Дизайн: студия «Кластер»
clusterstudio.ru

Амперка