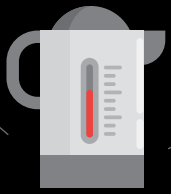
ИНТЕРНЕТ ВЕЩЕЙ

УПРАВЛЯЙ УСТРОЙСТВАМИ ЧЕРЕЗ ИНТЕРНЕТ

Амперка

Электронная версия книги:
iot.amperka.ru

СОДЕРЖАНИЕ



Включать приборы дистанционно

Управлять приборами со смартфона

Управлять светом

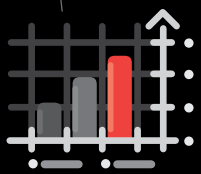
IOT, «INTERNET OF THINGS» — ИНТЕРНЕТ ВЕЩЕЙ.

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. И 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. Кроме того, управлять светом в квартире со смартфона — это очень весело!

Многие 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!

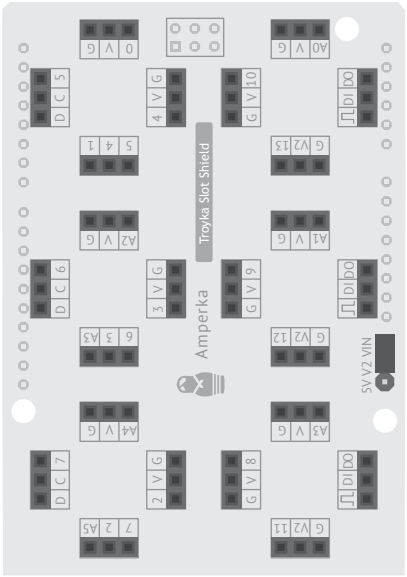
Собирать данные о погоде

Следить за объектами

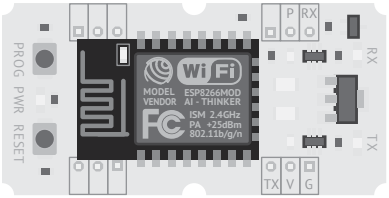
Измерять температуру

Собирать данные в единую базу

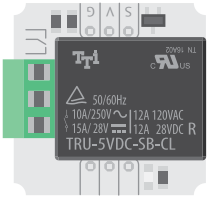
ЭЛЕМЕНТЫ В НАБОРЕ



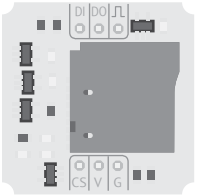
Troyka Slot Shield
Плата расширения для компактного подключения модулей



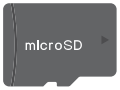
Wi-Fi модуль
Позволяет передавать данные по Wi-Fi-сети



Мини-реле
Управляет высоковольтными приборами



SD-картридер
Читает и записывает файлы на карту microSD



Карта microSD
Необходима для хранения файлов

ВНИМАНИЕ!

Если прошивка твоей платы ниже 1v91, обнови её. Как обновить прошивку — ищи на iot.amperka.ru → «Обновление прошивки».

Чтобы узнать версию прошивки твоей Iskra JS, подключи её к компьютеру и набери в консоли

```
reset();
```

Плата ответит тебе баннером, под которым будет указана версия прошивки.

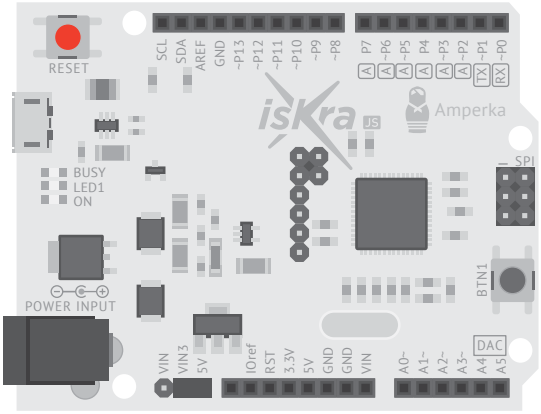
```
>reset();
=undefined

    ____          __
   / __/_____ / /__ _____ ____ _
  / /_/ ___// //_// ___// __ `/
 _/ / (__  )/ ,<  / /   / /_/ /
/___//____//_/|_|/_/    \__,_/
Based on Espruino 1v92.194
(c) 2016 G.Williams, Amperka LLC
Support the work of core developers:
http://espruino.com/Donate
>
```

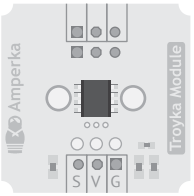
ЧТО ЕЩЁ ПОНАДОБИТСЯ

Для проектов IoT тебе потребуются платы и модули из базового набора «Йодо».

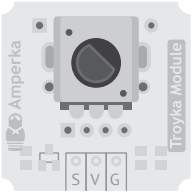
Кабель micro-USB
Соединяет Iskra JS с компьютером



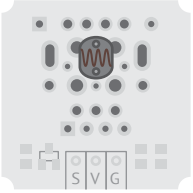
Iskra JS
Мозг твоего устройства



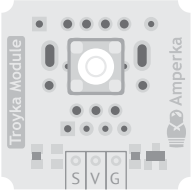
Термометр
Измеряет температуру воздуха



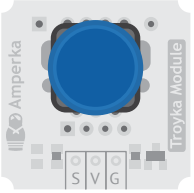
Потенциометр
Сообщает о повороте ручки



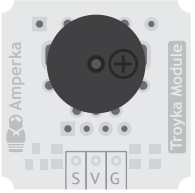
Датчик освещённости
Измеряет яркость света



Светодиод
Светит и мигает



Кнопка
Сообщает о нажатии



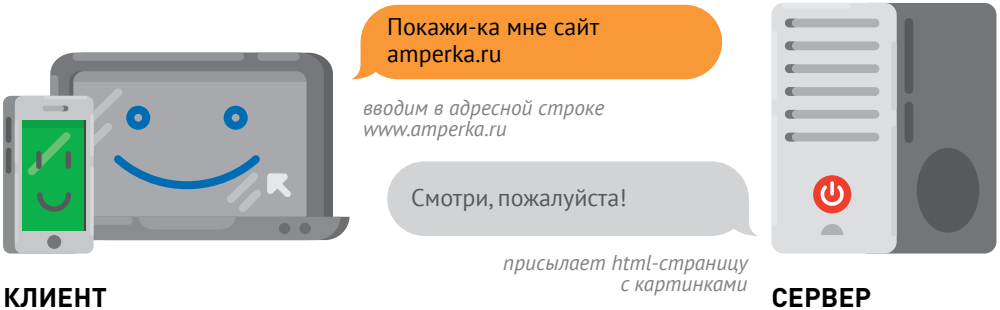
Зуммер
Пищит, издаёт звуки

Если ты уже используешь эти модули и платы в своих крутых проектах, а разбирать их не хочется, закажи недостающие на сайте iot.amperka.ru → «Необходимые модули из набора Йодо».

На сайте iot.amperka.ru → «Буклет Йодо» ты сможешь найти электронную версию буклета.

ИНТЕРФЕЙСЫ И ПРОТОКОЛЫ

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, называют *сервером*.



КЛИЕНТ
Делает запросы к серверу и получает от него ответ

СЕРВЕР
Принимает запросы от клиента и отдаёт ему ответ

Чтобы передать информацию в понятном друг для друга виде, устройства используют общепринятые *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. Для устройств очень важно передавать данные на одной скорости, иначе устройства не поймут сообщения друг друга.

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.

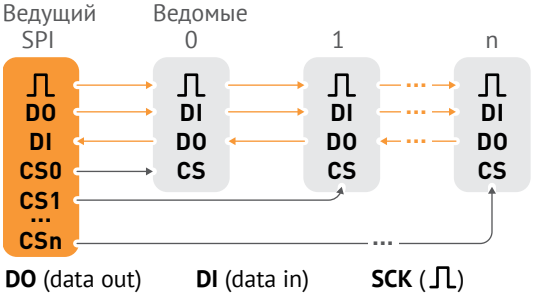
ИНТЕРФЕЙС UART

UART (Universal Asynchronous Receiver-Transmitter, Универсальный Асинхронный Приемопередатчик) использует для передачи данных два провода: по одному проводу в одну сторону, по другому — в другую. 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.

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

ИНТЕРФЕЙС SPI

Интерфейс SPI (Serial Peripheral Interface) позволяет соединить в сеть больше двух устройств. Одно из них становится ведущим (Master, мастером), а все 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 (⎍).



ПРОТОКОЛ HTTP

Протокол HTTP обеспечивает передачу html-страниц и медиафайлов. С ним работают все браузеры. В этом протоколе клиент и сервер общаются прерывно. В перерывах между запросами клиент и сервер отключаются друг от друга.

Передача данных по протоколу HTTP выглядит как простой текст из нескольких строк. Подробнее о протоколе читай в справочнике на странице 50.

ПРОТОКОЛ HTTPS

Протокол HTTPS — это тот же HTTP, но с шифрованием (S — security). Оно необходимо для защиты данных от перехвата злоумышленниками. Шифрование требует большого количества вычислительных ресурсов. Iskra JS — мощная платформа, её ресурсов достаточно для соединения по протоколу HTTPS.

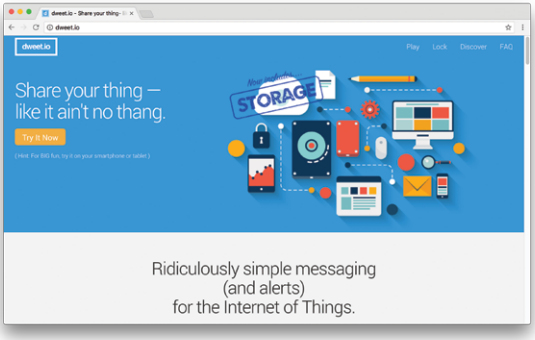
ПРОТОКОЛ WEBSOCKET

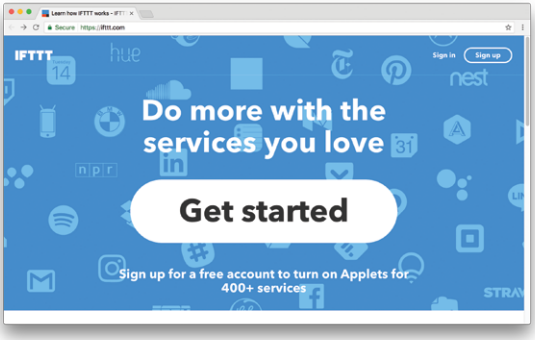
Протокол WebSocket — это непрерывный протокол. Соединение между сервером и клиентом постоянно поддерживается, и они могут в любой момент передавать данные, поэтому серверу не обязательно ждать запроса от клиента. Прежде чем установить такое соединение, клиент и сервер договариваются об этом по протоколу HTTP или HTTPS.

СЕРВИСЫ ДЛЯ ПРОЕКТОВ

Интернет вещей состоит из множества устройств, взаимодействующих между собой с помощью различных интерфейсов и протоколов. Они обмениваются информацией друг с другом и с людьми, и их число постоянно растёт.

Существует множество сервисов, упрощающих работу с устройствами через интернет. Некоторые из них мы будем использовать в наших экспериментах — давай познакомимся с ними поближе.





DWEET.IO

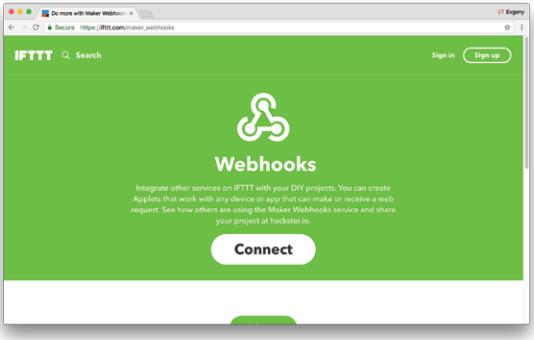
dweet.io — это сервис, позволяющий получать данные с устройств и выводить их в графическом виде. Можно установить датчик в своём аквариуме и наблюдать значение температуры воды из любой точки Земли. Каждое подключённое к сервису устройство имеет свой уникальный ключ, что позволяет не путать данные. Это почти как социальная сеть — только для приборов.

www.dweet.io

IFTTT

IFTTT (IF This Then That — если это, сделай то). Этот сервис позволяет подключать множество компонентов друг к другу по принципу «если произошло событие А, сделай действие Б». Сервисы Google, социальные сети, мессенджеры, записные книжки и многое другое можно связать между собой и автоматизировать рутинные действия. Такие действия с условиями называются *апплетами* (*applet, application* — приложение, *-let* — уменьшительный суффикс). Один из самых простых апплетов — отправить специально сформированный email в случае срабатывания условия. Мы будем пользоваться этой функцией в эксперименте № 6.

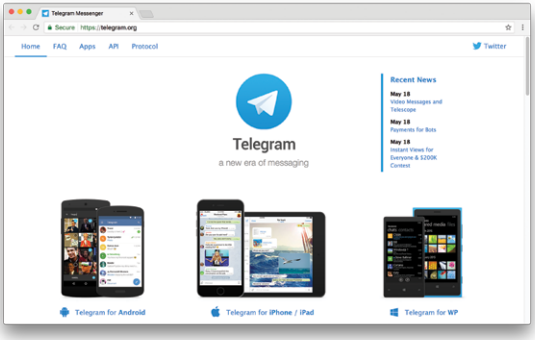
www.ifttt.com



WEBHOOKS

Для получения HTTP-команд с различных DIY-устройств у сервиса IFTTT есть специальный апплет — Webhooks. Этот апплет позволяет как принимать команды, так и отправлять их на устройства. Подробнее о нём ты узнаешь в эксперименте № 6.

www.ifttt.com/maker_webhooks



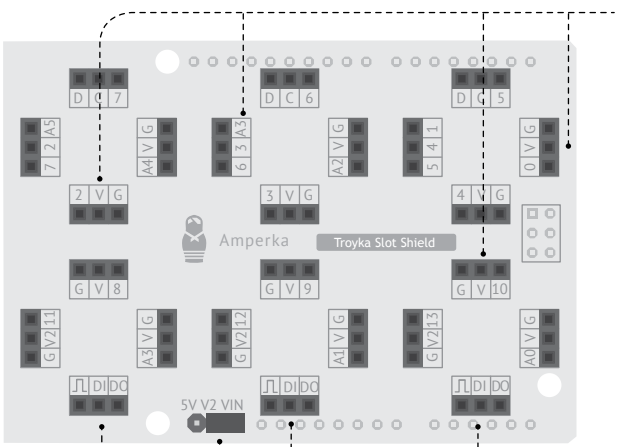
TELEGRAM

Telegram — продвинутый мессенджер с возможностью создавать и управлять ботами. Если ты ещё не используешь его — зайди на сайт telegram.org, установи приложение Telegram на свой смартфон и создай свой аккаунт, это пригодится тебе в эксперименте № 7.

www.telegram.org

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 троек контактов, в которые можно подключить один Troyka-модуль.



■ Тройка-контакты, соответствующие пинам на Iskra JS. Те, что отмечены «A», умеют работать с аналоговыми сигналами.

■ Пины для подключения устройств по интерфейсу SPI

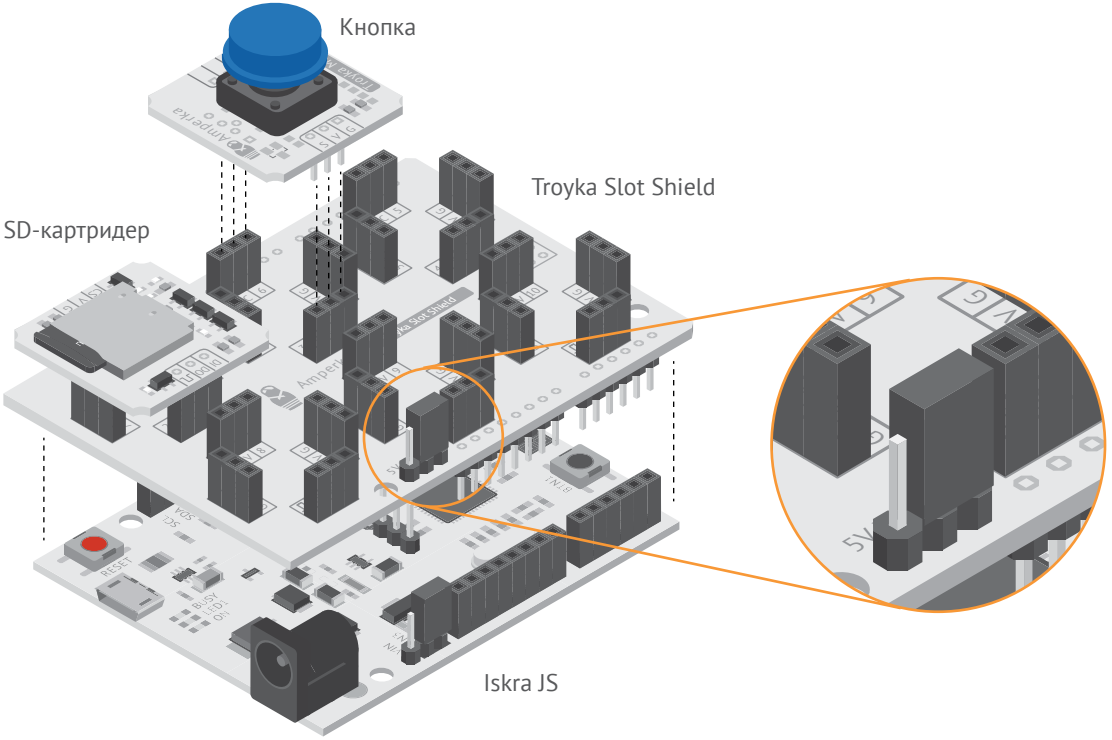
■ Джампер выбора напряжения

ЧТО МОЖНО ПОДКЛЮЧИТЬ

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

КАК ПОДКЛЮЧАТЬ

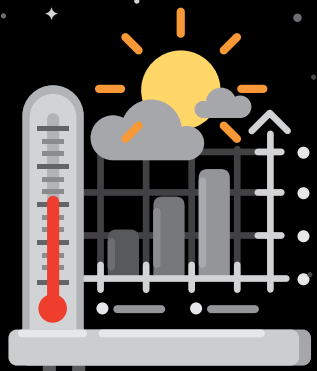
Вставь Slot Shield в пины Iskra JS сверху. Ты получишь единое устройство. Для соединения с компьютером, как и раньше, используй USB-порт на Iskra JS. А для подключения модулей используй разъёмы на Slot Shield.

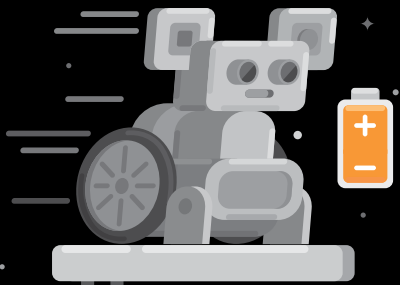


НАПРЯЖЕНИЕ ПИТАНИЯ

В трёх слотах используется альтернативная линия питания V2, напряжение на которой можно выбирать джампером:

- V2+5V — на V2 будет 5 вольт вне зависимости от рабочего напряжения управляющей платы;
- V2+Vin — на V2 будет напряжение порта Vin управляющей платы.





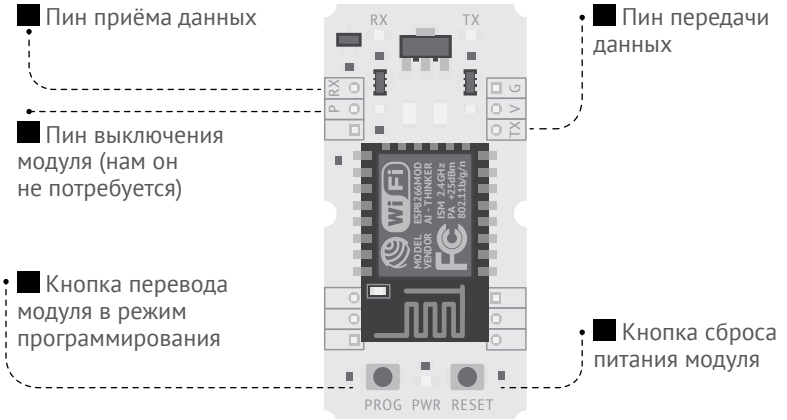
№ 1

УДАЛЁННЫЙ ТЕРМОМЕТР

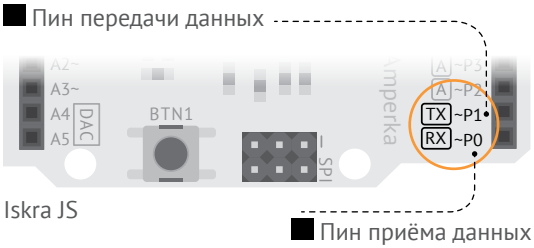
Плату Iskra JS в интернет выходить научи. Информацию о температуре в комнате своей в виде графиков на сайте dweet.io наблюдай.

TROYKA-МОДУЛЬ WI-FI

Будем отправлять данные в интернет через Wi-Fi. Для доступа к Wi-Fi воспользуемся специальным модулем. Модуль общается с Iskra JS по протоколу UART на скорости 115 200 бод.



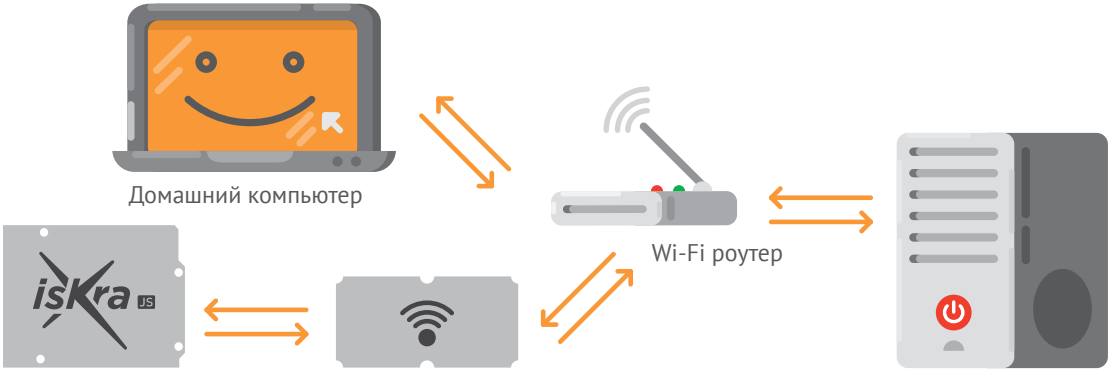
Обрати внимание, пин P0 на Iskra JS отмечен дополнительным значком RX. Это значит, что пин P0 может выполнять функцию приёма данных по протоколу UART. Пин P1 отмечен значком TX, он отправляет данные. Все эти обозначения присутствуют только на плате Iskra JS — поэтому не смущайся, если не видишь их на Slot Shield.



Wi-Fi — это стандарт связи. Сеть Wi-Fi создаётся роутером, к нему подключаются клиенты. Роутер позволяет клиентам выйти в интернет если он сам имеет к нему доступ. Для подключения к роутеру, необходимо знать SSID (имя сети) и пароль сети.

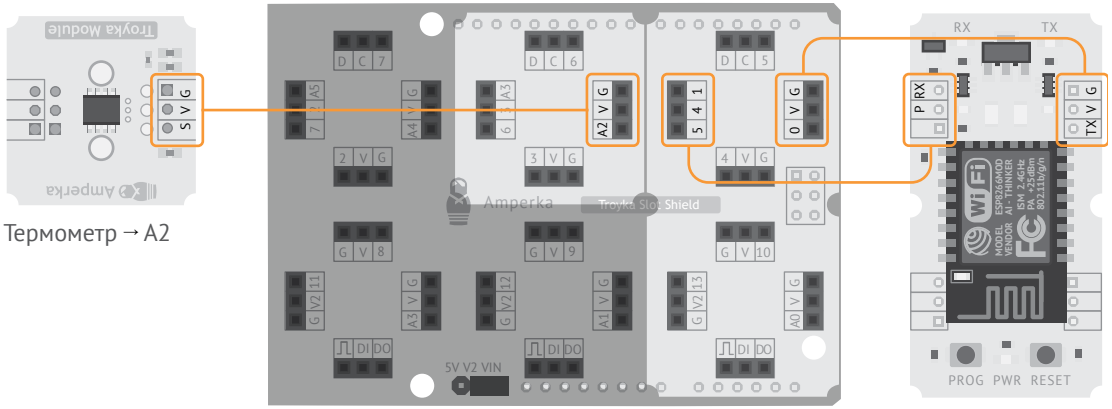
ВНИМАНИЕ!

Если ты не пользуешься роутером, переведи смартфон в режим «точки доступа» iot.amperka.ru → «Настройка точки доступа».



Установи модуль Wi-Fi и датчик температуры на Slot Shield.

Wi-Fi модуль:
Tx → P0 RX
Rx → P1 TX



Подключимся к сервису dweet.io и начнём отправлять показания датчика температуры, а в браузере будем наблюдать за ними на красивом графике.

Сервис dweet.io каждую секунду получает различную информацию от тысяч устройств. Чтобы различать их между собой, сервису нужно сообщить свой уникальный ключ, по которому можно однозначно определить отправителя. Ключ нужно придумать самому. Будем использовать в качестве такого ключа строку, состоящую из твоего имени и даты рождения, например `'Amperka03042011'` — Амперка, 03 апреля 2011.

ВНИМАНИЕ!
Чтобы не набирать код программы вручную, скопируй его с сайта iot.amperka.ru.

```
1  var SSID = 'имя_твоего_wi-fi';
2  var PASSWORD = 'пароль_твоего_wi-fi';
3  var NAME = 'твой_уникальный_ключ';
4
5  var temp = require('@amperka/thermometer').connect(A2);
6  var dweet = require('@amperka/dweetio').connect(NAME);
7
8  function run() {
9    setInterval(function() {
10     dweet.send({
11       temperature: temp.read('C')
12     });
13   }, 1000);
14 }
15
16 var wifi = require('@amperka/wifi').setup(function(err) {
17   wifi.connect(SSID, PASSWORD, function(err) {
18     print('Click this link', dweet.follow());
19     run();
20   });
21 });
```

1 Задаём имя Wi-Fi-сети и пароль.

2 Задаём `'твой_уникальный_ключ'` — только латинские буквы и цифры, без пробелов.

3 Подключаем библиотеку `'@amperka/dweetio'` для работы с сервисом dweet.io. В функцию connect передаём в качестве параметра переменную NAME, содержащую твой уникальный ключ.

4 Создаём функцию `run()`, которую запустим, как только Wi-Fi модуль подключится к сети. Интервальная функция `setInterval` каждую секунду будет вызывать `dweet.send()`.

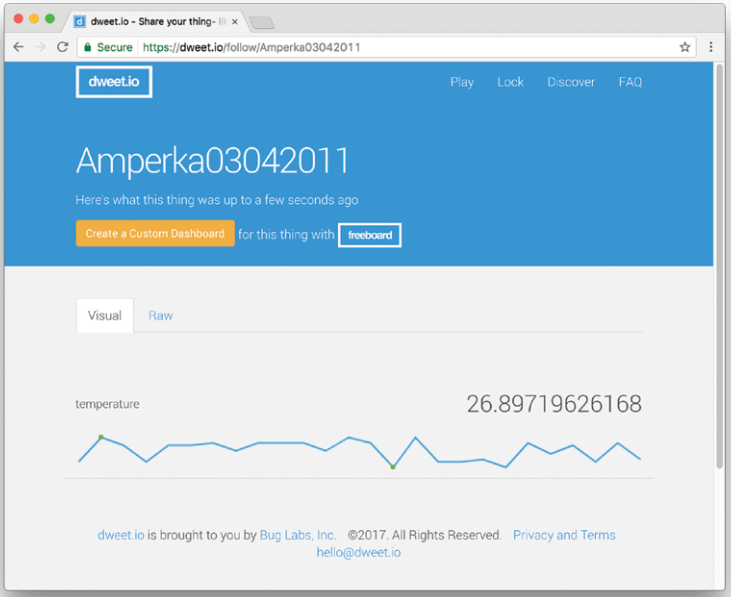
5 Функция `send()` отправляет данные в необходимом формате. В функцию передаём объект с полем `temperature`.

6 Подключаем библиотеку `'@amperka/wifi'` и сразу подключаемся к сети с именем, указанным в переменной SSID, и паролем из переменной PASSWORD. Библиотека сама задаёт скорость передачи данных для модуля Wi-Fi, это 115200 бод.

Загрузи код в Iskra JS, запусти программу и дождись ссылки в консоли.

```
http://dweet.io/follow/Amperka03042011
```

Кликни на неё и в браузере увидишь температуру своей комнаты.



ВНИМАНИЕ!

Сервис dweet.io позволяет отправлять данные не чаще 1 раза в секунду, поэтому нет смысла задавать интервал вызова функции `dweet.send()` меньше 1000 миллисекунд.

7 Как только модуль подключится к сети, функция `follow()` получит адрес, по которому можно отслеживать данные о температуре. А функция `print()` выведет строку `'Click this link'` и адрес в консоль.

8 Вызываем функцию `run()`, чтобы начать отправку данных сервису dweet.io.

ЗАДАНИЕ

Набор «Йодо» вспомни ты и ограничь точность приборов показаний до значений целых.

Другие датчики подключить попробуй. Разные параметры ты сможешь наблюдать. На синтаксис кода внимание обрати, простой запятой объекты разделяй.

Замени NAME на `'Amperka'`, а параметр `'temperature'` на своё имя. Возможно, в этот момент кто-нибудь сделает так же и вы сможете сравнить температуры в ваших комнатах.

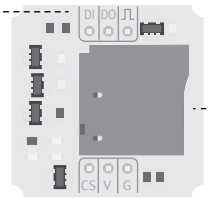
№ 2 ВОСЬМИБИТНЫЙ АУДИОПЛЕЕР

Музыки проигрыватель сделаем себе. На флеш-карте треки хранить будем, зуммером их воспроизводить и кнопкой одной лишь переключать.

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!

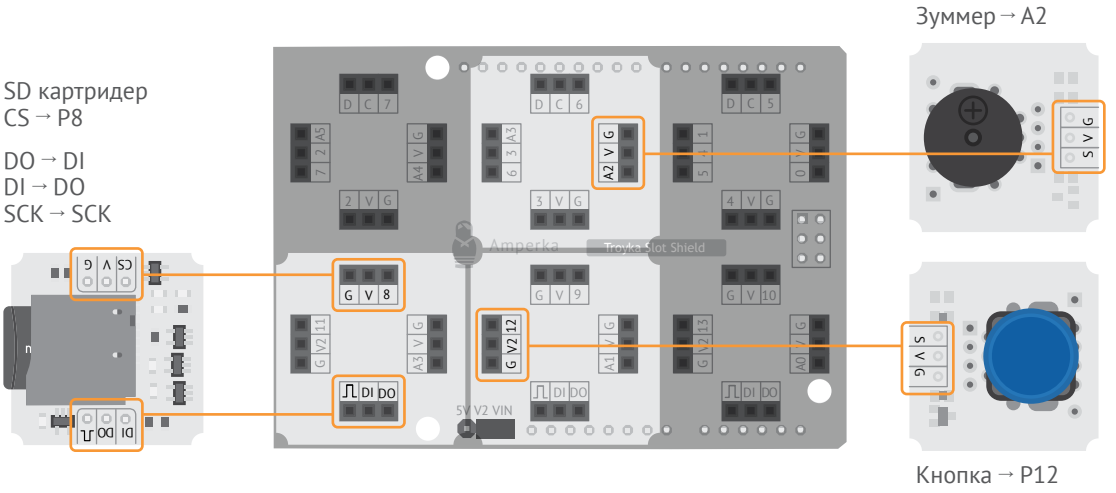
На твоей флешке уже есть файлы. Они потребуются для этого и следующих проектов. Не изменяй и не удаляй их, пока не пройдёшь все эксперименты. Но если вдруг что-то пойдёт не так, скачай файлы с сайта iot.amperka.ru → «Файлы на флеш-карте».

1 Подключаем библиотеку для работы с картридером. Задаём номер пина, управляющий линией CS интерфейса SPI. Библиотека сама задаёт скорость передачи данных для интерфейса.

2 Подключаем библиотеку воспроизведения рингтонов. Функция `connect(A2)` говорит, что звук будет воспроизводиться с модуля, подключённого к пину A2.

3 Нажимая на кнопку `shuffle`, будем менять композицию.

4 Прежде чем включить следующий трек, нужно выключить текущий. Это и делает функция `stop()`.




```
var sdCard = require('@amperka/card-reader').connect(P8);
var player = require('@amperka/ringtone').create(A2);
var shuffle = require('@amperka/button').connect(P12);

shuffle.on('click', function() {
  player.stop();
  var melody = sdCard.readRandomFile('/music');
  player.play(melody);
});

shuffle.on('hold', function() {
  player.stop();
});
```

5 Функция `player.play()` начнёт воспроизведение мелодии.

6 Функция `readRandomFile()` возвращает содержимое произвольного файла в папке `'/music'`.

7 Если кнопку `shuffle` удерживать в течение 1 секунды, воспроизведение остановится.

Собери устройство и нажми на кнопку. Чтобы остановить воспроизведение, зажми кнопку на секунду. Попробуй угадать все мелодии из плейлиста (подсказка: всего на флешке их 4).

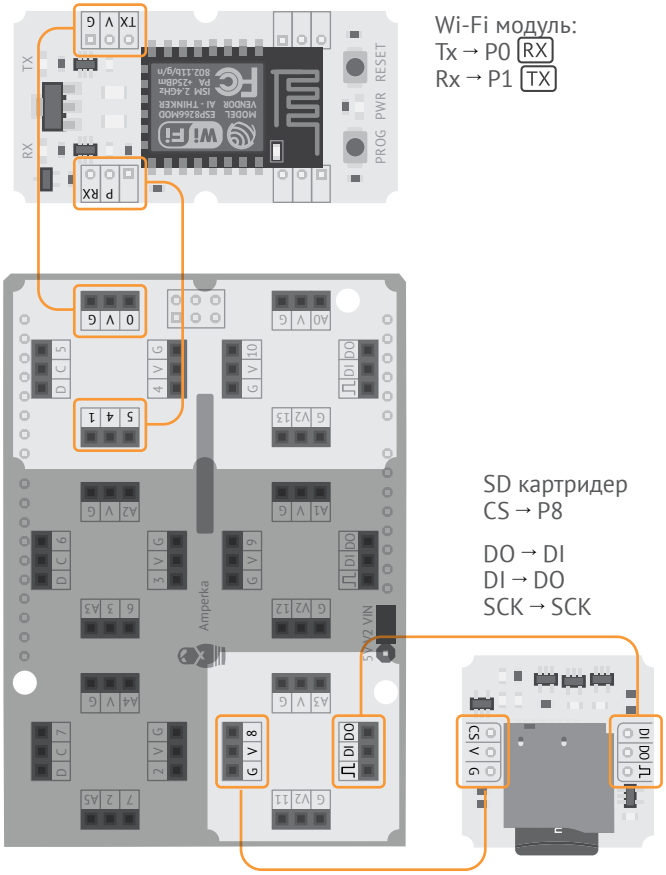
ЗАДАНИЕ

В проекте кнопку на ИК-приёмник замени. На расстоянии музыкой управляй. Проектами № 17 и № 18 из набора «Йодо» воспользуйся.

№ 3

БРАУЗЕРНЫЙ DENDY

Роботом управлять будем в игре браузерной.
С флеш-карты её загрузим и настоящими джедаями станем.



1 Подключаем библиотеку `'@amperka/server'`. Функция `create()` создаёт HTTP-сервер, который будет обрабатывать запросы клиентов.

2 Обработчик событий `'/'` будет выдавать клиентам html-страницу, на которой и будет находиться наша игра. `'/'` означает, что клиент запросил главную страницу сайта. В нашем случае это `'race.html'`. Этот файл лежит в корневой папке SD-карты. Читаем его функцией `readFile()`, а функцией `send()` отдаём обратно клиенту. Содержимое файла `'race.html'` будет ответом нашего сервера.

3 С помощью функции `getIP` объекта `wifi` узнаём IP-адрес сервера (IP-адрес Iskra JS). По этому адресу будем подключаться и отправлять HTTP-запросы.

```
var SSID = 'имя_твоего_wi-fi';
var PASSWORD = 'пароль_твоего_wi-fi';

var sdCard = require('@amperka/card-reader').connect(P8);
var server = require('@amperka/server').create();

server.on('/', function(req, res) {
  var content = sdCard.readFile('race.html');
  res.send(content);
});

var wifi = require('@amperka/wifi').setup(function(err) {
  wifi.connect(SSID, PASSWORD, function(err) {
    wifi.getIP(function(err, ip) {
      server.listen();
      print('Game is ready! http://'+ip+'/');
    });
  });
});
```

4 `listen()` запускает сервер. Буквально это означает: «Сервер, начни слушать, что у тебя запросят клиенты».

5 Выводим в консоль ссылку, по которой можно запустить гонки на Робоняшах. С помощью оператора `'+'` объединяем строку и переменную `ip` для вывода в консоль.

Загрузи код в плату и дождись ссылки в окне консоли, похожей на эту:

```
I'm ready! Click this link: http://192.168.10.20/
>
```

Открой страницу, кликнув на ссылку, и выигрывай гонки как настоящий джедай!

ЗАДАНИЕ

Рекордом своим в социальных сетях с другими джедаями поделись.

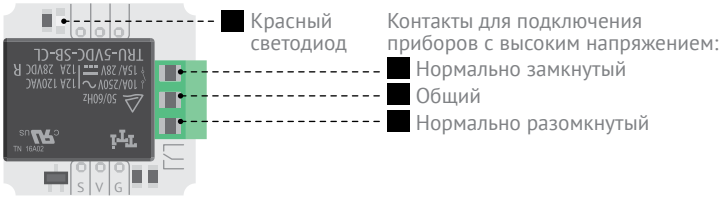
№ 4

УМНЫЙ ДОМ

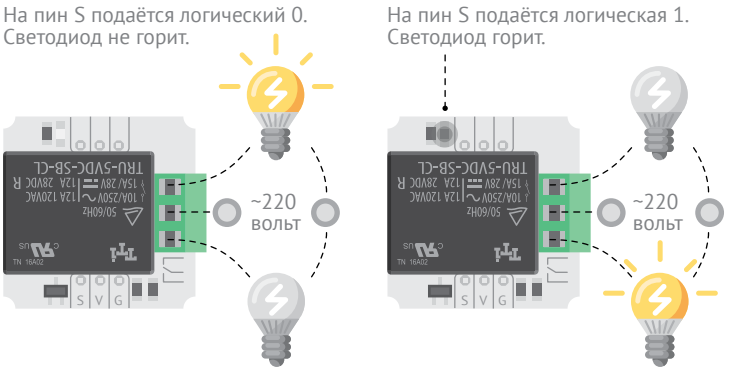
Дом свой разумом надели. Железку новую освой для этого, которая напряжением высоким управлять может.

TROYKA-МОДУЛЬ МИНИ-РЕЛЕ

Мини-реле способно выключать и включать питание, будто выключатель света на стене. Только тот управляется вручную, а реле управляется программой с Iskra JS. Реле способно работать под напряжением 220 вольт.



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 пару контактов.



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.

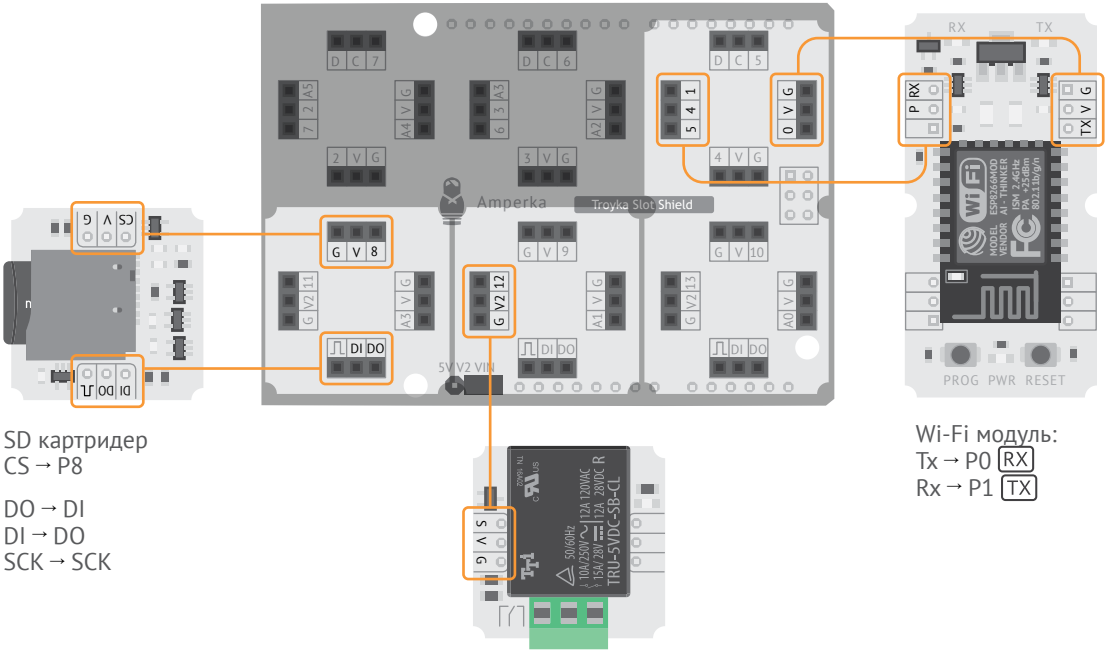
Вместо этого используй светодиод на реле. Если горит — реле включено. Если нет — выключено.

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 Iskra JS. Внезапно тёмным вечером Iskra JS 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.

Будем включать и выключать реле из браузера. Сделаем из Iskra JS домашний сервер, работающий по протоколу HTTP.

На флеш-карте будем хранить html-страницу для отображения в браузере.



SD картридер
CS → P8

DO → DI
DI → DO
SCK → SCK

Мини-реле → P12

Wi-Fi модуль:
Tx → P0 RX
Rx → P1 TX

```
var sdCard = require('@amperka/card-reader').connect(P8);
var relay = require('@amperka/relay').connect(P12);
var server = require('@amperka/server').create();

var SSID = 'имя_твоего wi-fi';
var PASSWORD = 'пароль_wi-fi';

server.on('/', function(req, res) {
  var content = sdCard.readFile('light.html');
  res.send(content);
});

server.on('/turnOn', function() {
  relay.turnOn();
});

server.on('/turnOff', function() {
  relay.turnOff();
});

var wifi = require('@amperka/wifi').setup(function(err) {
  wifi.connect(SSID, PASSWORD, function(err) {
    wifi.getIP(function(err, ip) {
      server.listen();
      print('Relay control is ready! http://'+ip+'/');
    });
  });
});
```

1 Подключаем библиотеку для управление реле `'@amperka/relay'`. Подключаем реле к пину P12.

2 На первый запрос клиентов к серверу выдаём html-страницу `'light.html'`.

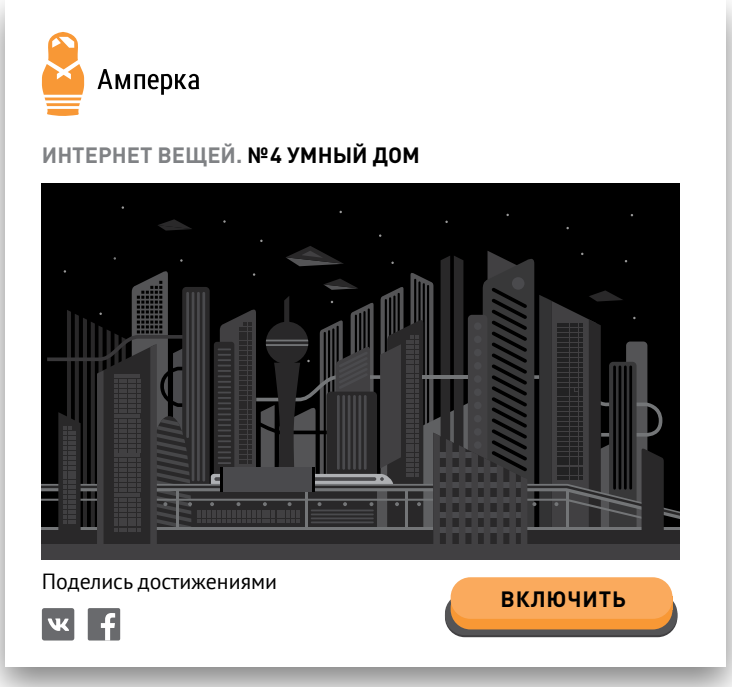
3 Обработчик событий `'/turnOn'` сработает, когда ты нажмёшь на кнопку «Включить» в браузере. При нажатии на кнопку браузер сделает дополнительный запрос серверу. Сервер обработает запрос и включит реле функцией `turnOn()`.

4 Обработчик событий `'/turnOff'` выключит реле функцией `turnOff()`, если ты нажмёшь кнопку «Выключить» в браузере.

5 Узнаём IP-адрес Iskra JS, по которому будем подключаться к панели управления светом.

```
I'm ready! Click this link: http://192.168.10.20/
>
```

Запусти панель управления, кликнув на ссылку, и 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.



ЗАДАНИЕ

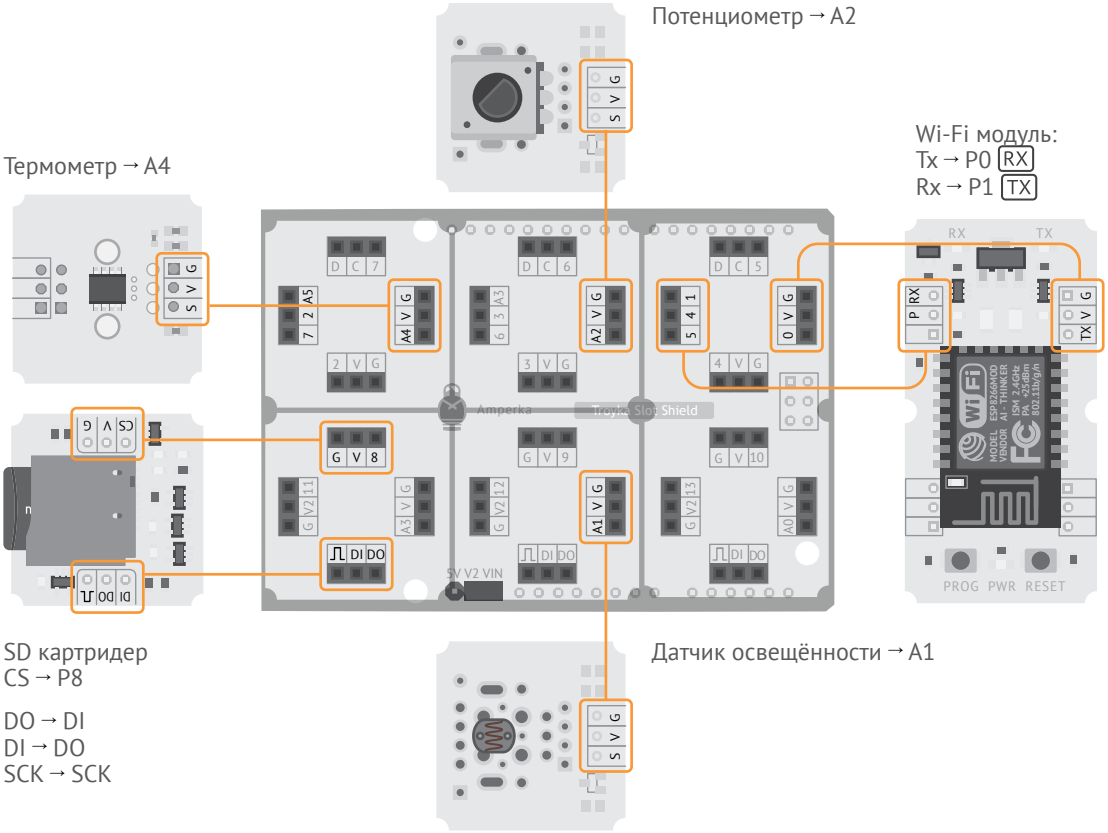
Зуммер на Slot Shield установи. Звуковым сигналом включение света снабди.

№ 5 ИНТЕРАКТИВНЫЙ ДОМ

На содержимое веб-страницы силами природы и волей своей в реальном времени влиять будем. Прибор для этого с датчиками соберём.

Воспользуемся HTML-страницей home.html в корневой папке флеш-карты.

Подключим аналоговые датчики и будем влиять на разные элементы страницы в зависимости от положения ручки потенциометра, температуры окружающей среды и уровня освещённости в комнате.


Потенциометр → A2
Термометр → A4
Wi-Fi модуль:
Tx → P0 RX
Rx → P1 TX
SD картридер
CS → P8
DO → DI
DI → DO
SCK → SCK
Датчик освещённости → A1
Amperka
Troyka Slot Shield
RX
TX
PROG PWR RESET

```
var sdCard = require('@amperka/card-reader').connect(P8);
var light = require('@amperka/light-sensor').connect(A1);
var pot = require('@amperka/pot').connect(A2);
var temp = require('@amperka/thermometer').connect(A4);

var SSID = 'имя_твоего_wi-fi';
var PASSWORD = 'пароль_wi-fi';

var server = require('ws').createServer(function(req, res) {
  var content = sdCard.readFile('home.html');
  res.end(content);
});

function prepareData() {
  var data = {
    speed: Math.floor(pot.read() * 41) - 20,
    light: Math.floor(light.read('lx') / 640),
    temperature: Math.floor(temp.read('C'))
  };
  return JSON.stringify(data);
}

server.on('websocket', function(ws) {
  setInterval(function() {
    ws.send(prepareData());
  }, 200);
});

var wifi = require('@amperka/wifi').setup(function(err) {
  wifi.connect(SSID, PASSWORD, function(err) {
    wifi.getIP(function(err, ip) {
      print('I\'m ready! http://'+ip+'/');
      server.listen(80);
    });
  });
});
```

1 Подключаем библиотеку `'ws'` для работы с протоколом WebSocket. Обрати внимание, имя этой библиотеки не начинается с `'@amperka/'`. Сразу же создаём сервер для подключения. На запрос клиента сервер ответит HTML-страницей `'home.html'`.

2 Функцией `prepareData()` подготавливаем данные с датчиков в удобный формат для передачи по протоколу. Собираем данные в единый объект с полями `'speed'`, `'light'` и `'temperature'`, а затем упаковываем объект в строку функцией `JSON.stringify()`.

3 Обработчик событий `server.on()` начнёт свою работу, как только браузер предложит серверу установить соединение по протоколу WebSocket. Как только соединение установится, сервер начнёт отправлять браузеру данные с аналоговых датчиков функцией `send()`.

4 Функция `read()` для аналоговых датчиков возвращает значения от 0 до 1. Умножив результат на 41, получаем диапазон от 0 до 41. Функция `Math.floor()` округляет дробное число до ближайшего целого в меньшую сторону. Отнимаем 20 и получаем диапазон от −20 до 21. В этом диапазоне будем менять скорость космического дома.

5 Запускаем сервер функцией `listen()`. Параметр 80 означает, что мы используем 80-й порт. Это стандартный номер порта для HTTP-запросов.

Как и в прошлом проекте, дождись ссылки в консоли и кликни на неё.

```
I'm ready! Click this link: http://192.168.10.20/
>
```

Посмотри, как показания датчиков влияют на страницу в браузере:
- Попробуй покрутить ручку потенциометра;
- Посвети на датчик освещённости или закрой его от света рукой;
- Нагрей ладонью датчик температуры или приложи к нему что-нибудь из холодильника.

ЗАДАНИЕ

Поуправляй другими элементами страницы.

Свойство	*Параметр объекта*	*Диапазон значений*	*Пример преобразования информации датчика в нужный диапазон*
Координата робота-пылесоса	`cleaner`	`0…600`	`Math.floor(pot.read() * 601)`
Скорость космического корабля	`speed`	`−20…20`	`Math.floor(pot.read() * 41) - 20`
Температура в печи	`oven`	`0…255`	`Math.floor((temp.read('C') - 20) * 20)`
Количество включённых лампочек	`light`	`0…5`	`Math.floor(light.read('lx') / 640)`
Температура внутри корабля	`temperature`	`−40…+125`	`Math.floor(temp.read('C'))`

№ 6

НАПОМИНАЛЬНИК

Важные события жизни своей в почте оставляй. Кота покормил? В почту себе напоминание пришли, чтобы в тонусе падавана держать.

Для отправки писем используется особый протокол: SMTP. Он не очень удобен для работы напрямую с Iskra JS. Гораздо проще использовать готовые сервисы. Воспользуемся одним из таких — IFTTT.

Будем высылать на свой email-адрес письмо, содержащее время последнего кормления кота или выполнения любого другого рутинного действия. Свяжем между собой два компонента: сервис Webhooks и электронную почту.

Сервис Webhooks умеет принимать простые HTTP-запросы и перенаправлять их IFTTT.

Заведи себе учётную запись на сайте ifttt.com

Покормил кота?

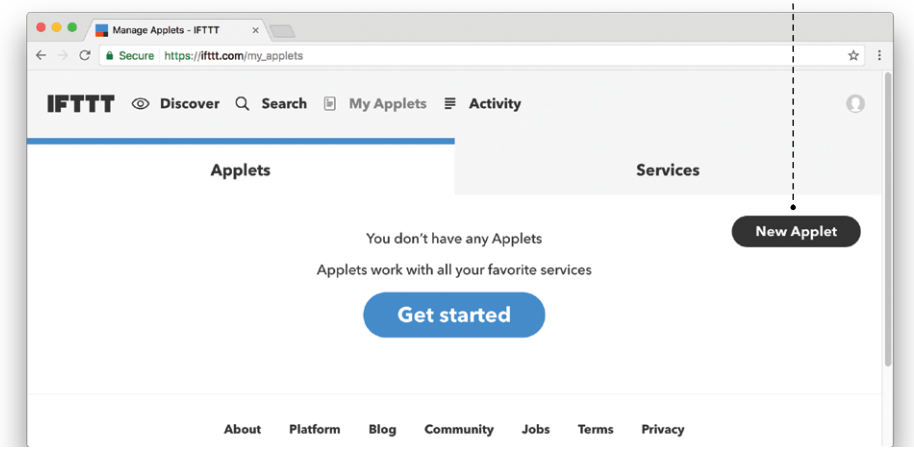
Нажми на кнопку, Iskra JS отправит HTTP-запрос к сервису Webhooks.

Webhooks перенаправит запрос сервису IFTTT.

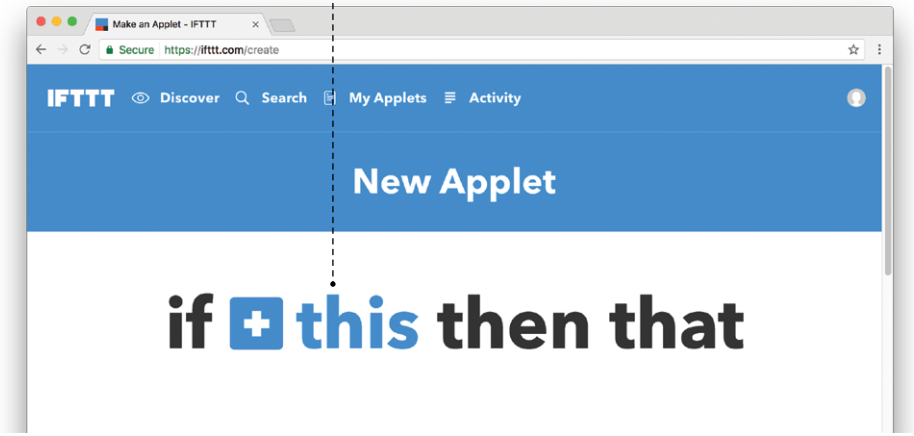
IFTTT отправит на твой адрес письмо.

Если кот попросит еды, проверь в почте, не забыл ли с утра его покормить.

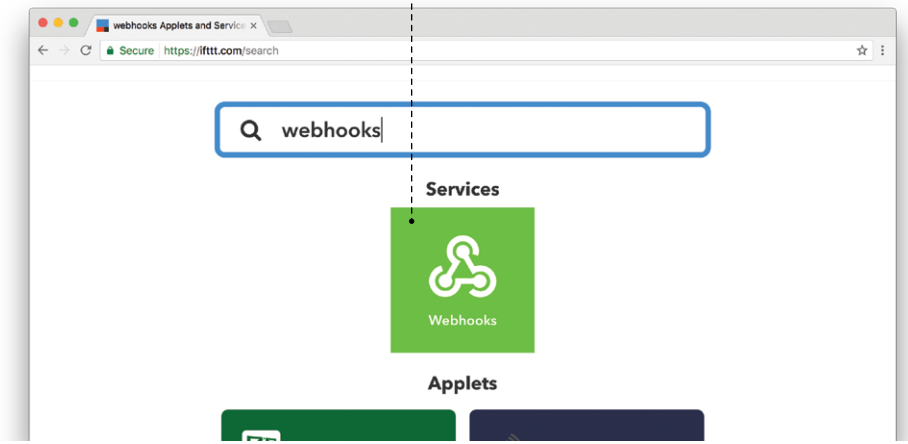
1 Зайди на ifttt.com. Нажми вверху кнопку «My Applets», а затем «New Applet».



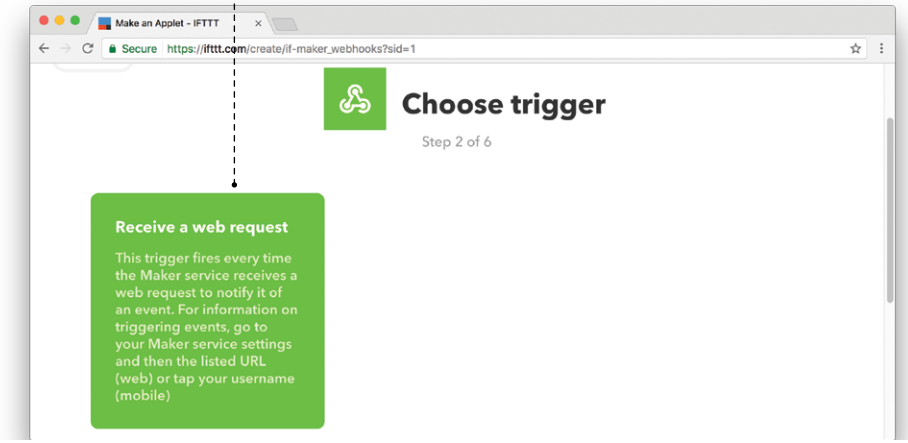
2 Нажми кнопку «this».



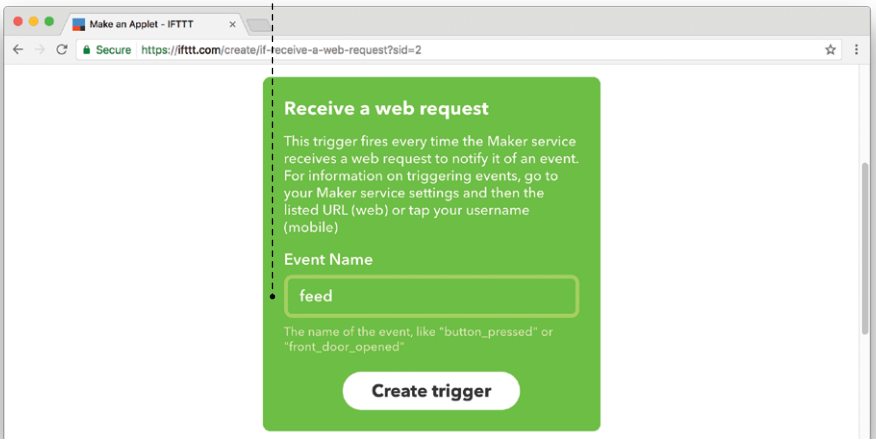
3 Набери в поисковой строке «webhooks». В появившемся списке выбери «Webhooks».



4 Выбери действие «Receive a web request».

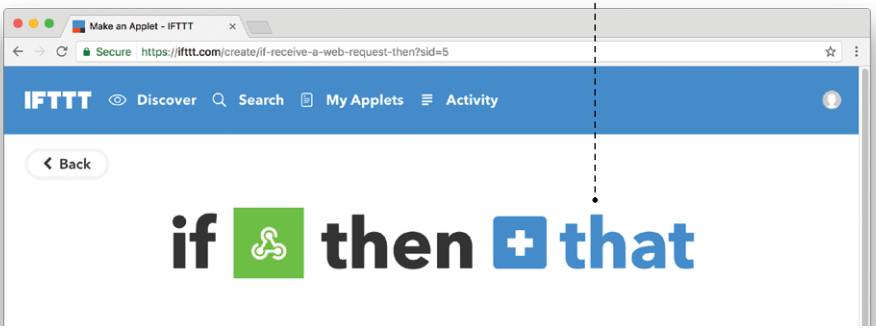


5 Поле «Event Name» назови «feed» и нажми кнопку «Create trigger».

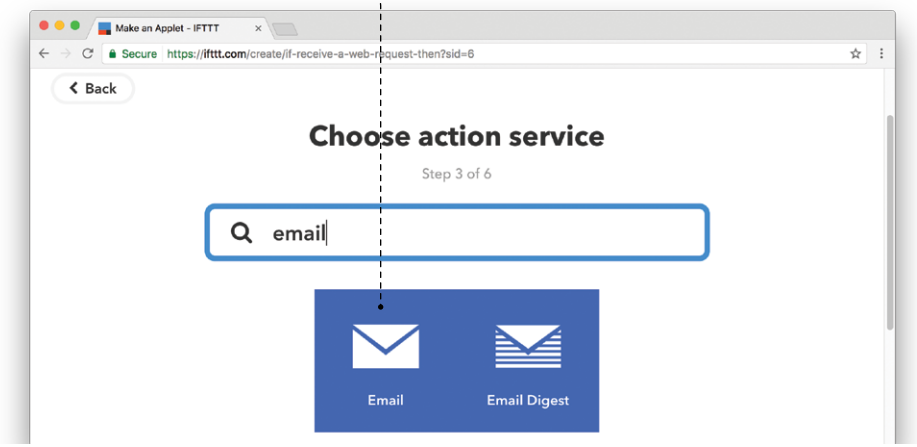


6 Мы закончили создавать условие (триггер). Теперь сервис Webhooks готов принимать HTTP-запросы от Iskra JS. Это похоже на работу сервиса dweet.io, но Webhooks не строит графики, а передаёт информацию дальше сервису IFTTT. Теперь для IFTTT нужно задать действие при срабатывании нашего триггера.

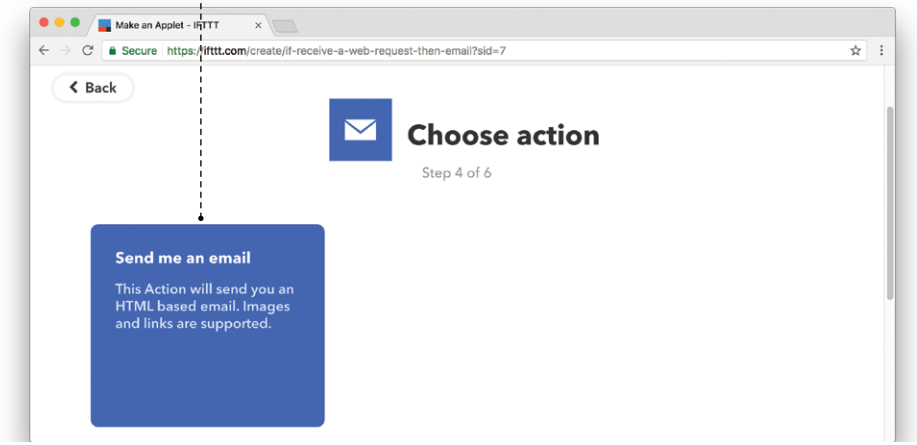
Нажми кнопку «that».



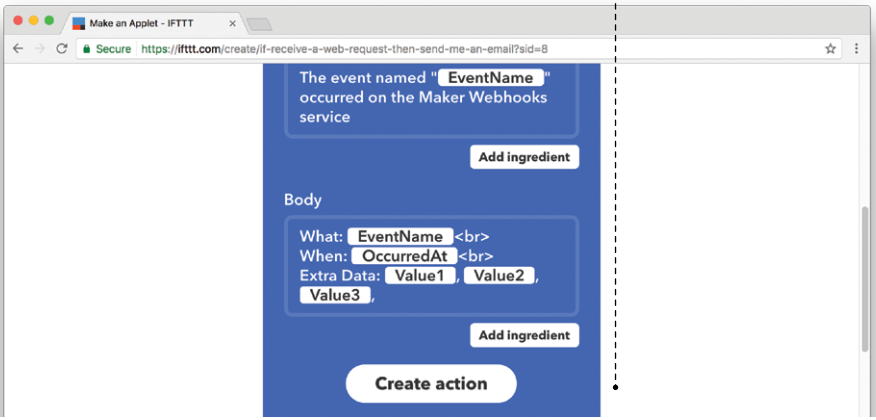
7 Набери в поисковой строке «email» и выбери появившийся компонент.



8 Выбери действие «Send me an email».

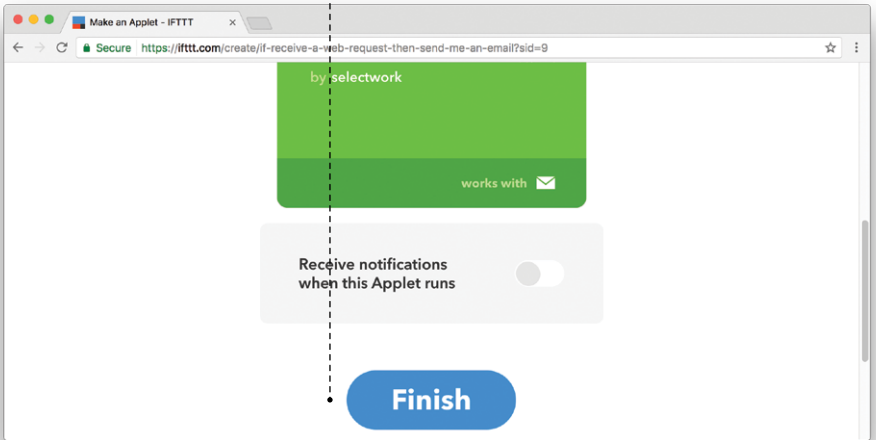


9 Задай образец письма. Придумай заголовок (Subject), а тело письма (Body) оставь нетронутым. Затем нажми «Create action».



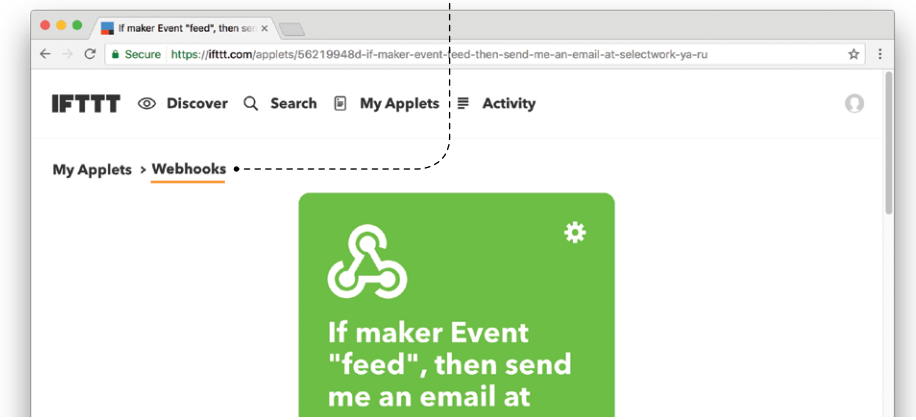
10 Поля Value1, Value2 и Value3 позволяют передавать в письме дополнительные данные. Отправляя с Iskra JS запрос сервису Webhooks, можно указать значения этих полей.

Наконец, жми «finish».

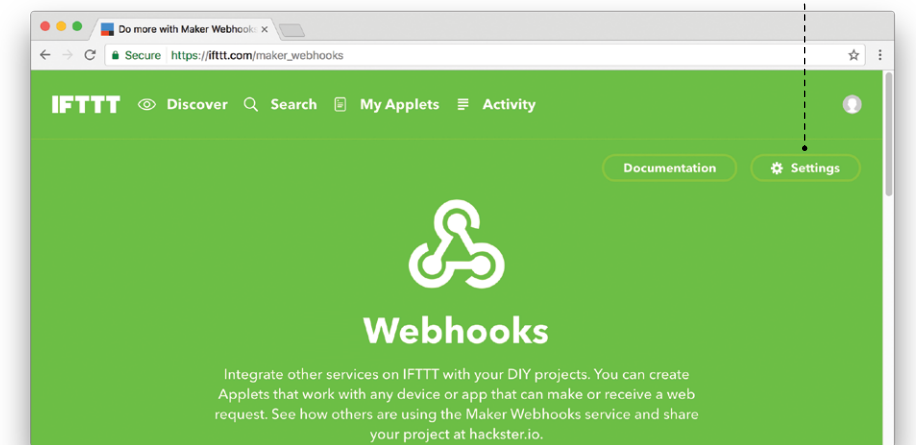


11 Всё готово. Помнишь, в проекте с dweet.io ты придумывал себе уникальный ключ? Webhooks тоже использует уникальный ключ, но выдаёт его самостоятельно.

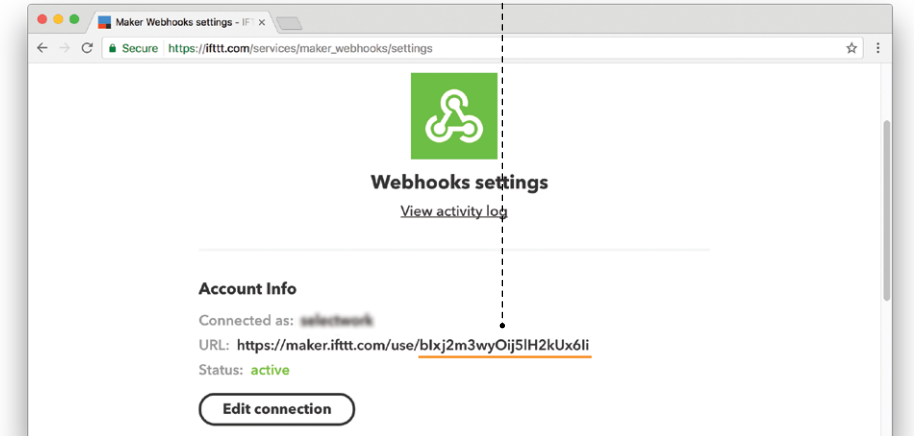
Чтобы узнать этот ключ, перейди во вкладку «Webhooks».



12 Нажми кнопку «Settings»



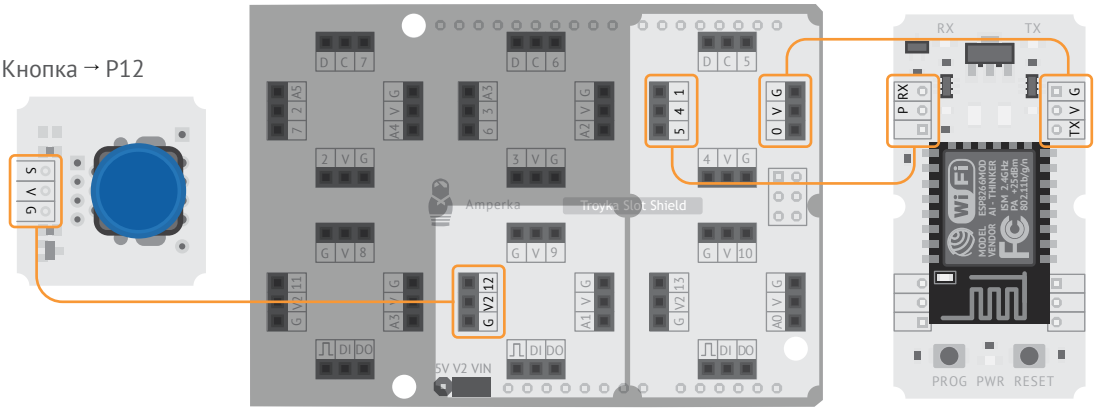
13 Скопируй часть ссылки «URL». Потребуется только последняя часть адреса. Вставь её в код для Iskra JS.



14 Установи на Slot Shield модуль кнопки.

Wi-Fi модуль:
Tx → P0 RX
Rx → P1 TX

Кнопка → P12



```
var SSID = 'имя_твоего wi-fi';
var PASSWORD = 'пароль_wi-fi';

var trigger = require('@amperka/button').connect(P12);
var maker = require('@amperka/ifttt-webhooks').create({
  token: 'токен_сервиса_webhooks',
  action: 'feed'
});

var wifi = require('@amperka/wifi').setup(function(err) {
  wifi.connect(SSID, PASSWORD, function(err) {
    print('Ok, hold button.');
  });
});

var data = {
  value1: 'I did it!'
};

trigger.on('hold', function() {
  maker.send(data, function(response) {
    print(response);
  });
});
```

1 Подключаем библиотеку `'@amperka/ifttt-webhooks'` для работы с сервисом. Передаём в функцию `create` параметры `token` и `action`. `token` — уникальная часть адреса сервиса Webhooks, на который будем отправлять запросы, `action` — название нашего события, в письме оно указано как EventName.

2 Готовим данные для отправки сервису (`'Extra data'` в теле письма). Храним их в переменной `data`. Это объект с полем value1 и значением `'I did it!'` («Я сделал это!»). Сервис Webhooks может принять максимум 3 поля: `value1`, `value2` и `value3`. Поля могут принимать любые значения. Указывать их вовсе не обязательно, поэтому будем использовать только `value1`.

3 Функция `send` отправляет HTTP-запрос к сервису Webhooks. В ответ сервис возвращает строку, выводим её в окно консоли.

Дождись сообщения в консоли, что Iskra JS подключилась к Wi-Fi.

```
I'm ready!
>
```

15 Нажми и удерживай кнопку. В консоли увидишь надпись:

```
I'm ready!
Congratulations! You've fired the feed event
```

В течение нескольких минут на твою почту придёт сообщение с твоим заголовком и дополнительным полем `'I did it!'`.

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

ЗАДАНИЕ

Набор «Йодо» вспомни ты. Ультразвуковой дальномер используй, чтобы двери открытия триггер создать.

№ 7

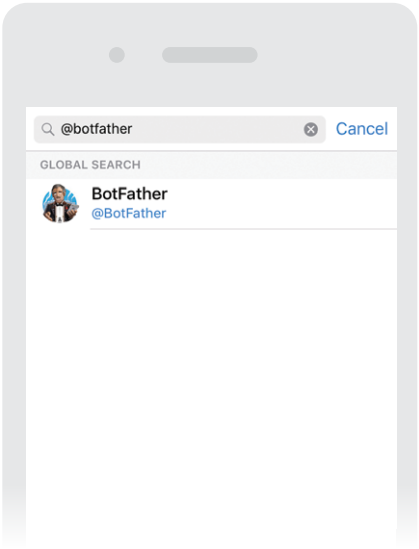
TELEGRAM-BOT

Мессенджером модным овладей. Iskra JS ему обучи.

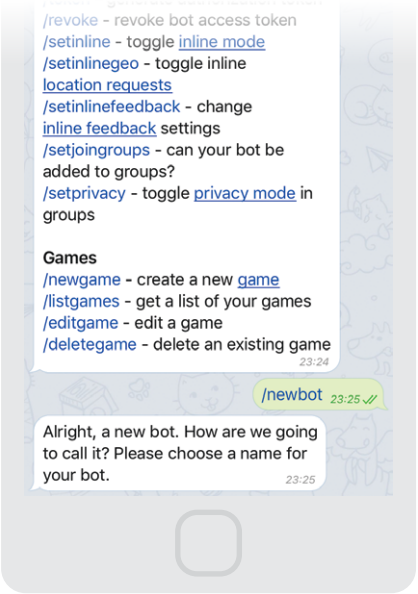
Будем использовать мессенджер Telegram. Если у тебя ещё нет аккаунта в Telegram, заведи его! Установи приложение на свой смартфон или воспользуйся web-версией.

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

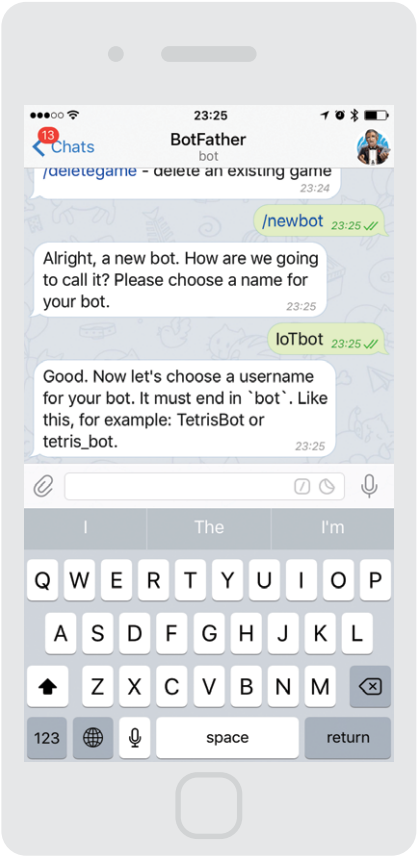
1 Открой мессенджер. В поиске найди главного бота — «@botfather». Заведи с ним диалог и нажми кнопку «START».



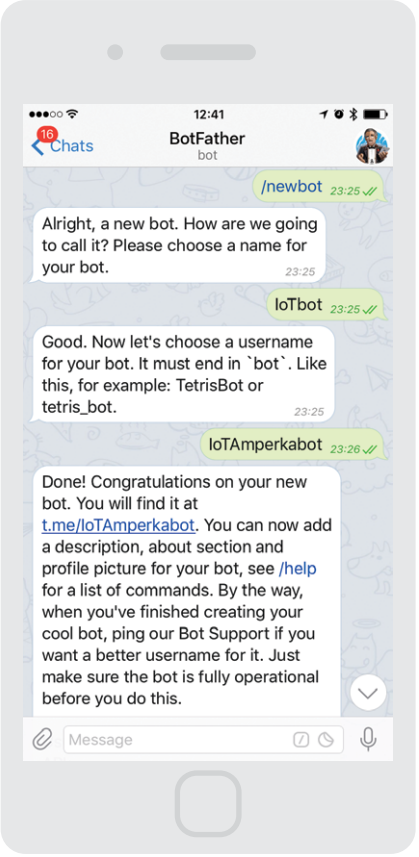
2 Выбери пункт «/newbot».



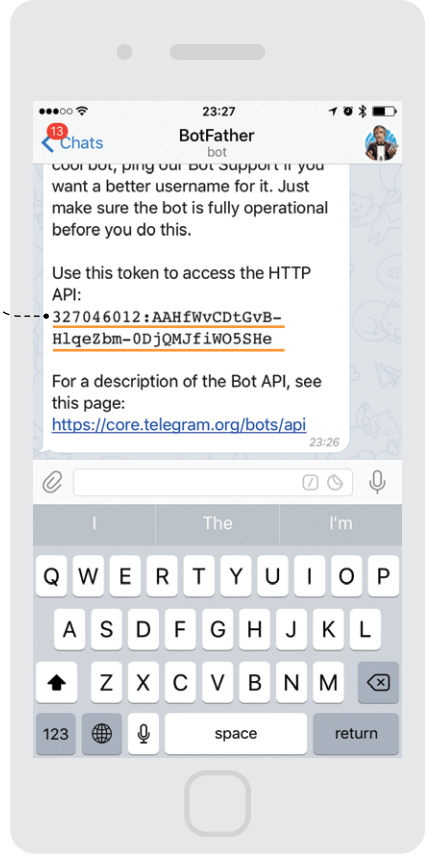
3 Придумай имя своему боту.

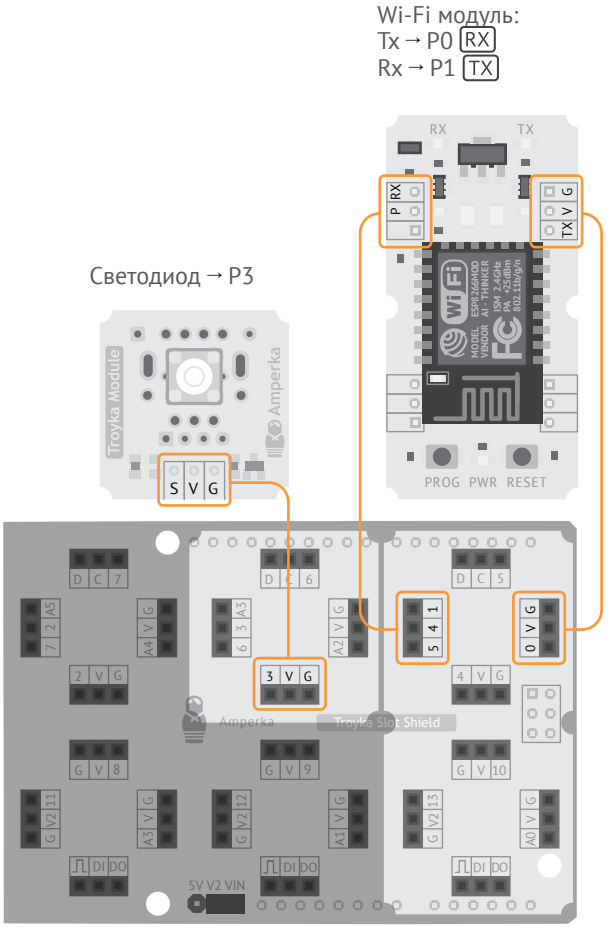


4 Теперь придумай уникальный логин бота. Он обязательно должен заканчиваться на «bot».



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!





```
var SSID = 'имя_твоего wi-fi';
var PASSWORD = 'пароль_wi-fi';

var bot = require('@amperka/telegram').create({
  token: 'здесь_будет_твой_токен',
  polling: { timeout: 10 }
});
var light = require('@amperka/led').connect(P3);

bot.on('/start', function(msg) {
  var keyboard = bot.keyboard([
    ['/TurnOff', '/TurnOn']
  ], { resize: true });
  bot.sendMessage(msg.from.id, 'Light control', {
    markup: keyboard
  });
});

bot.on('/TurnOff', function(msg) {
  light.turnOff();
  bot.sendMessage(msg.from.id, 'Light is off');
});

bot.on('/TurnOn', function(msg) {
  light.turnOn();
  bot.sendMessage(msg.from.id, 'Light is on');
});

var wifi = require('@amperka/wifi').setup(function(err) {
  wifi.connect(SSID, PASSWORD, function(err) {
    print('I\'m ready!');
    bot.connect();
  });
});
```

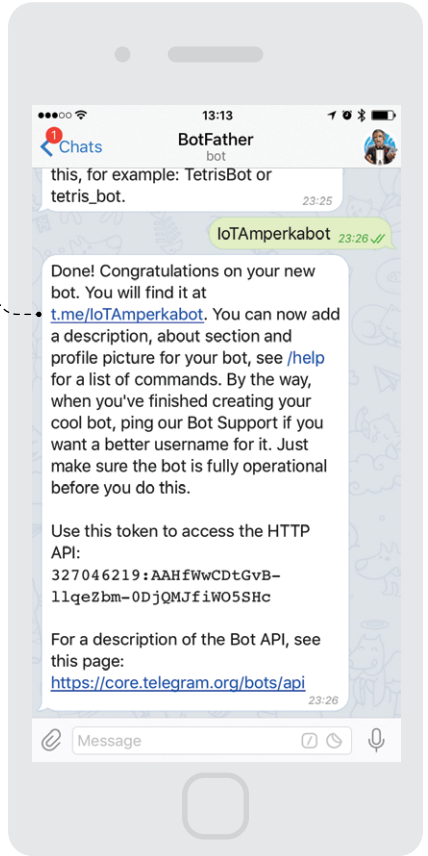
1 Подключаем библиотеку `'@amperka/telegram'` и вызываем функцию `create()`. В параметре `token` указываем токен, который выдал `@BotFather`, а в параметре `polling` указываем `timeout` 10 секунд, чтобы Iskra JS не отправляла запросы к серверам Telegram слишком часто.

2 Обработчик событий `'/start'` будет вызван, когда бот получит сообщение, что собеседник нажал кнопку `'/start'`. В ответ бот пришлёт клавиатуру с двумя кнопками: включить свет и выключить. Клавиатура отправляется собеседнику функцией `sendMessage` с дополнительным параметром `markup`.

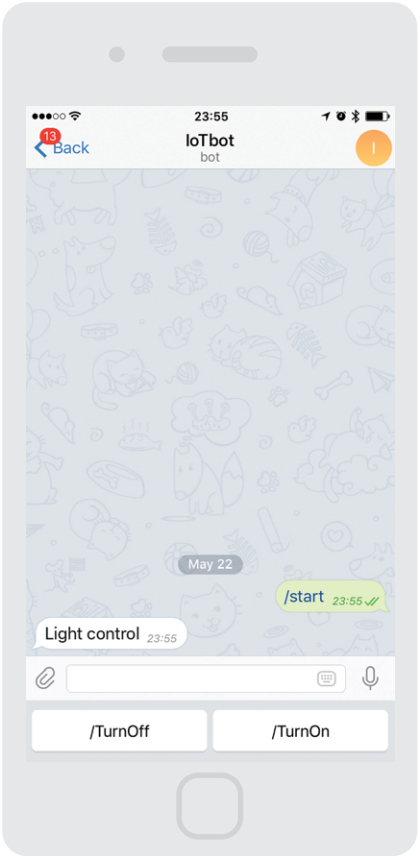
3 События `'/TurnOn'` и `'/TurnOff'` вызываются по нажатию кнопок на клавиатуре. В ответ высылаем подтверждение, что бот исполнил команду.

4 Функция `connect()` запускает работу с сервисом Telegram.

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



7 Теперь добавь своего бота в контакты и нажми кнопку «START»!



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

Обрати внимание. Сервис Telegram работает только по протоколу HTTPS (шифрованное соединение HTTP). Для шифрования данных требуется много вычислительных ресурсов, поэтому задержка отклика Iskra JS на сообщение в Telegram заметна глазу.

ЗАДАНИЕ

Подключи к Iskra JS датчики освещённости и температуры. Пусть по команде сообщение тебе придёт о температуре и яркости в комнате (чтобы обновить клавиатуру на экране, набери `/start` в текстовом окне чата). Воспользуйся примерами клавиатур, чтобы сделать свою собственную.

```
var keyboard = bot.keyboard([
    ['/TurnOff', '/TurnOn'],
    ['/LightUp', '/LightDown']
], { resize: true });
```


```
var keyboard = bot.keyboard([
    ['/TurnOff', '/TurnOn'],
    ['/WarmUp_Teapot'],
    ['/Backward', '/Play_Pause', '/Forward']
], { resize: true });
```

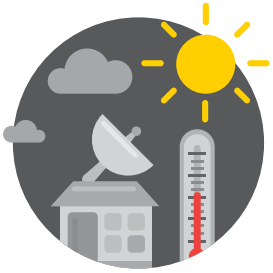

```
var keyboard = bot.keyboard([
  ['/Wau']
], { resize: false });
```

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!

ИДЕИ ПРОЕКТОВ

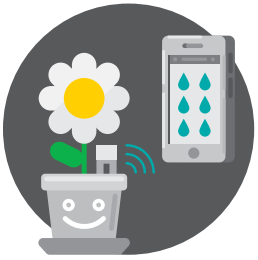
Прошёл набор «IoT»? Молодец!

Вот тебе идеи для приборов, которые пригодятся в быту. Применяй знания, комбинируй 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.

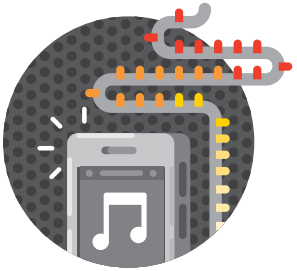


АВТОПОЛИВ

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 Webhooks!

ПОЖАРНАЯ СИГНАЛИЗАЦИЯ

Датчик горючих и угарного газов MQ-9 даст тебе уверенность в безопасности твоего жилья. В случае, если он обнаружит в воздухе присутствие опасных газов, он сможет прислать сообщение тебе прямо в Telegram!



СВЕТОМУЗЫКА

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

АНТИВОР

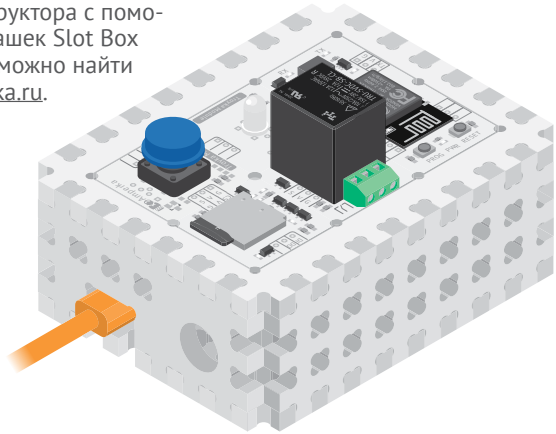
Датчик шума и ультразвуковой дальномер позволят тебе построить систему сигнализации в твоей комнате — ты всегда будешь знать, если в ней появятся непрошеные гости. А модуль MP3-плеера в комбинации с модулем аудиовыхода сможет твоим голосом вежливо попросить их покинуть твою территорию.

АВТОНОМНЫЙ ГАДЖЕТ

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

#СТРУКТОР

Устройства на Troyka Slot Shield можно удобно оформлять в корпус из #Структора с помощью набора плашек Slot Box (#Структор). Их можно найти на сайте amperka.ru.



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

СПРАВОЧНИК

ПРОТОКОЛ HTTP

Данные по протоколу HTTP передаются в виде текстовых строк. В начале текста записаны заголовки (headers), а 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 всякую служебную информацию.

Например, запрос браузера к сайту «Амперки» (запрос клиента к серверу по протоколу HTTP) выглядит примерно так:

```
Host: amperka.ru
User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5
Accept: text/html
```

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

2 `User-Agent` сообщает название браузера клиента. Это необходимо для правильного отображения страницы в разных браузерах.

3 Поле `Accept` означает, в каком формате клиент готов принять ответ от сервера. В данном случае клиент ожидает html-страницу.

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 запрошенная страница в HTML
```

1 Первая строчка в ответе сервера означает статус ответа. 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: длину в байтах, кодировку, язык, 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.

СЕРВИС DWEET.IO

```
var dweet = require('@amperka/dweetio').connect('твой_уникальный_ключ');

dweet.send({
  temperature: 36.6,
  light: 700,
  noise: 40
})
```
— отправить сервису объект с полями: temperature, light и noise.

`dweet.follow()` — получить строку с адресом страницы, где видны графики.

Подробнее на wiki.amperka.ru/js:dweetio.

WI-FI

```
var wifi = require('@amperka/wifi').setup(function(err) {
  print(err);
});
```
— подключить модуль Wi-Fi и вывести в консоль информацию о возможных ошибках.

```
wifi.connect(SSID, PASSWORD, function(err) {
  print(err);
});
```
— подключиться к сети Wi-Fi с именем SSID и паролем PASSWORD.

```
wifi.getIP(function(err, ip) {
  print(err, ip);
});
```
— получить IP-адрес, выделенный роутером. Вывести на печать адрес и возможные ошибки.

Подробнее на wiki.amperka.ru/js:wifi.

КАРТРИДЕР

`var sdCard = require('@amperka/card-reader').connect(P4);` — подключить картридер с пином CS на P4.

`sdCard.readDir('/');` — получить список всех файлов и папок в корне SD-карты.

`sdCard.isDirectory('folder');` — проверить, является «folder» папкой или файлом.

`sdCard.readFile('/music/ringtone.rtttl');` — прочитать содержимое файла «ringtone.rtttl» из папки «music».

`sdCard.readRandomFile('/music');` — прочитать содержимое произвольного файла в папке «music».

`sdCard.writeFile('new_data.txt', 'IoT will save the world');` — создать новый файл с именем «new_data.txt» и записать в него строку «IoT will save the world».

`sdCard.appendFile('data.txt', 'Iskra JS');` — дописать в файл «data.txt» строку «Iskra JS».

Подробнее на wiki.amperka.ru/js:card-reader.

HTTP-СЕРВЕР

```
var server = require('@amperka/server').create();
server.listen(80);
```
— запустить HTTP-сервер на порту 80.

```
server.on('/', function(req, res) {
  res.send('hello from server');
});
```
— обработать запрос и отправить клиенту строку «hello from server».

```
server.on('/turnOn', function() {
  relay.turnOn();
});
```
— обработать запрос на путь «turnOn».

Подробнее на wiki.amperka.ru/js:server.

СЕРВЕР WEBSOCKET

```
var socket = require('ws').createServer(function(req, res) {
  var content = sdCard.readFile('font.html');
  res.end(content);
});
```
— создать HTTP-сервер, поддерживающий протокол websocket.

```
socket.listen(80);
```
— запустить сервер на порту 80.

```
socket.on('websocket', function(ws) {
  ws.send('ping');
});
```
— при запросе клиента на создание соединения websocket отправить клиенту строку «ping».

Подробнее на wiki.amperka.ru/js:websocket.

СЕРВИС WEBHOOKS ДЛЯ IFTTT

```
var maker = require('@amperka/ifttt-webhooks').create({
  token: 'токен_сервиса_webhooks',
  action: 'название_действия_в_апплетах_IFTTT'
});
```
— подключить библиотеку для сервиса Webhooks и задать токен и действие.

```
maker.send(data, function(response) {
  print(response);
});
```
— отправить сервису Webhooks триггер события.

Подробнее на wiki.amperka.ru/js:ifttt-webhooks.

TELEGRAM-БОТ

```
var bot = require('@amperka/telegram').create({
  token: 'токен_сервиса_telegram',
  polling: { timeout: 10 }
});
```
— подключить библиотеку Telegram-бота с заданным токеном и временем удержания соединения при обновлении.

```
var keyboard = bot.keyboard([
  ['/TurnOff', '/TurnOn'],
], { resize: true });
```
— создать объект-клавиатуру.

```
bot.on('/start', function(msg) {
  bot.sendMessage(msg.from.id, 'Light control', { markup: keyboard });
});
```
— отправить клавиатуру собеседнику бота, когда тот нажмёт кнопку «START» в диалоге.

```
bot.on('/TurnOff', function(msg) {
  light.turnOff();
});
```
— выключить свет, когда собеседник нажмёт кнопку «/TurnOff».

`bot.sendMessage(msg.from.id, 'Light control', { markup: keyboard });` — отправить сообщение с текстом «Light control» и клавиатурой.

`bot.connect();` — запустить автоматическое получение обновлений.

Подробнее на wiki.amperka.ru/js:telegram.

Амперка

В компании Амперка надеемся мы, что понравился наш набор.

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

За порцией вдохновения к видеоканалу обращайся: youtube.com/AmperkaRU

Руководства и инструкции подробные ищи на wiki.amperka.ru

За платами новыми и модулями могучими в магазин специальный иди: amperka.ru

Электронная версия книги: iot.amperka.ru

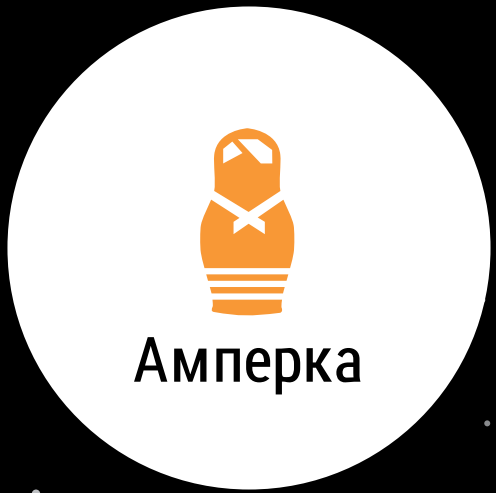
vk.com/amperkaru

facebook.com/amperka.ru

instagram.com/amperkaru

twitter.com/amperka

Дизайн: студия «Кластер»
clusterstudio.ru


Амперка