University of Minho
School of Engineering

ISLab

Ambient Intelligence

Autonomous Systems

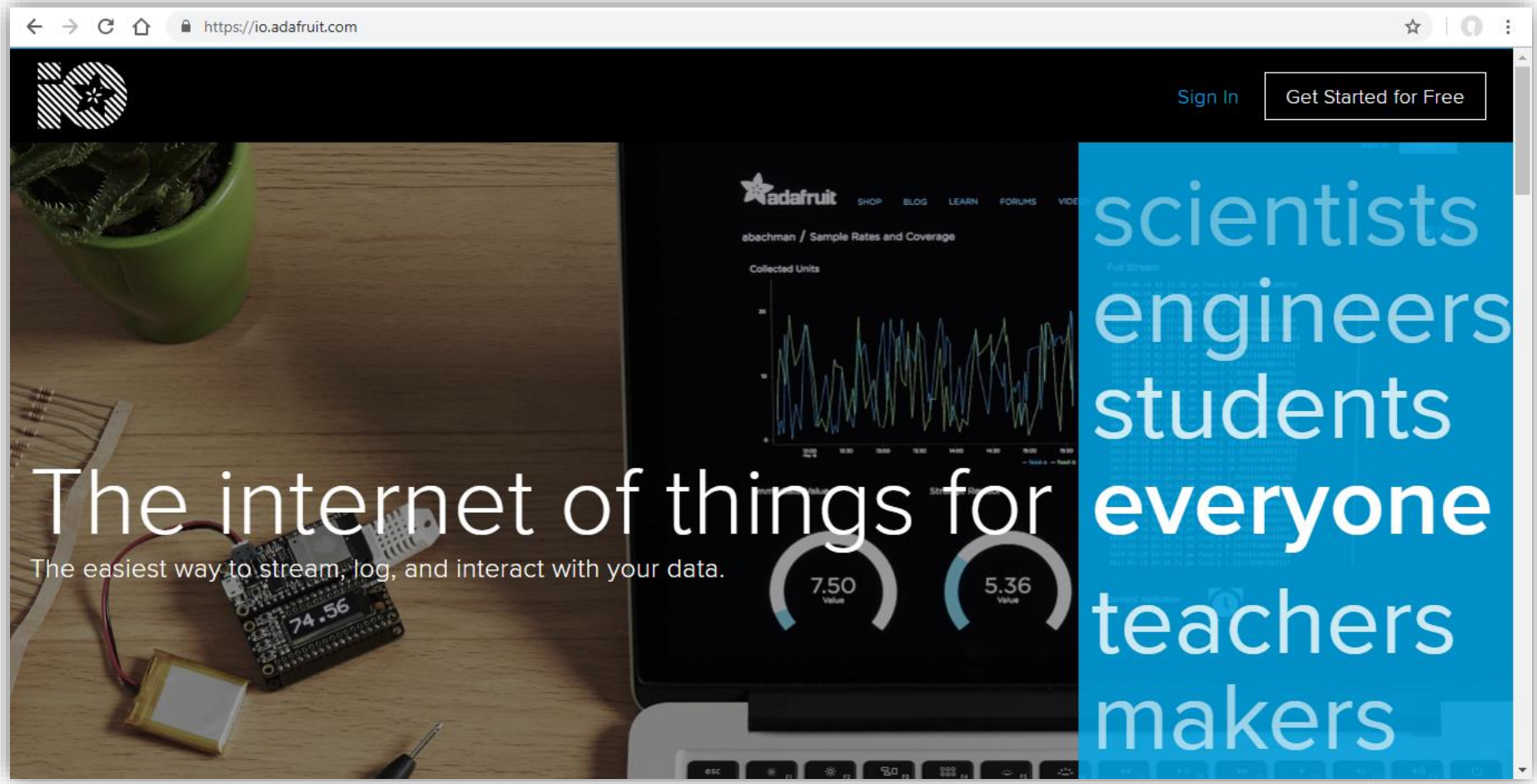
Perfil Sistemas Inteligentes @ MEI/MiEI 1º/4º - 2º semestre

Bruno Fernandes, Cesar Analide, Fábio Silva

18/03/2019

Contents

Adafruit IO


https://io.adafruit.com
Sign In
Get Started for Free
adafruit
SHOP
BLOG
LEARN
FORUMS
abachman / Sample Rates and Coverage
Collected Units
7.50
5.36
The internet of things for everyone
The easiest way to stream, log, and interact with your data.
scientists
engineers
students
teachers
makers

Adafruit IO

Adafruit IO

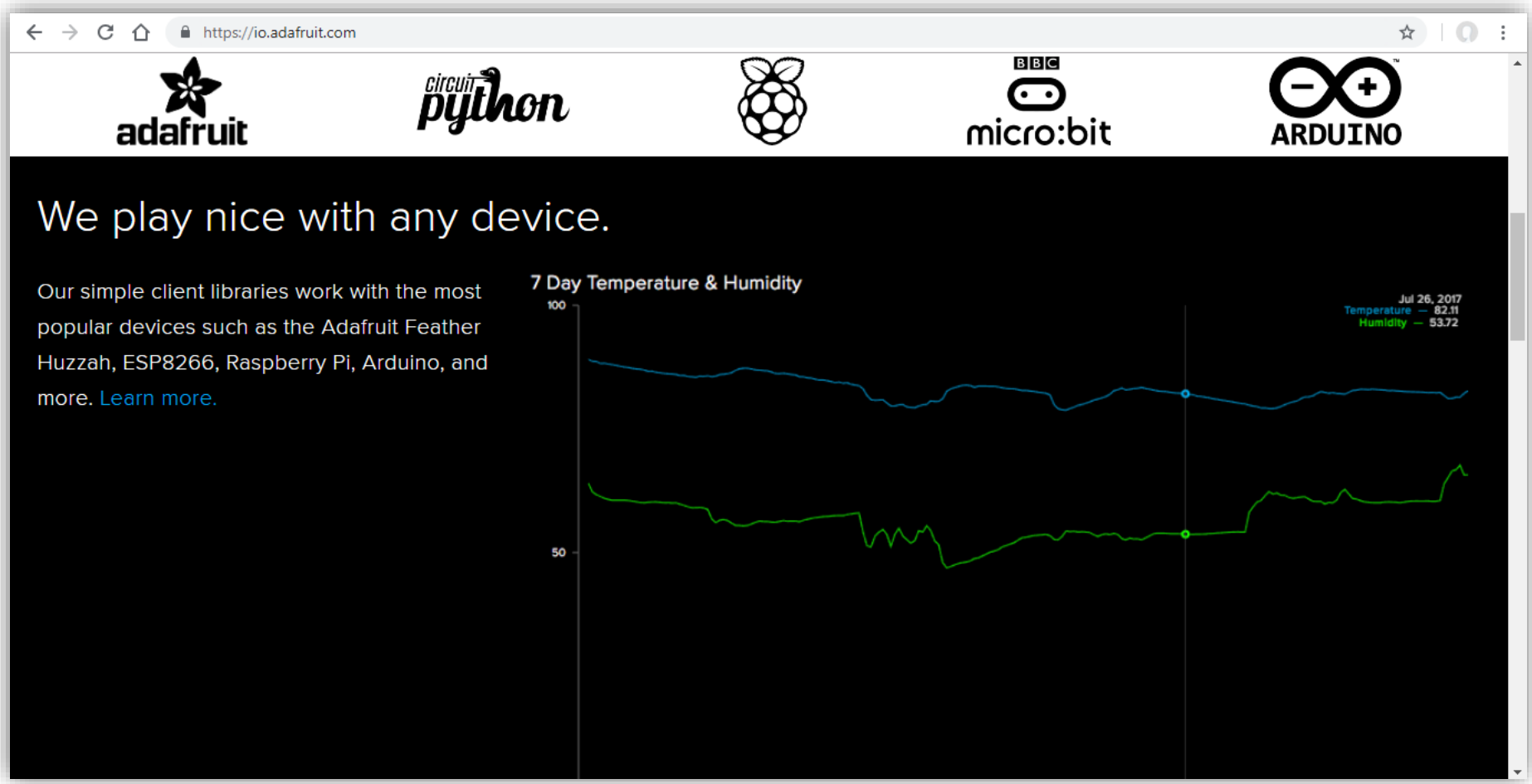
- Adafruit.io is a cloud service
- It's meant primarily for storing and then retrieving data (using feeds)
  - It can also display our data in real-time
- Feeds are the core of Adafruit IO holding both the uploaded data (data our sensors push to Adafruit IO) and its meta-data

Adafruit IO

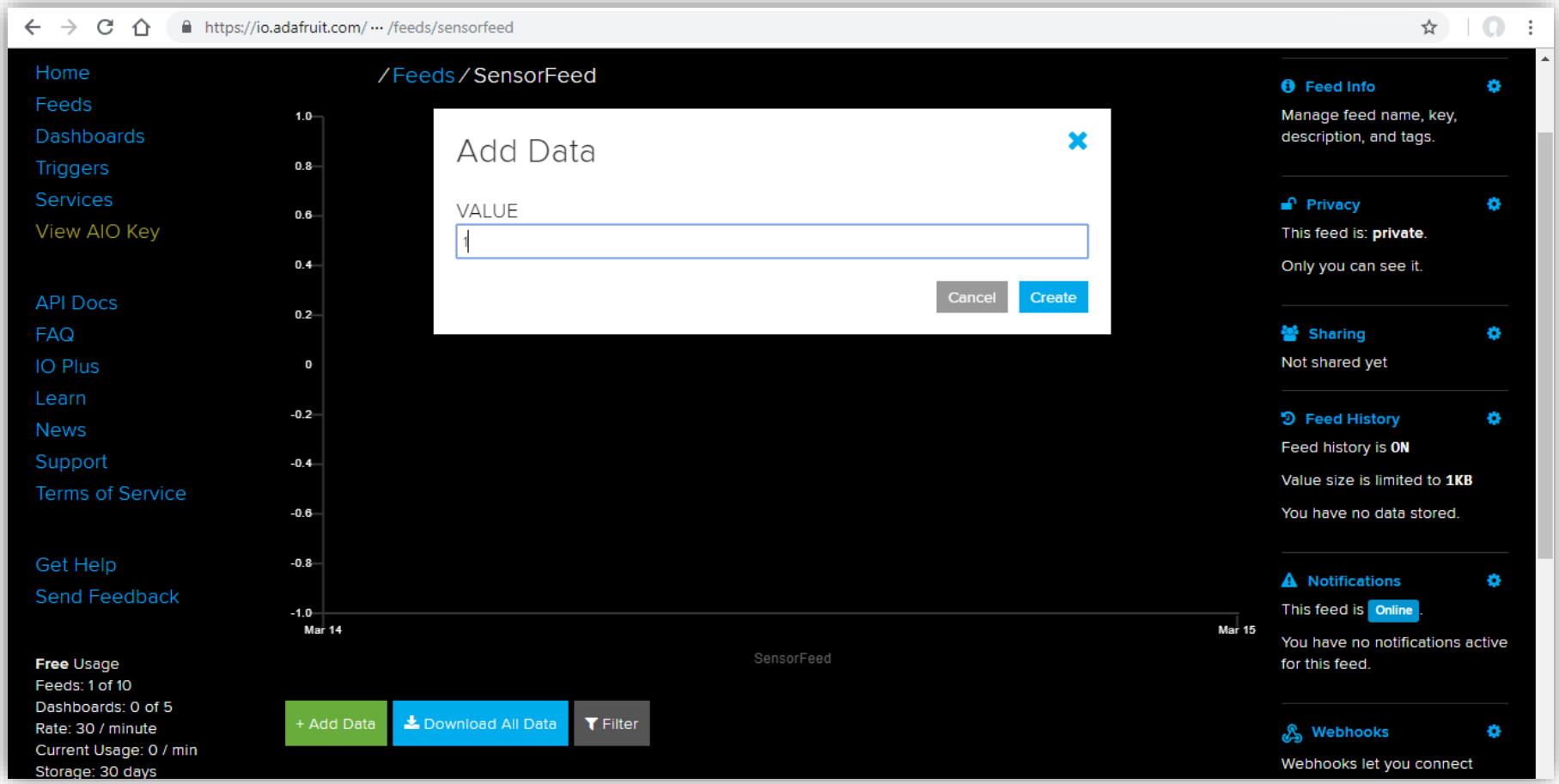
Adafruit IO

- Nice and intuitive dashboards integrated into Adafruit IO
- Allows us to define triggers to control and react to our data (ex.: triggers to email us when a temperature sensor gets too hot, for example)
- Integration with IFTTT

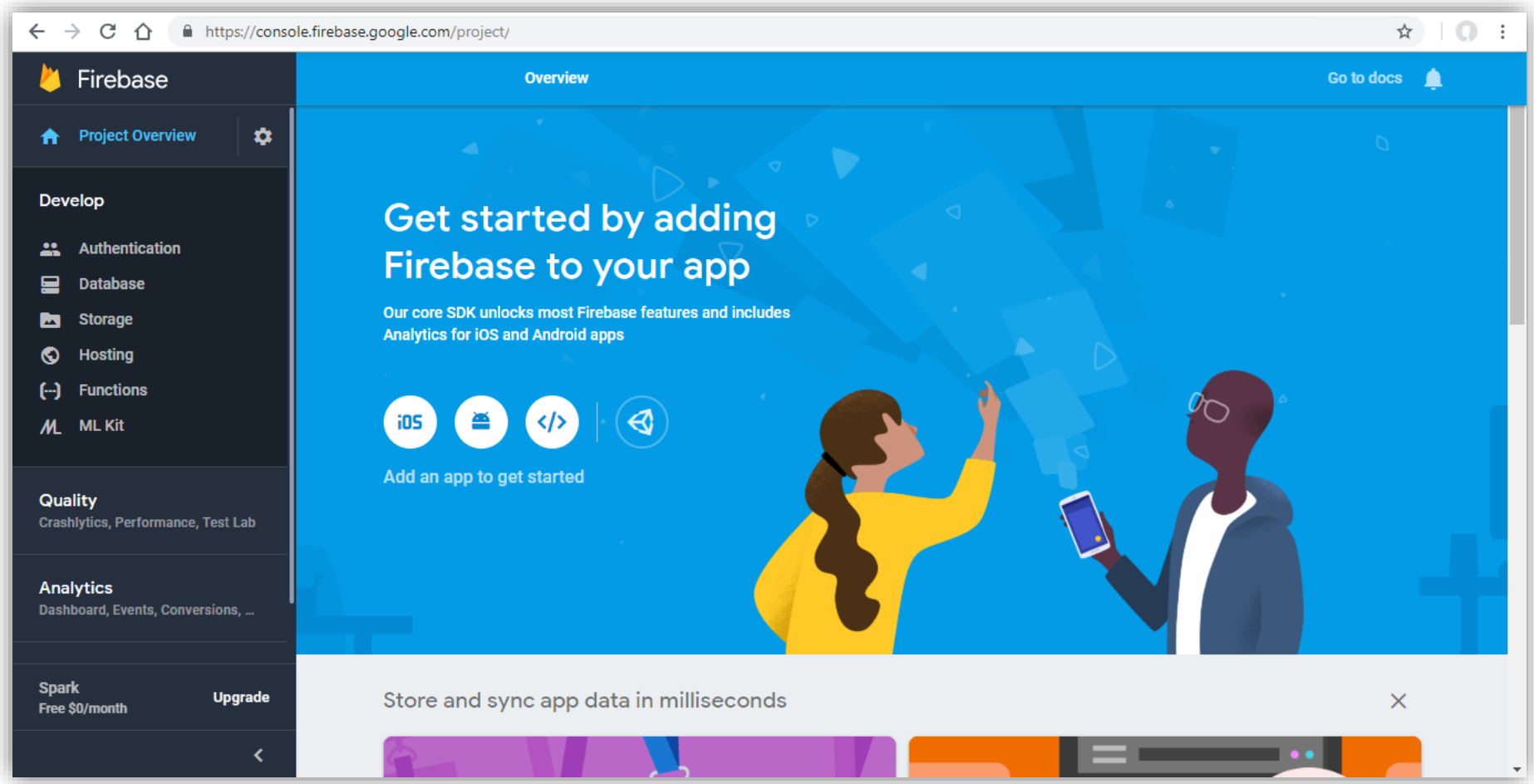
Adafruit IO


https://io.adafruit.com
adafruit
circuit python
micro:bit
ARDUINO
We play nice with any device.
Our simple client libraries work with the most popular devices such as the Adafruit Feather Huzzah, ESP8266, Raspberry Pi, Arduino, and more. Learn more.
7 Day Temperature & Humidity
100
50
Jul 26, 2017
Temperature — 82.11
Humidity — 53.72

Adafruit IO


https://io.adafruit.com/ ··· /feeds/sensorfeed
Home
Feeds
Dashboards
Triggers
Services
View AIO Key
API Docs
FAQ
IO Plus
Learn
News
Support
Terms of Service
Get Help
Send Feedback
/Feeds/SensorFeed
Add Data
VALUE
1
Cancel
Create
1.0
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1.0
Mar 14
Mar 15
SensorFeed
Free Usage
Feeds: 1 of 10
Dashboards: 0 of 5
Rate: 30 / minute
Current Usage: 0 / min
Storage: 30 days
+ Add Data
Download All Data
Filter
Feed Info
Manage feed name, key, description, and tags.
Privacy
This feed is: private.
Only you can see it.
Sharing
Not shared yet
Feed History
Feed history is ON
Value size is limited to 1KB
You have no data stored.
Notifications
This feed is Online.
You have no notifications active for this feed.
Webhooks
Webhooks let you connect

Firebase


https://console.firebase.google.com/project/
Firebase
Overview
Go to docs
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab
Analytics
Dashboard, Events, Conversions, ...
Spark
Free $0/month
Upgrade
Get started by adding Firebase to your app
Our core SDK unlocks most Firebase features and includes Analytics for iOS and Android apps
iOS
Add an app to get started
Store and sync app data in milliseconds

Firebase

Firebase

- All the tools we need to build a successful app (said by them, not me)!
  - For example, they can handle all the authentication process by us (it can take up to months to set up our own authentication system)
- Handles backend, database, user engagement, scalability, …
- It is a mobile and web app development platform that provides developers with a set of tools and services to help them develop high-quality apps

Firebase

Firebase Database

- Firebase Realtime Database is a cloud-hosted NoSQL database that lets us store and sync our sensors data in realtime
  - it is really just one big JSON object that the developers can manage in realtime

- Cloud Firestore (yet another database) takes a more structured approach
  - It has just gone officially out of beta and in to General Availability
  - Faster queries and performance than the realtime database
  - It is now the new main Firebase database

Firebase

https://console.firebase.google.com/project/

Firebase

Project Overview

Develop

Authentication

Database

Storage

Hosting

Functions

ML Kit

Quality
Crashlytics, Performance, Test Lab

Analytics
Dashboard, Events, Conversions, ...

Spark
Free $0/month

Upgrade

Go to docs

Database

Realtime Database

Data Rules Backups Usage

https:// .firebaseio.com/

ProbeData1

ProbeData10

probes

0

mac: "da:a1:19:67:08:6

previousMillisDetected: 45587

rssi: "-96"

1

mac: "e8:93:09:03:0a:3

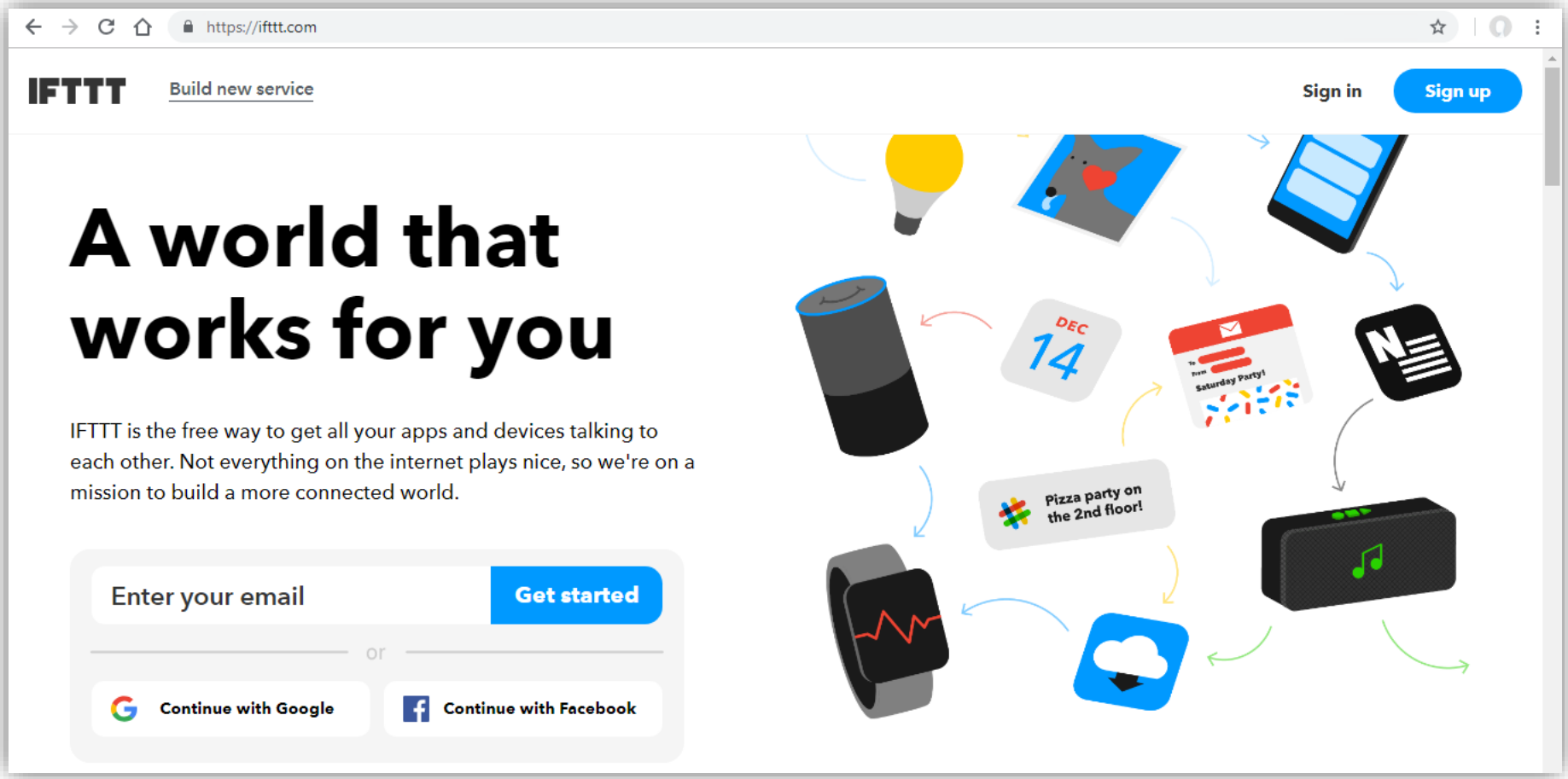
previousMillisDetected: 45623

rssi: "-83"

2

mac: "7c:2e:dd:c1:c1:4

IFTTT - If This Then That

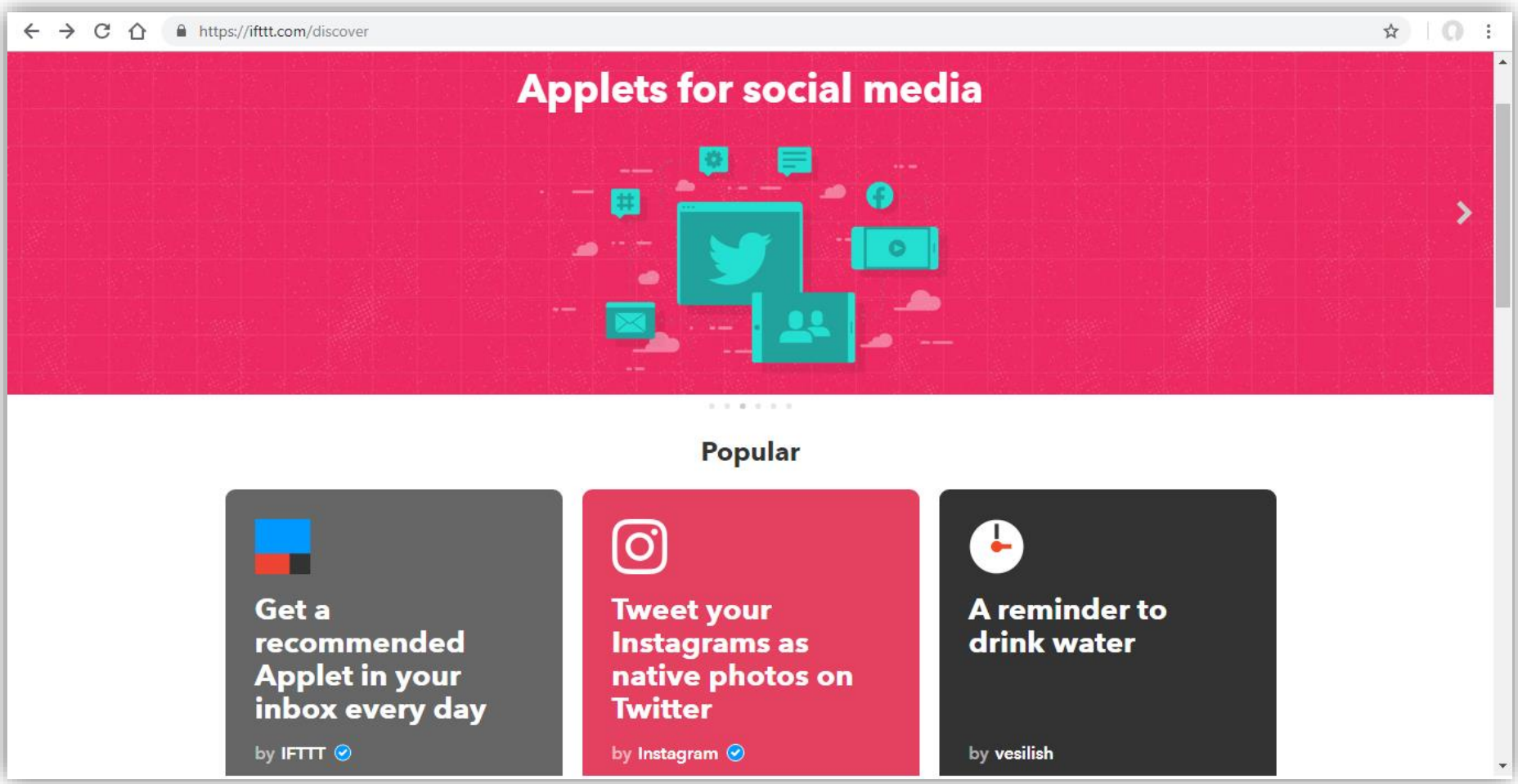


IFTTT

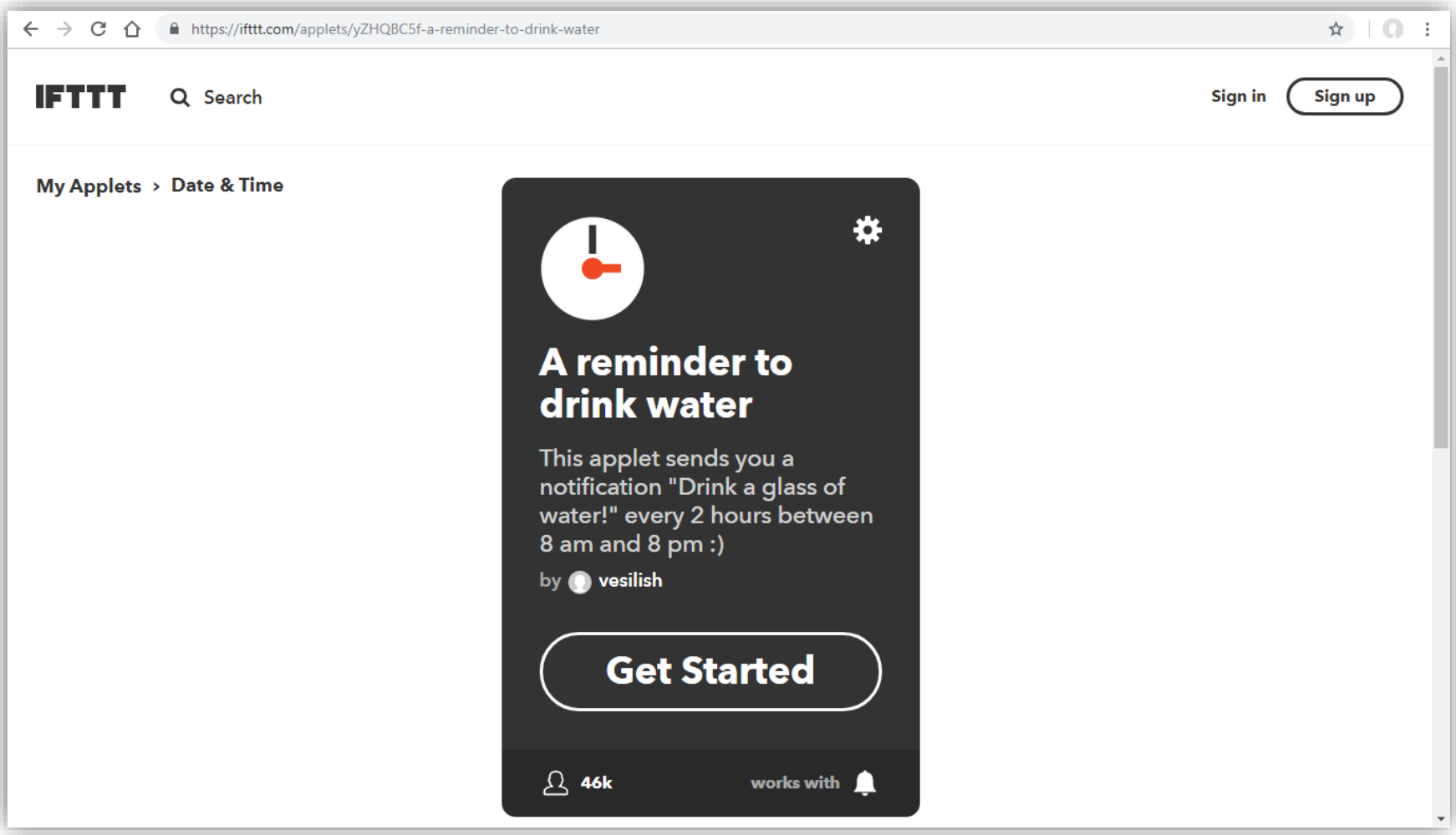
IFTTT - If This Then That

- A service to create chains of conditional statements, called applets
- Is triggered by changes that occur within other web services
- After triggered, it executes an actionable service in the platform
- Besides the web-based application, it runs on iOS and Android.

IFTTT


https://ifttt.com/discover
Applets for social media
Popular
Get a recommended Applet in your inbox every day
by IFTTT
Tweet your Instagrams as native photos on Twitter
by Instagram
A reminder to drink water
by vesilish

IFTTT

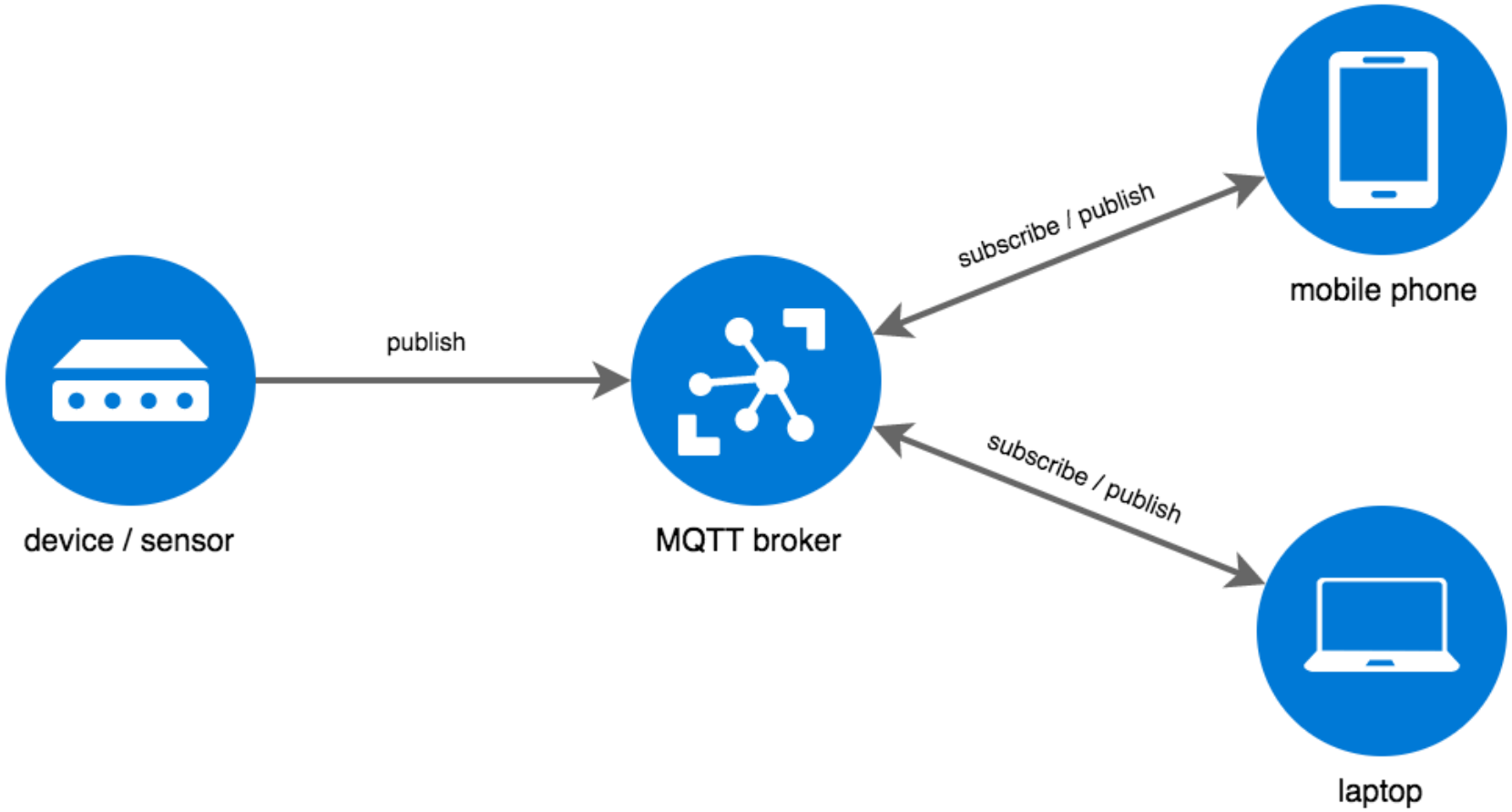

https://ifttt.com/applets/yZHQBC5f-a-reminder-to-drink-water
IFTTT
Search
Sign in
Sign up
My Applets › Date & Time
A reminder to drink water
This applet sends you a notification "Drink a glass of water!" every 2 hours between 8 am and 8 pm :)
by vesilish
Get Started
46k
works with

MQTT

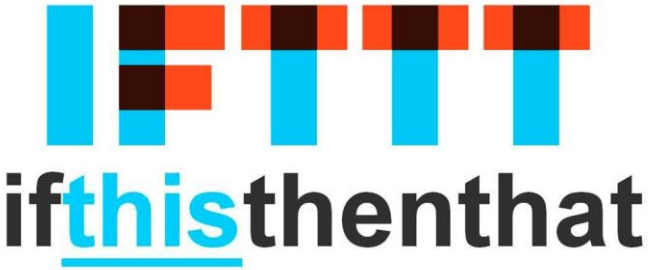
MQTT (Message Queue Telemetry Transport) Protocol

- MQTT is a Machine-to-Machine/Internet of Things connectivity protocol
  - MQTT was originally developed out of IBM's pervasive computing team
- It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency networks
- MQTT messages are sent to feeds in an MQTT broker (such as Adafruit IO), which then distributes them through the devices that subscribed those feeds
- Lightweight message protocol
  - Connecting to a server only takes about 80 bytes
  - Push data from server to device is about 20 bytes

MQTT


publish
subscribe / publish
subscribe / publish
device / sensor
MQTT broker
mobile phone
laptop

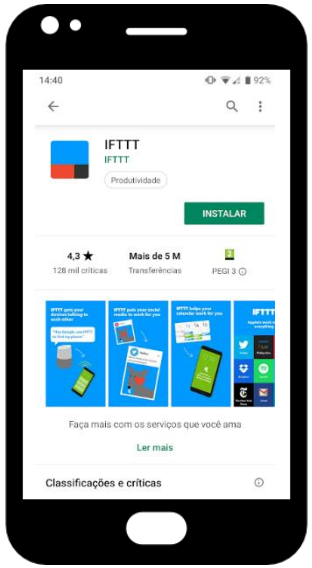
Adafruit IO + IFTTT



Adafruit IO + IFTTT

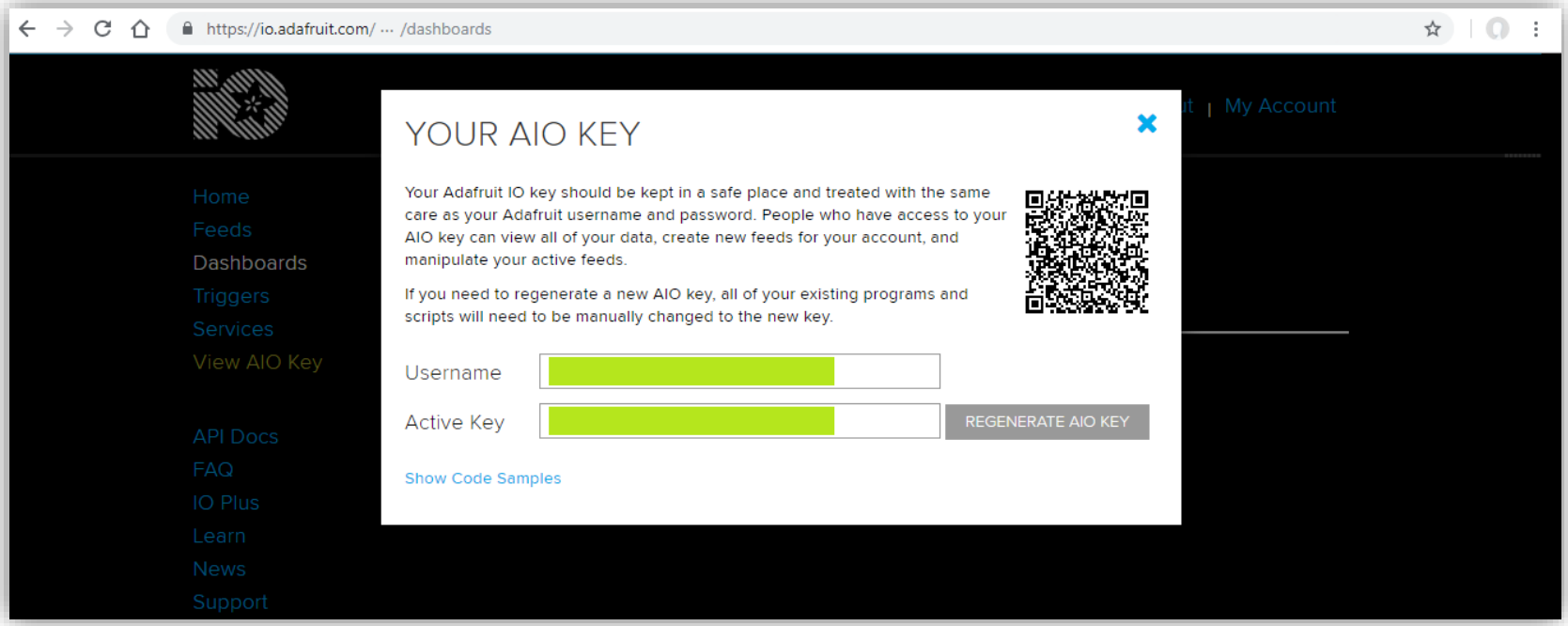
In this How To lets:

- Develop an IFTTT applet that reacts to values in Adafruit IO feeds
- The goal is to monitor sensor values sent to Adafruit IO feeds and take actions in some particular context
- To complete this How To you need:
  - An IFTTT account
  - An Adafruit IO account
  - A smartphone with the IFTTT application installed



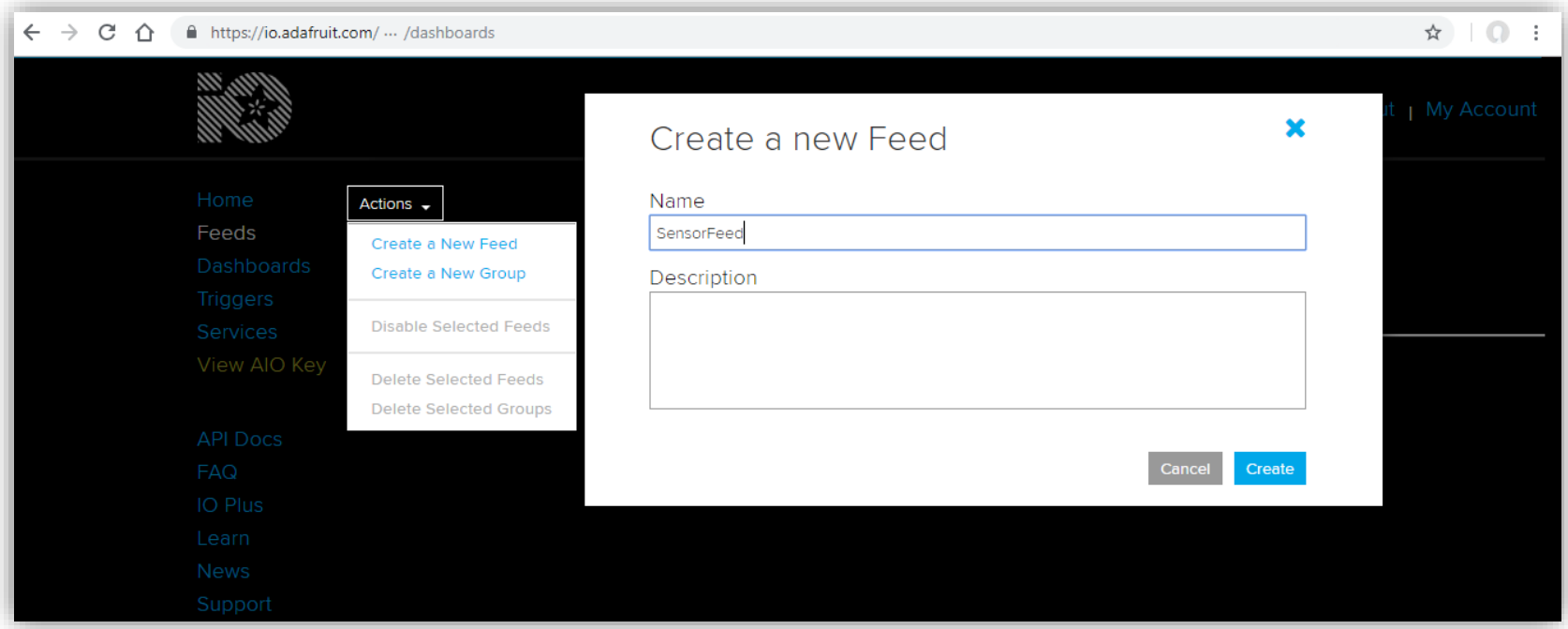
Adafruit IO + IFTTT

Take note of your username and AIO Key



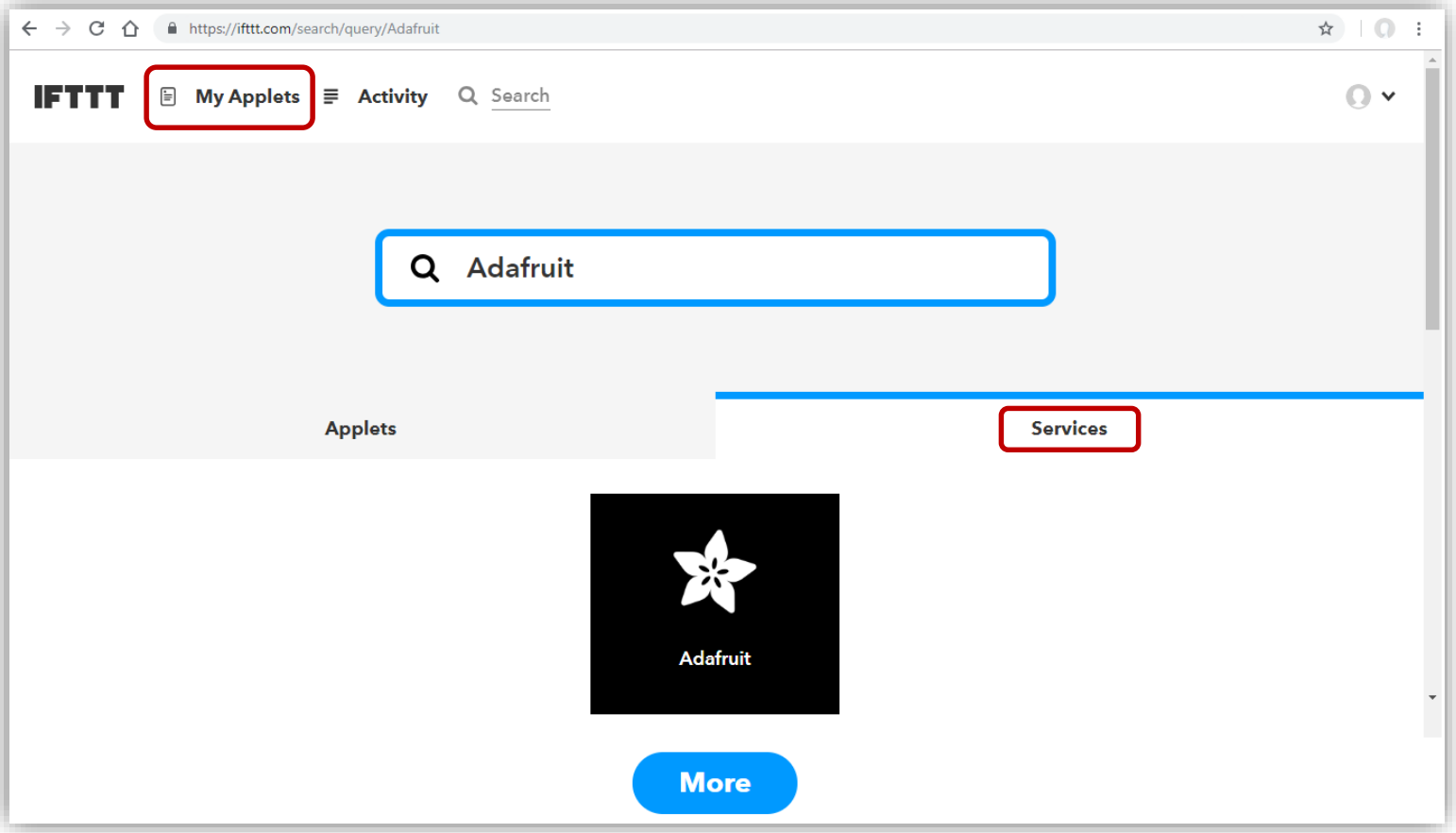
Adafruit IO + IFTTT

Create a new feed named SensorFeed in Adafruit IO



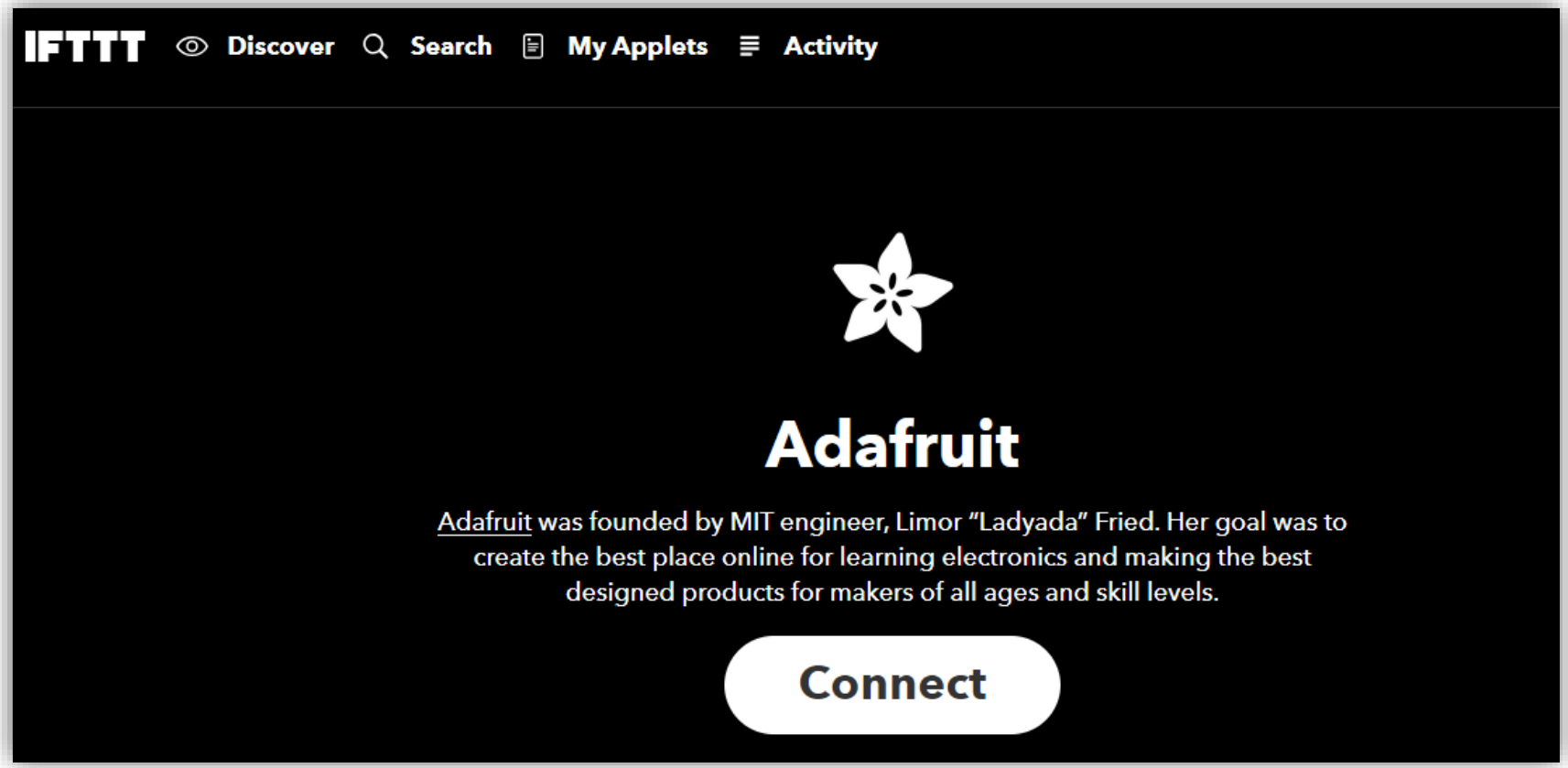
Adafruit IO + IFTTT

Login into IFTTT and search for the Adafruit service



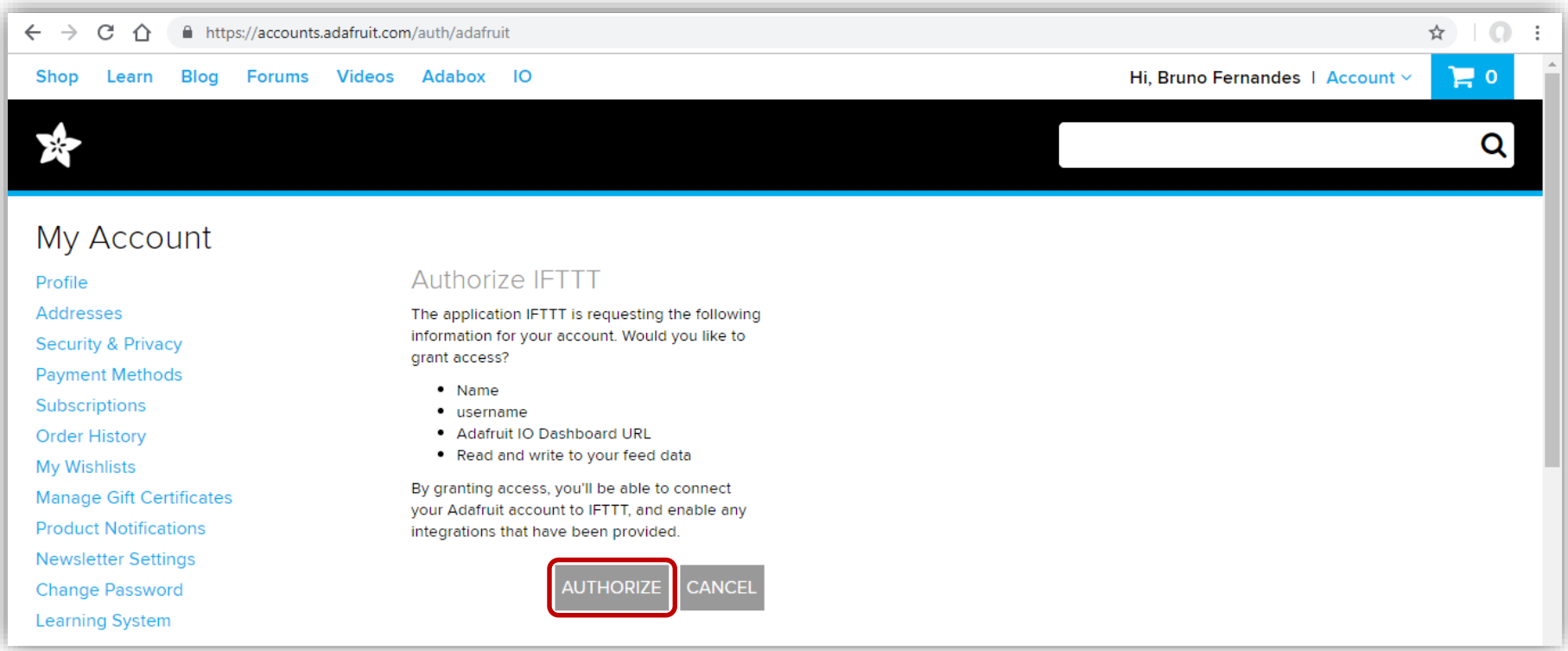
Adafruit IO + IFTTT

Connect IFTTT to the Adafruit platform



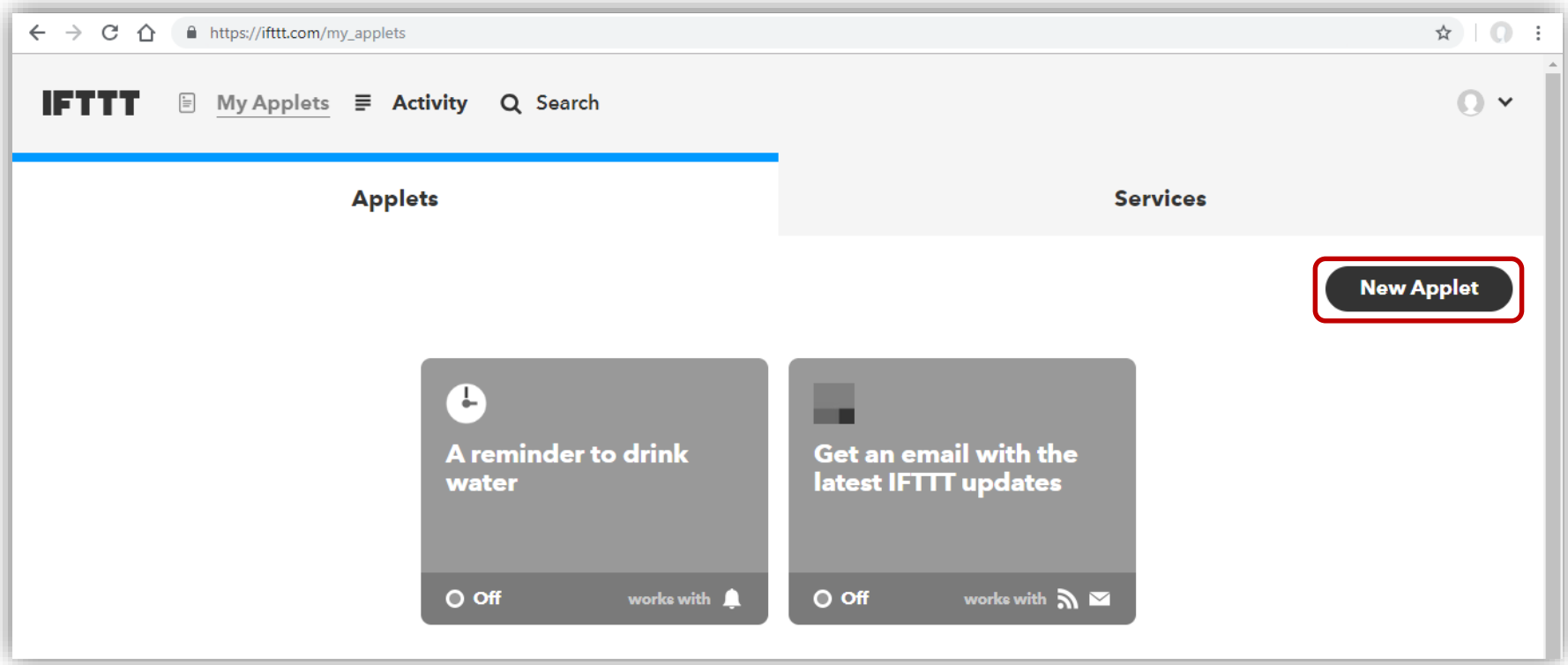
Adafruit IO + IFTTT

Authorize IFTTT access



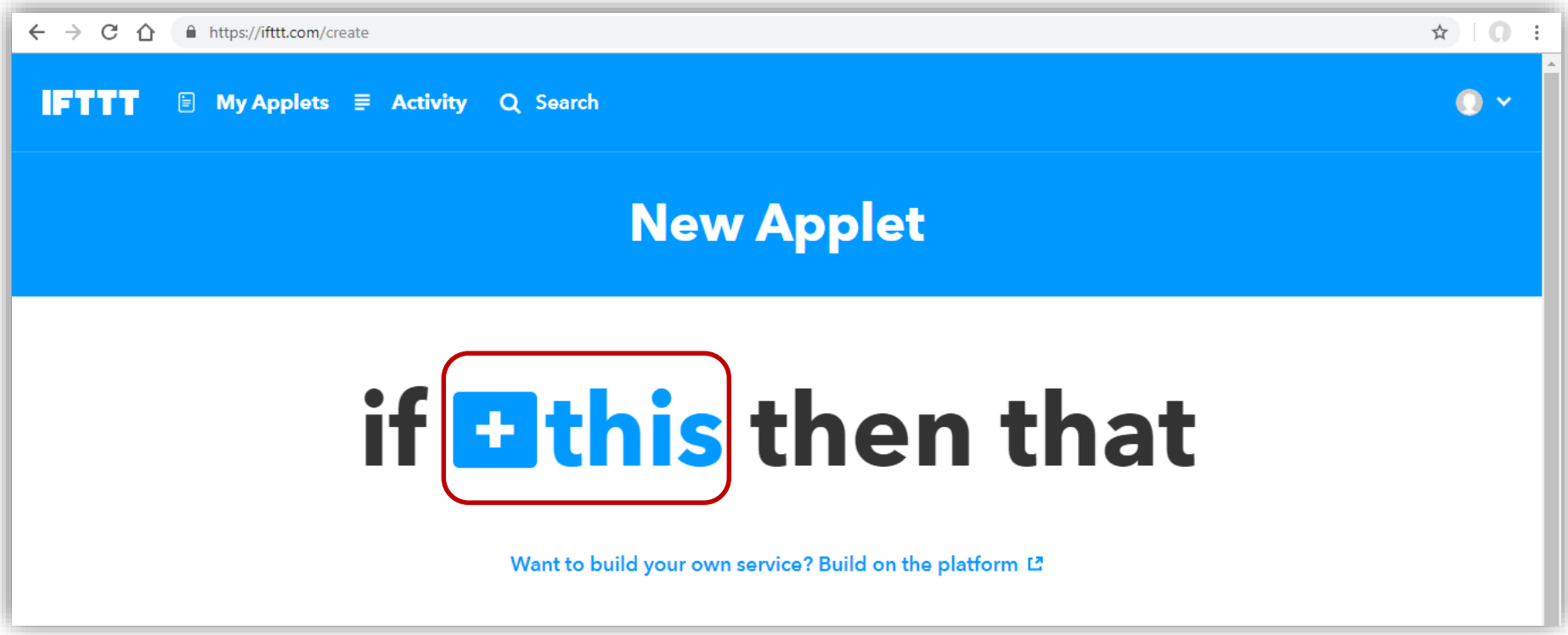
Adafruit IO + IFTTT

On the IFTTT platform lets create a new applet



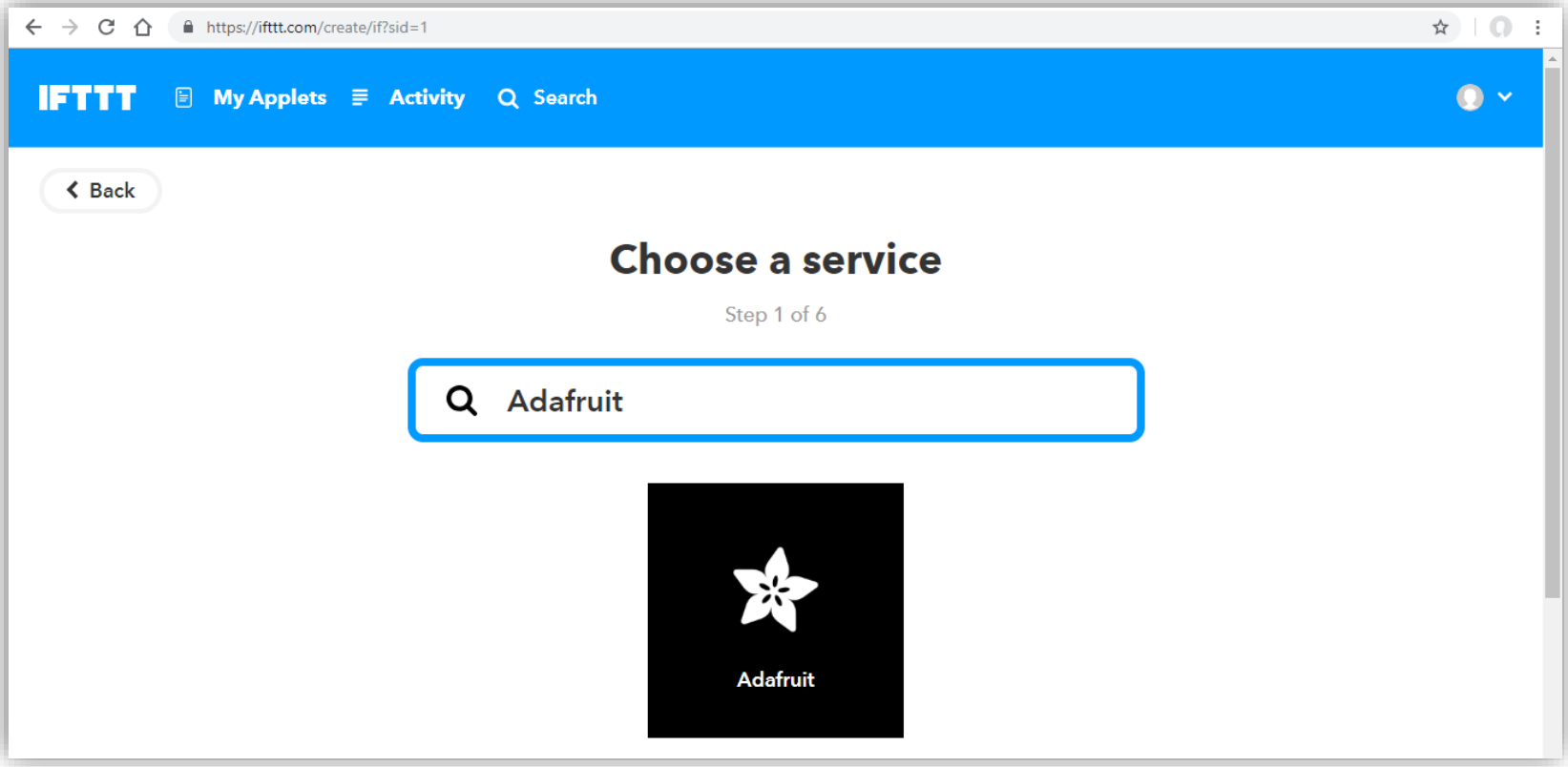
Adafruit IO + IFTTT

On the IFTTT platform lets create a new applet (If This)



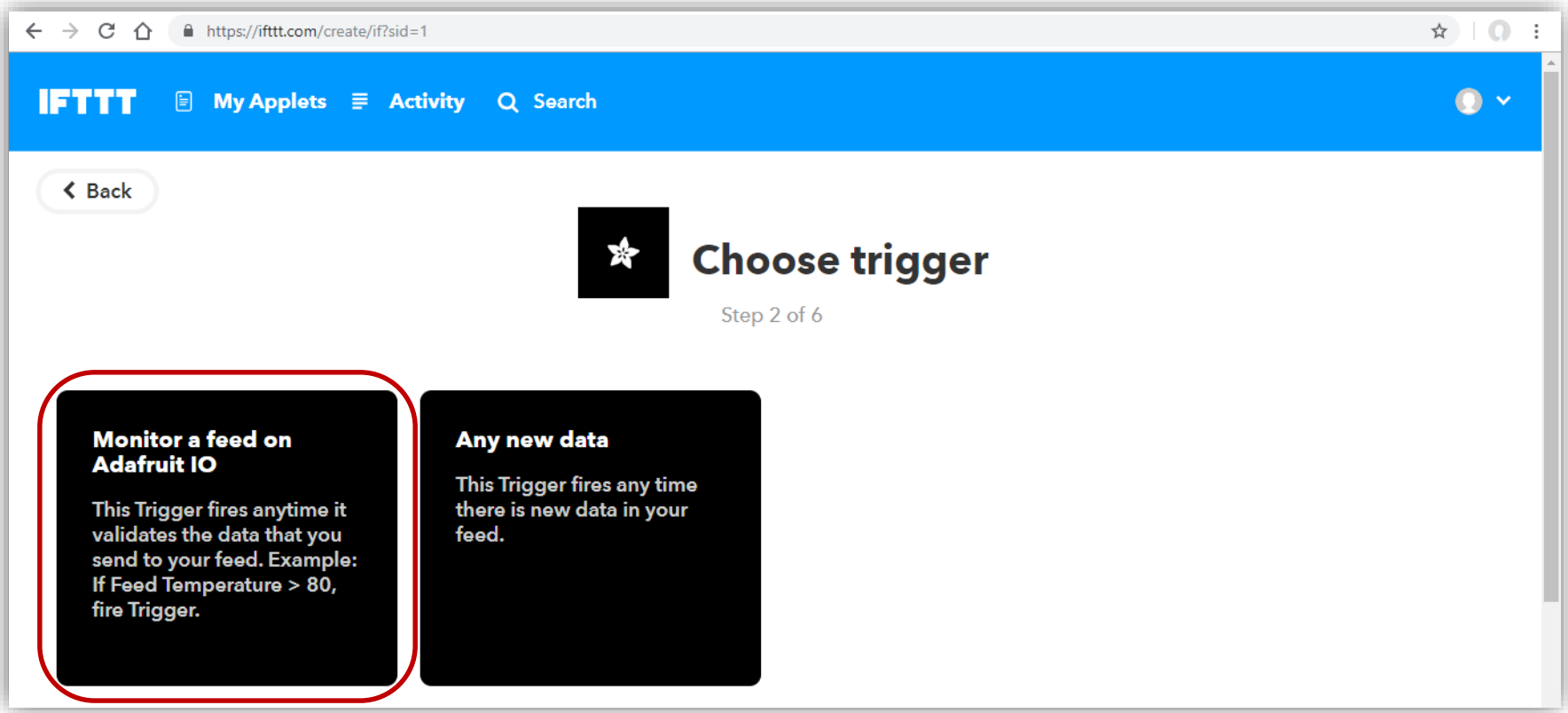
Adafruit IO + IFTTT

Search for the Adafruit service



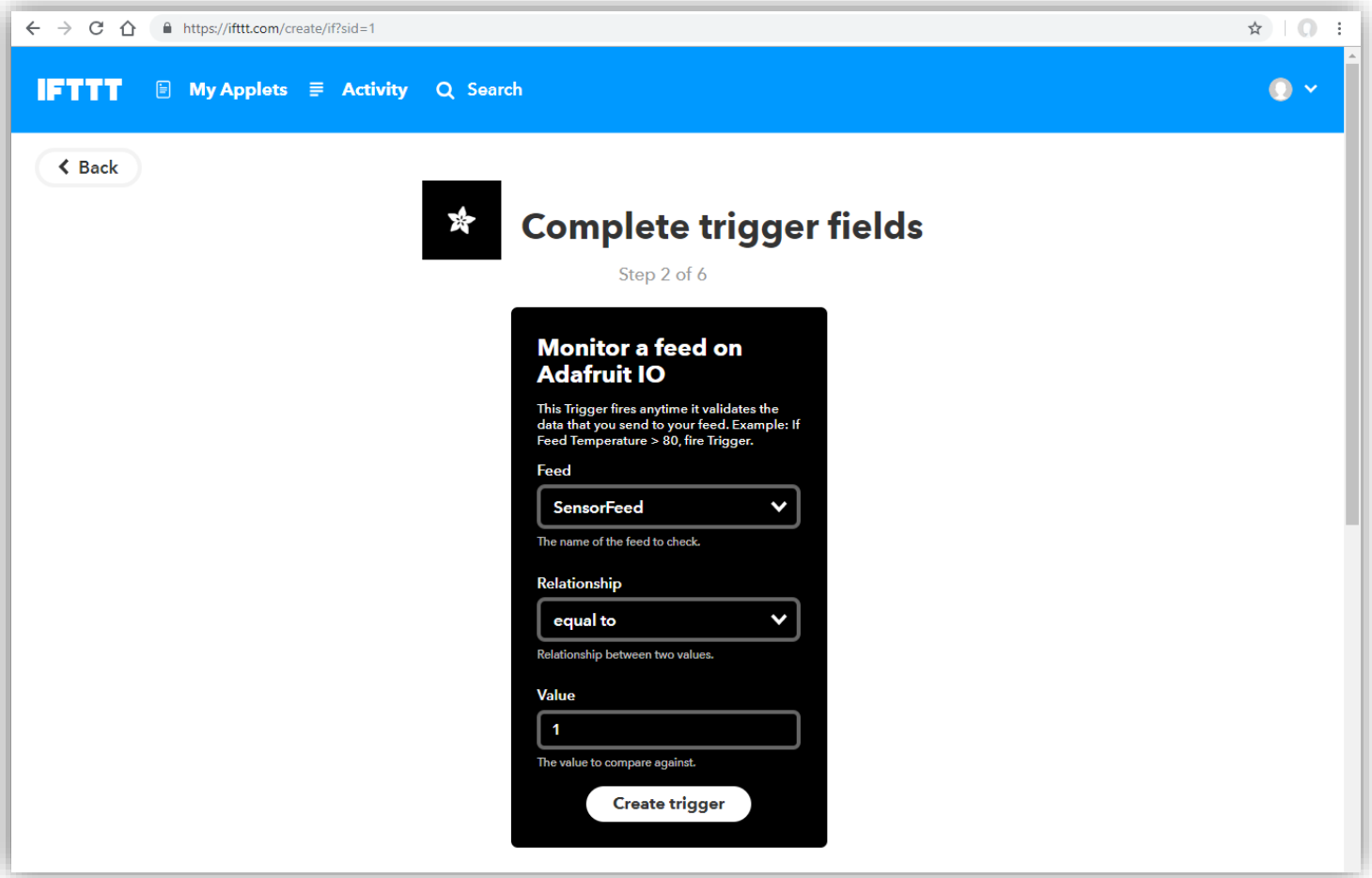
Adafruit IO + IFTTT

Let us monitor a feed and fire a trigger if some condition is met



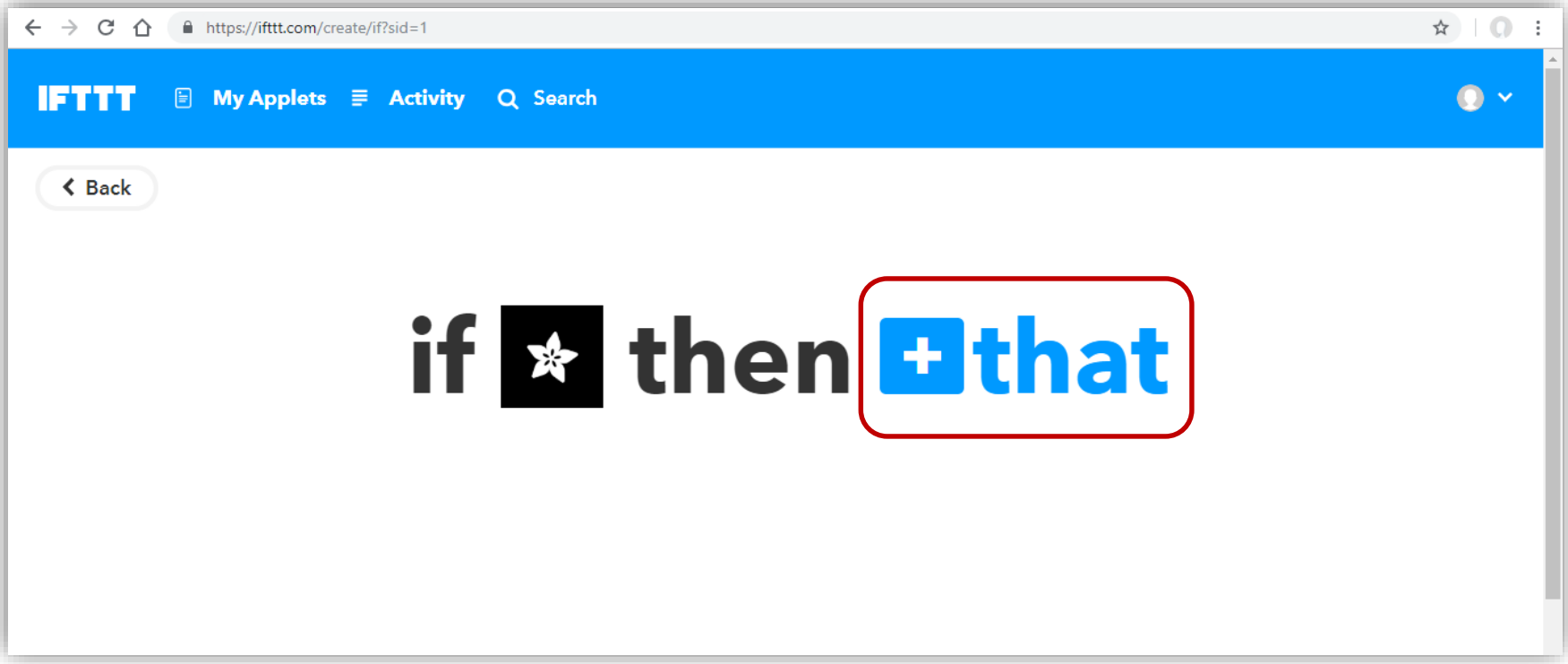
Adafruit IO + IFTTT

Let us monitor SensorFeed and fire a trigger if the value 1 is sent to the feed



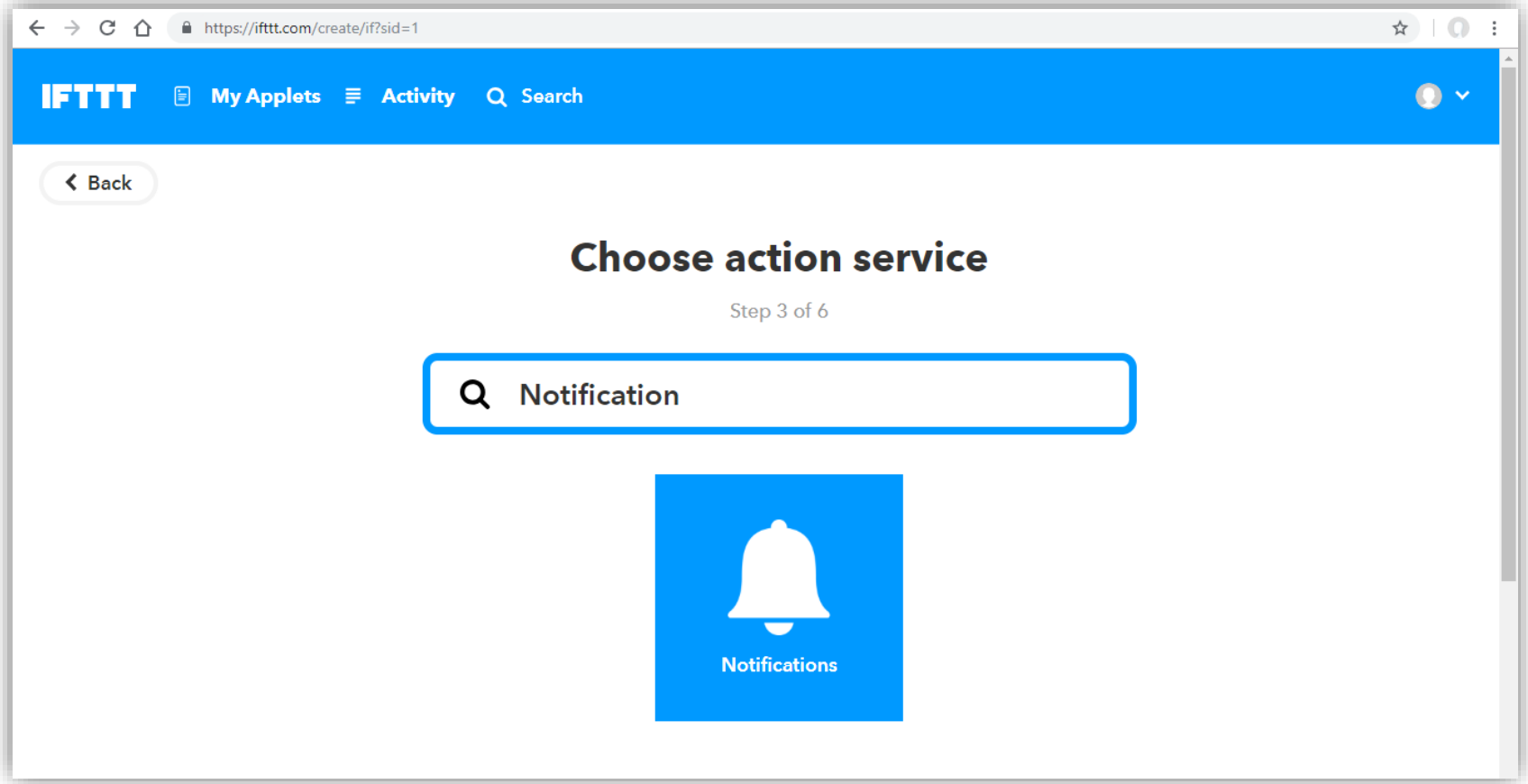
Adafruit IO + IFTTT

If … Then That



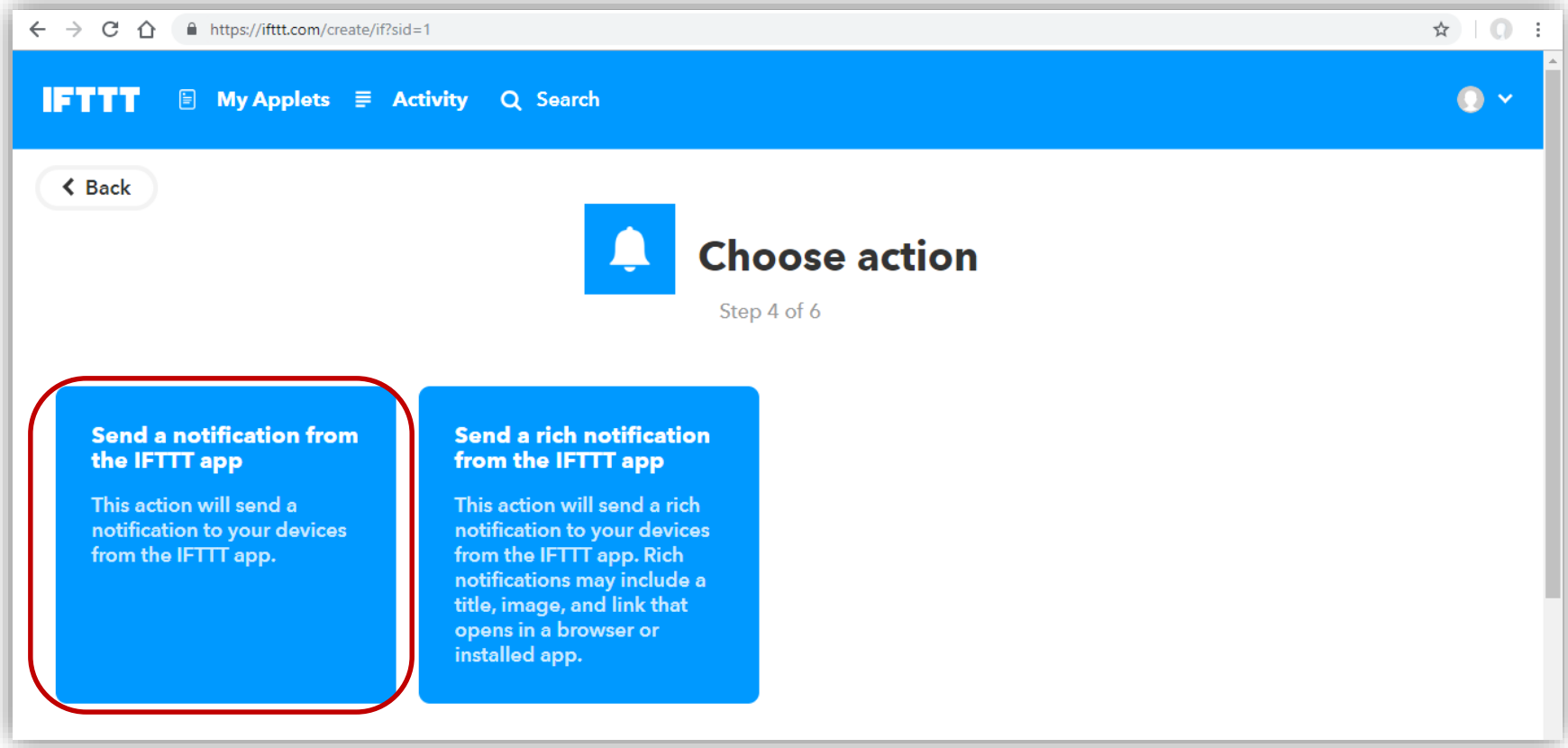
Adafruit IO + IFTTT

Search for the Notifications service



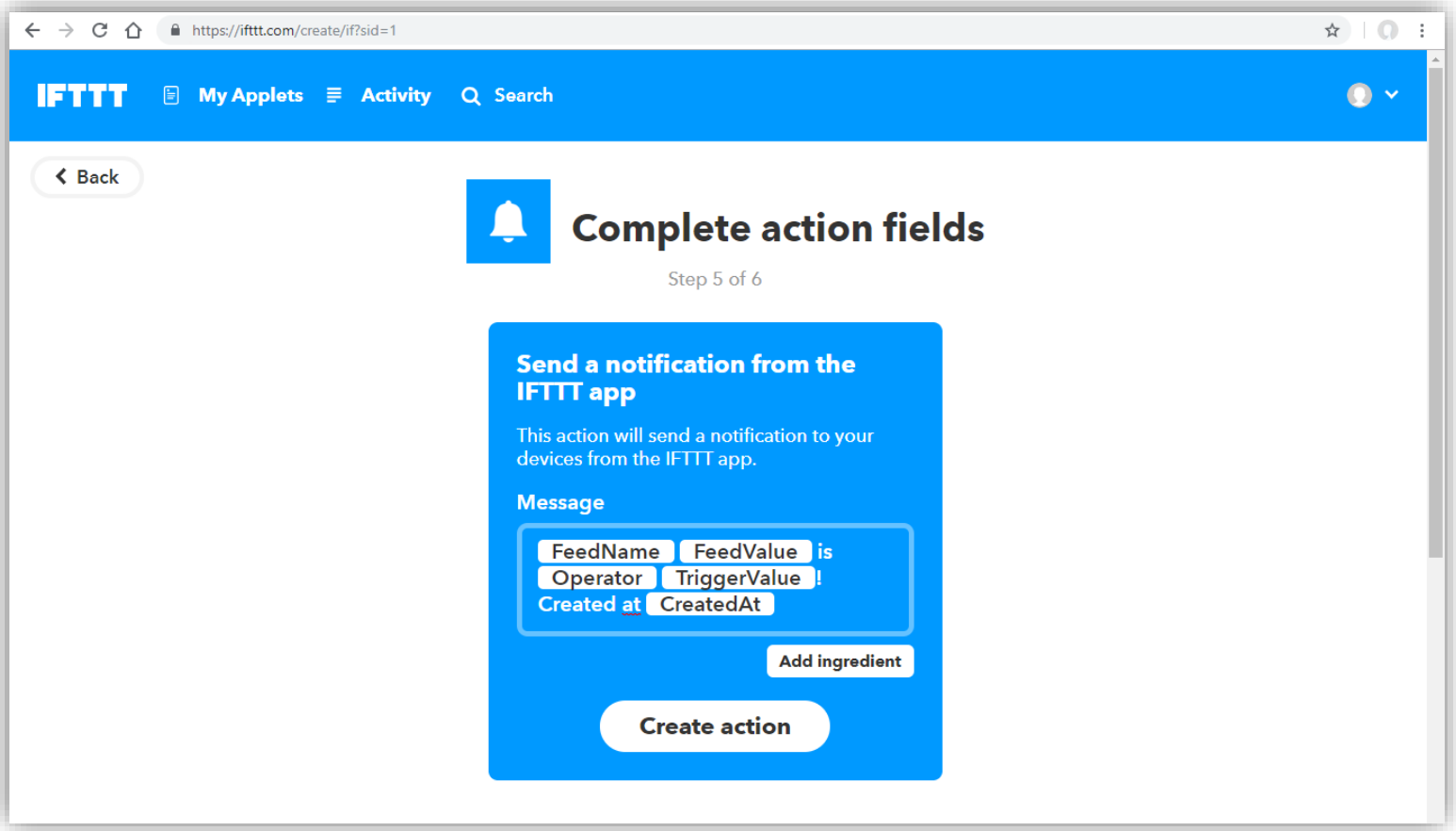
Adafruit IO + IFTTT

Let us send a notification from the IFTTT app



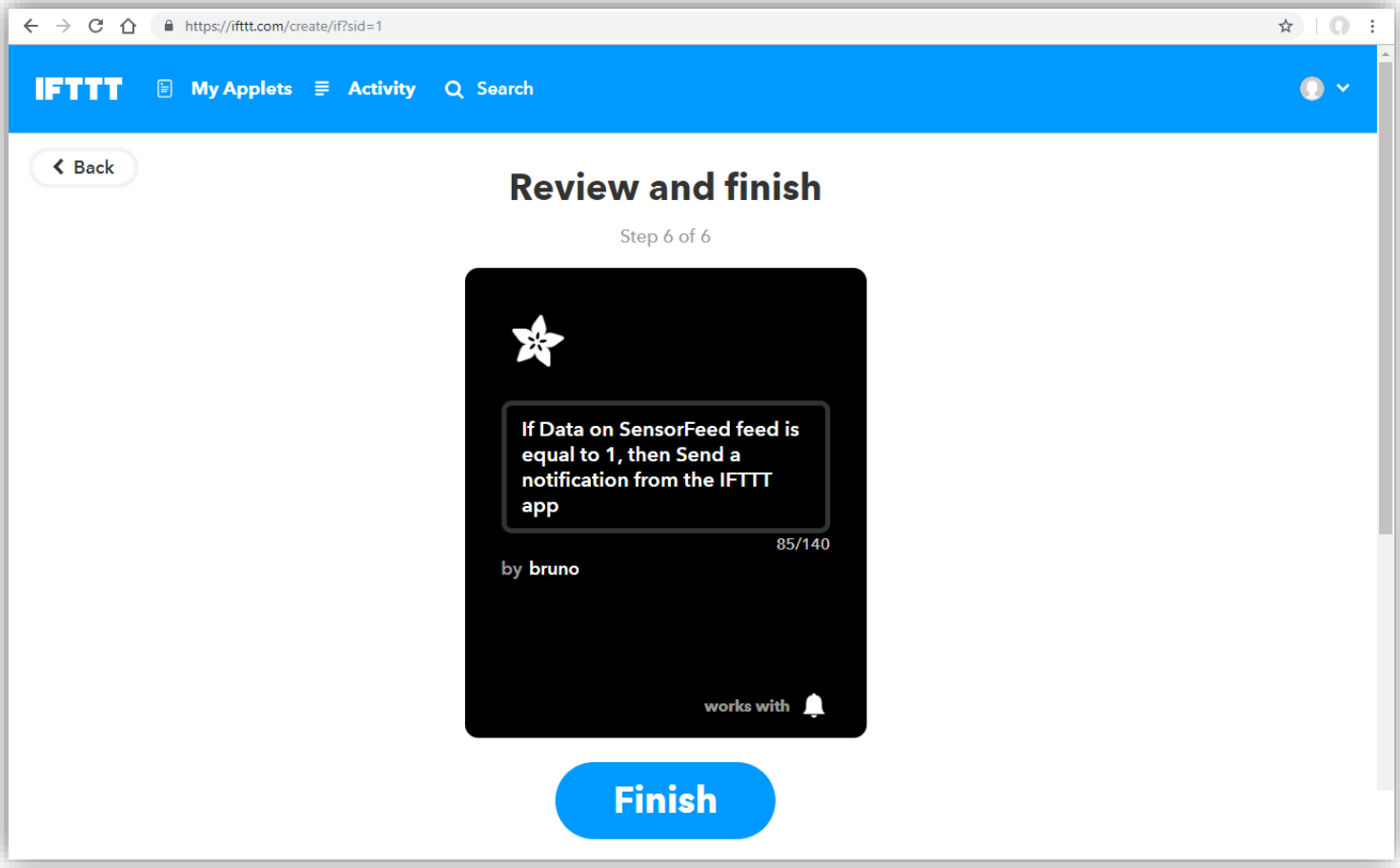
Adafruit IO + IFTTT

Create a custom notification to display in the device where the IFTTT app is installed. You can obtain values from the feed by adding ingredients!



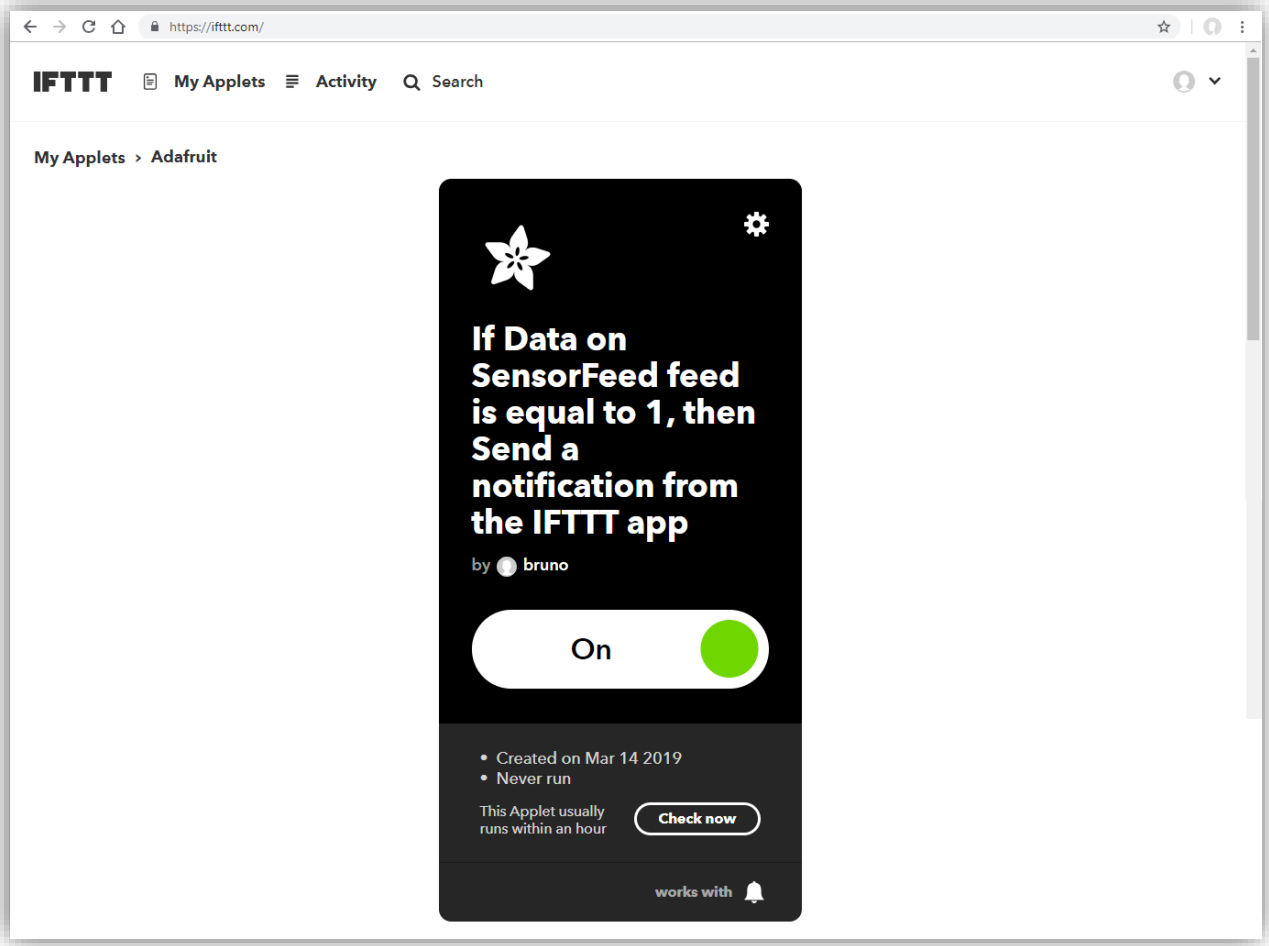
Adafruit IO + IFTTT

Review the created rule and confirm it!



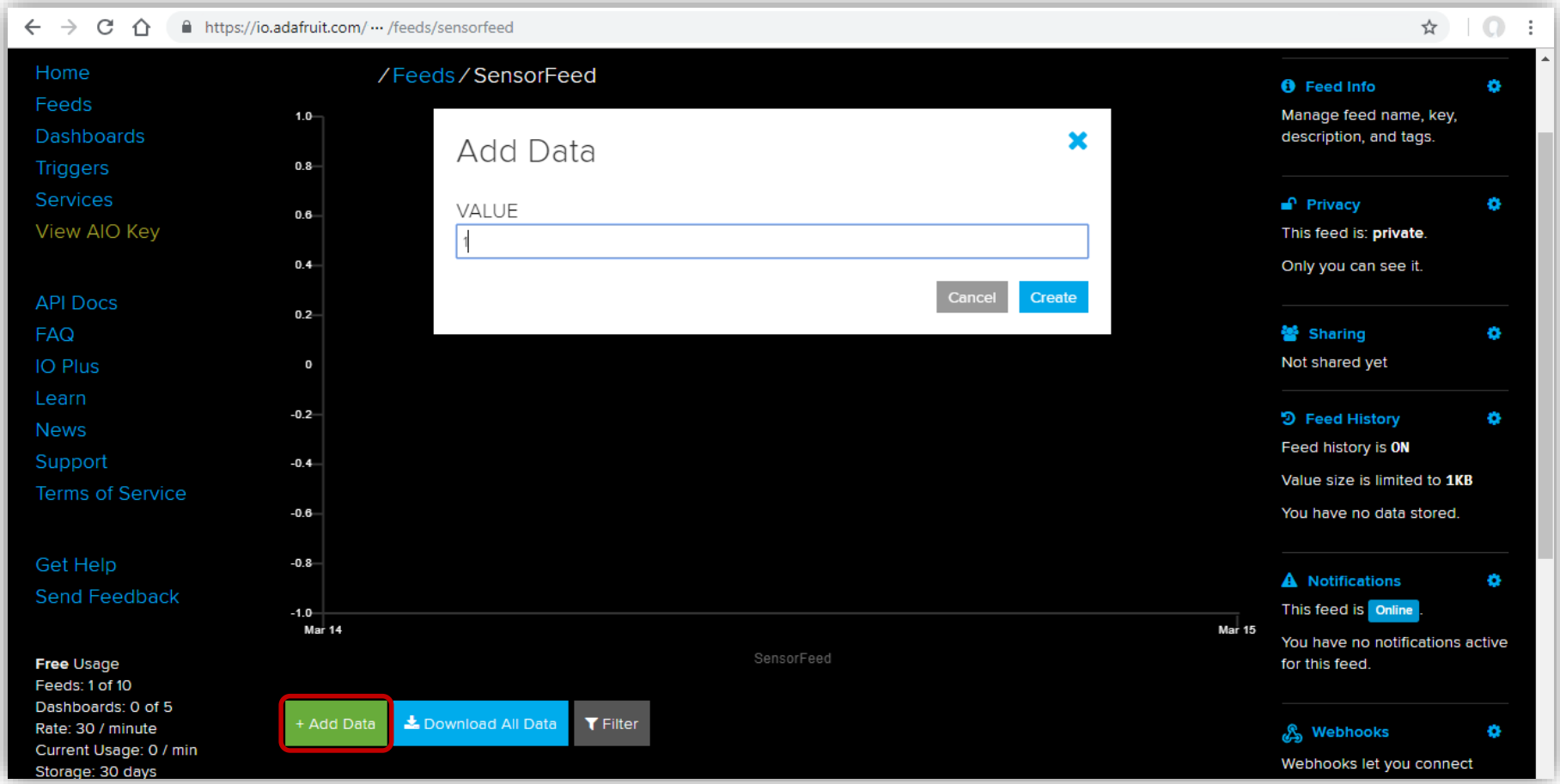
Adafruit IO + IFTTT

Activate it!



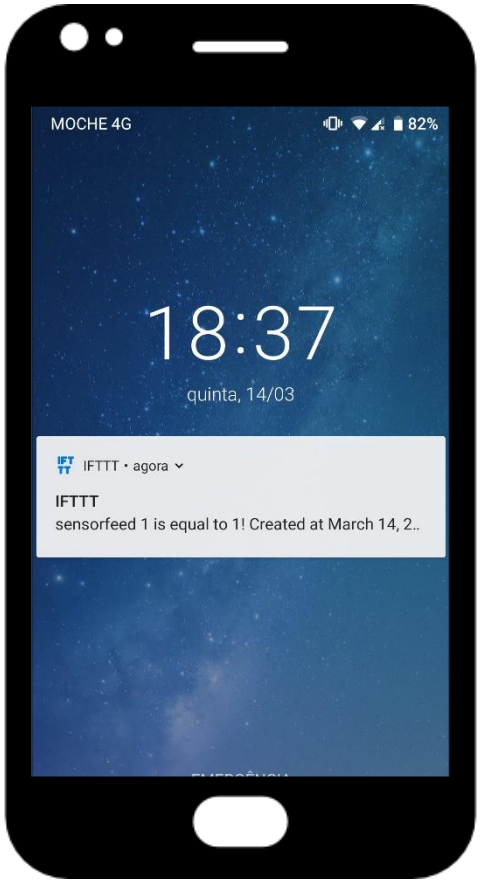
Adafruit IO + IFTTT

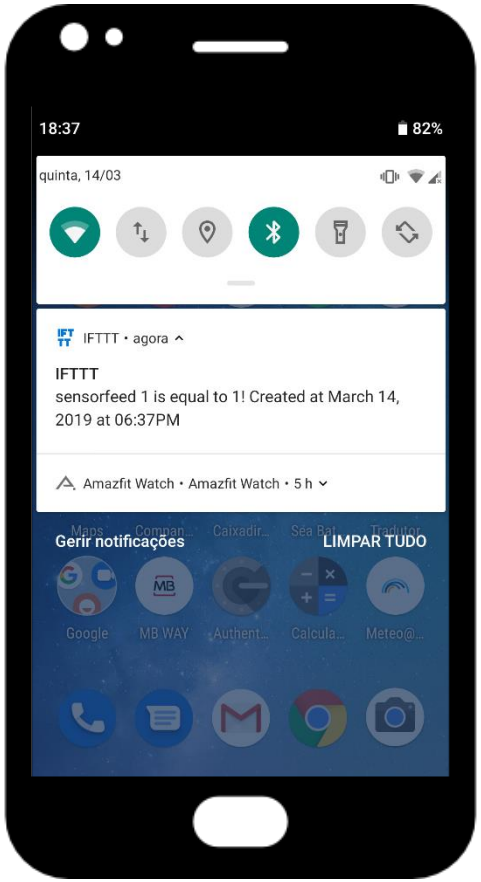
Go back to Adafruit IO and add a new data point with value 1 to the SensorFeed feed!



Adafruit IO + IFTTT

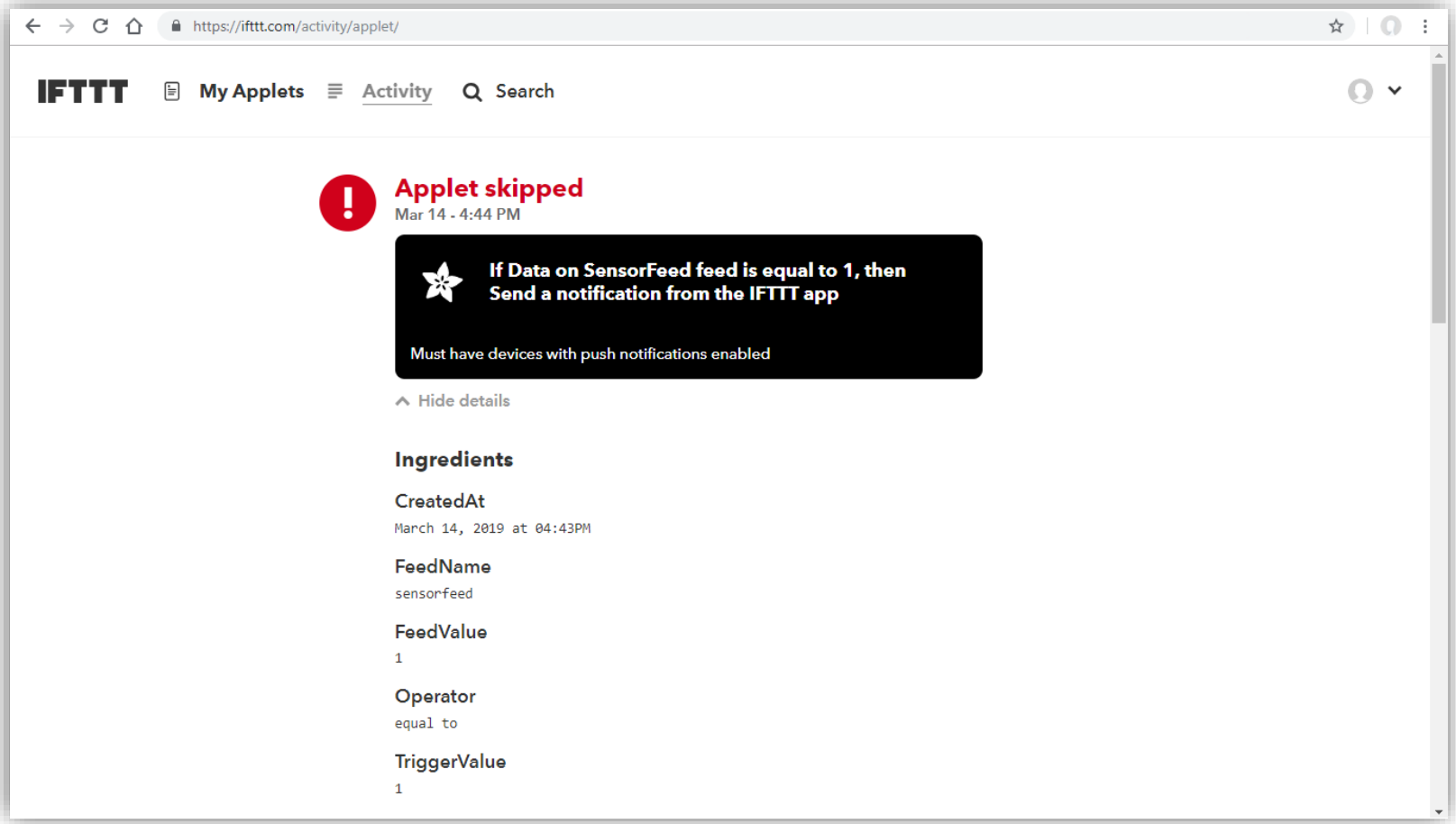
Notification received in the Smartphone!





Adafruit IO + IFTTT

Or not!!! You may need to enable push notifications in the Smartphone!



Adafruit IO + Java

Adafruit IO + Java

In this How To lets:

- Implement a simple Java Program to test interaction between Adafruit IO and Java
  - We will publish a value to the previously created feed named SensorFeed
- Eclipse Paho project provides an open-source client implementation of the MQTT protocol aimed at emerging applications for the Internet of Things
- References:
  - https://www.eclipse.org/paho
  - http://wiki.eclipse.org/Paho
  - http://www.eclipse.org/paho/clients/java

Adafruit IO + Java


```
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class MQTT_Test{

    private final static String ADAFRUIT_USERNAME = "YOUR_AIO_USERNAME";
    private final static String ADAFRUIT_AIO_KEY  = "YOUR_AIO_KEY";

    public static void main(String[] args) {

        String topic                  = ADAFRUIT_USERNAME + "/feeds/sensorfeed";
        String msg_content            = "Hello from java (not the island)!";
        int qos                       = 1; //QoS: 0 - at most once, 1 - at least once, 2 - exactly once
        String broker                 = "tcp://io.adafruit.com:1883"; //Adafruit IO broker
        String client_id              = "JavaSample";
        MemoryPersistence persistence = new MemoryPersistence();
        …
```

Adafruit IO + Java


```
try {
    MqttClient mqtt_client = new MqttClient(broker, client_id, persistence);

    MqttConnectOptions connOpts = new MqttConnectOptions();
    connOpts.setCleanSession(true);
    connOpts.setUserName(ADAFRUIT_USERNAME);
    connOpts.setPassword(ADAFRUIT_AIO_KEY.toCharArray());

    System.out.println("Connecting to broker: " + broker);
    mqtt_client.connect(connOpts);
    System.out.println("Connected. Publishing message: " + msg_content);

    MqttMessage message = new MqttMessage(msg_content.getBytes());
    message.setQos(qos);
    mqtt_client.publish(topic, message);
    System.out.println("Message published");
    mqtt_client.disconnect();
    System.out.println("Disconnected");
    System.exit(0);
}
```

Adafruit IO + Java


```
        ...
        catch(MqttException me) {
            System.out.println("reason: " + me.getReasonCode());
            System.out.println("msg: " + me.getMessage());
            System.out.println("loc: " + me.getLocalizedMessage());
            System.out.println("cause: " + me.getCause());
            System.out.println("excep: " + me);
            me.printStackTrace();
        }
    }

}
```

Adafruit IO + JavaScript

Adafruit IO + JavaScript

In this How To lets:

- Implement a simple web page to test interaction between Adafruit IO and JavaScript
  - We will publish a value to the previously created feed named SensorFeed
  - We will subscribe to the feed and update the page on a received message

- Eclipse Paho project also provides an open-source JS client implementing the MQTT protocol

- References:
  - https://www.eclipse.org/paho
  - http://wiki.eclipse.org/Paho
  - http://www.eclipse.org/paho/clients/js

- You may want to give a look at this too (using the REST API):
  - https://io.adafruit.com/api/docs/

Adafruit IO + JavaScript


```
<html>
  <head>
    <title>Adafruit IO + JS</title>
  </head>
  <body>
    <h1> Test it! </h1>
    <button onclick="publish()">Publish</button>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js"
            type="text/javascript"></script>
    <script>
      //create a client instance
      client = new Paho.MQTT.Client("io.adafruit.com", Number(443), "JS_Client");

      //set callback handlers
      client.onConnectionLost = onConnectionLost;
      client.onMessageArrived = onMessageArrived;
      //connect the client
      client.connect({onSuccess:onConnect, userName:"YOUR_AIO_USERNAME",
          password:"YOUR_AIO_KEY", useSSL:true, mqttVersion:4});
```

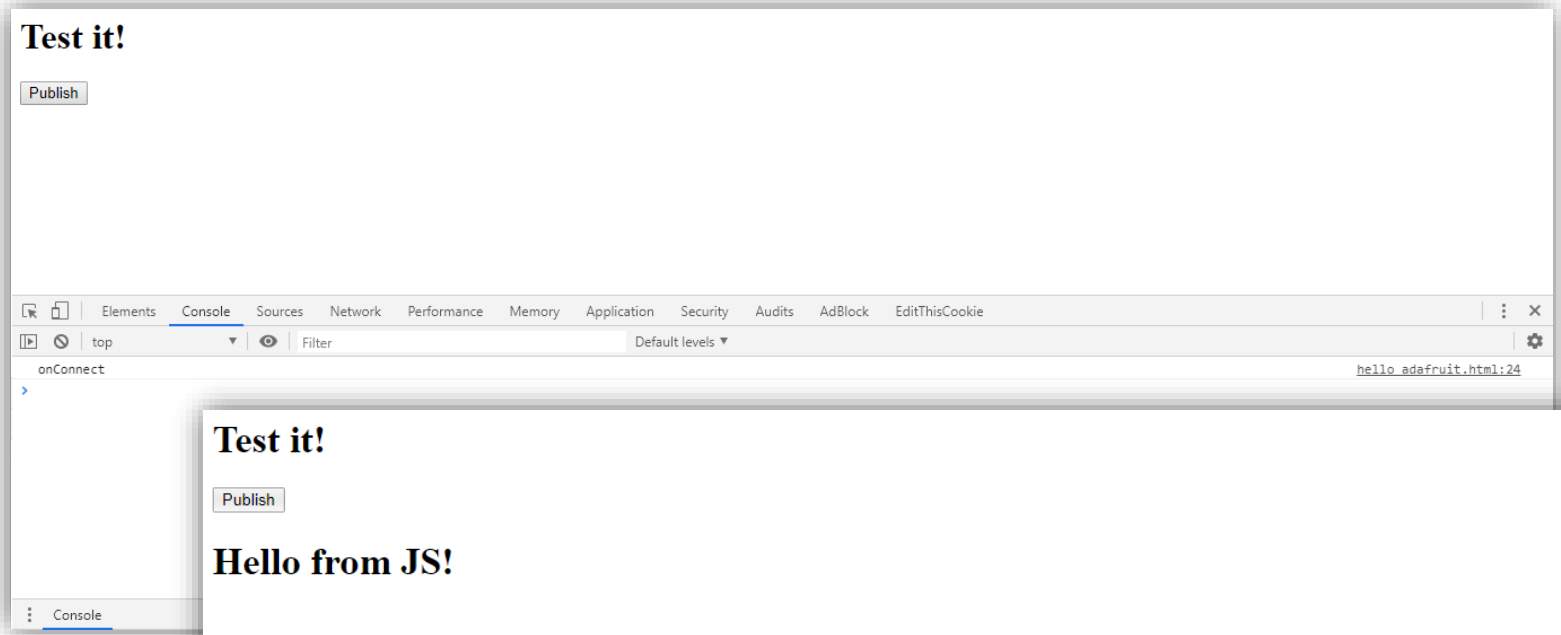
Adafruit IO + JavaScript

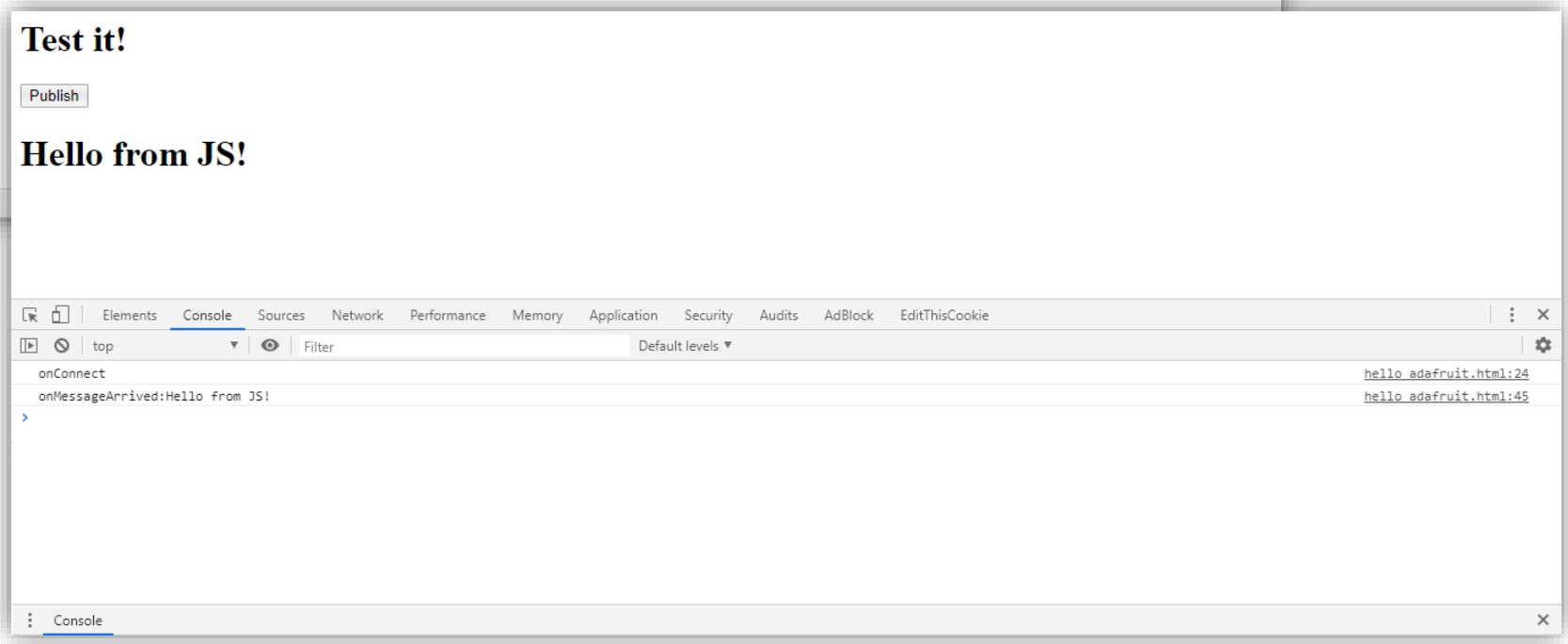

```
//called when the client connects
function onConnect() {
    console.log("onConnect");
    //subscribe
    client.subscribe("YOUR_AIO_USERNAME/feeds/sensorfeed");
}

function publish(){
    //send message
    message = new Paho.MQTT.Message("Hello from JS!");
    message.destinationName = "YOUR_AIO_USERNAME/feeds/sensorfeed";
    client.send(message);
}

//called when the client loses its connection
function onConnectionLost(responseObject) {
    if (responseObject.errorCode !== 0) {
        console.log("onConnectionLost:" + responseObject.errorMessage);
    }
}
```

Adafruit IO + JavaScript


```
        //called when a message arrives
        function onMessageArrived(message) {
            console.log("onMessageArrived:" + message.payloadString);
            var h1 = document.createElement("h1");
            h1.appendChild(document.createTextNode(message.payloadString));
            document.body.appendChild(h1);
        }
    </script>

  </body>
</html>
```

Adafruit IO + JavaScript





Adafruit IO + Arduino

Adafruit IO + Arduino

In this How To lets:

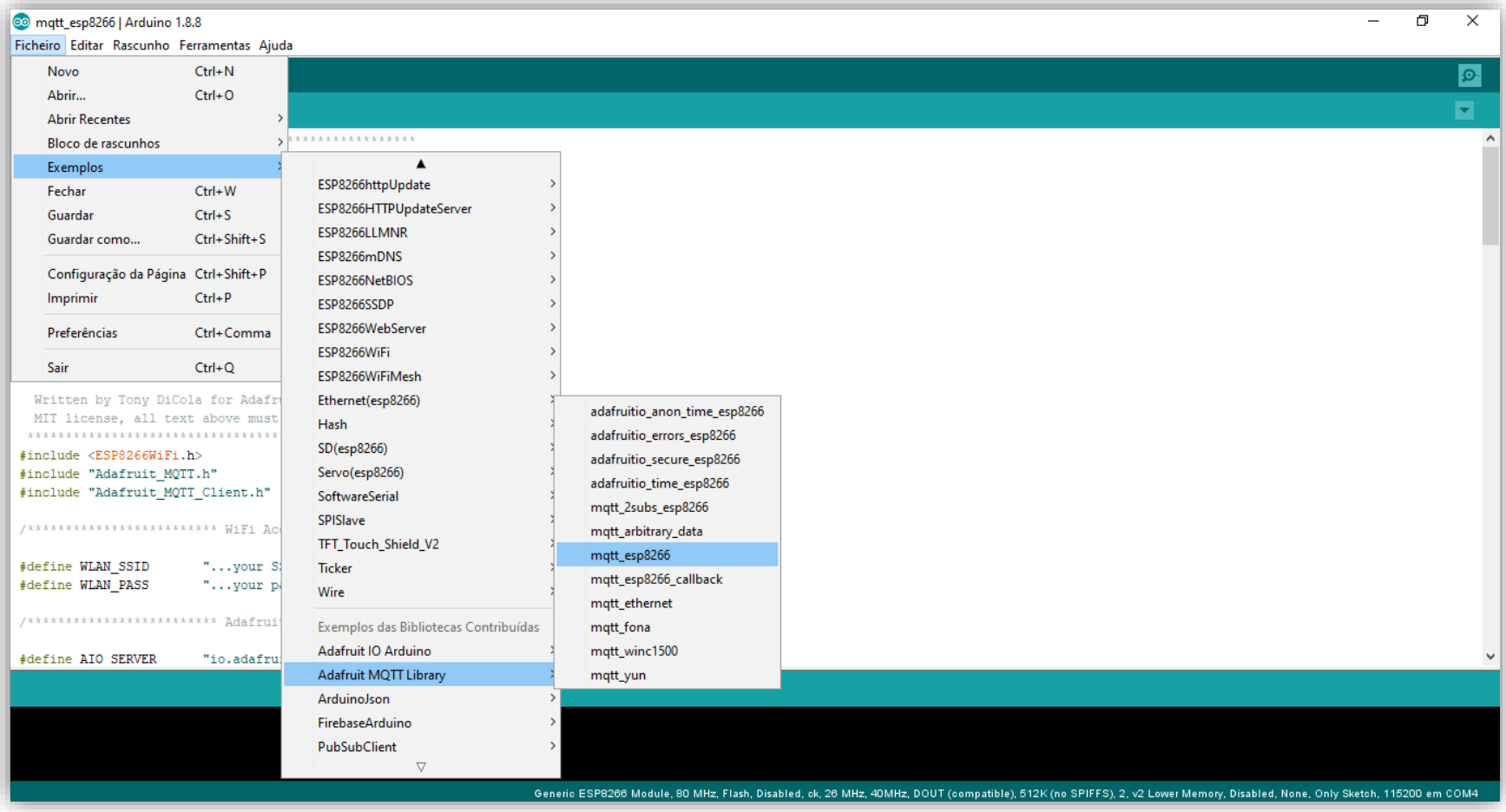
- Implement a simple sketch to test interaction between Adafruit IO and Arduino boards
  - We will publish a value to the previously created feed named SensorFeed
  - We will subscribe to the feed

- We will use the Adafruit MQTT library
  - We could use others such as the PubSubClient MQTT Library

- References:
  - https://learn.adafruit.com/adafruit-io/arduino
  - https://learn.adafruit.com/mqtt-adafruit-io-and-you/intro-to-adafruit-mqtt

Adafruit IO + Arduino

We will use mqtt_esp8266 example from the Adafruit MQTT Library to get started!



Adafruit IO + Arduino


```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************* WiFi Access Point *********************************/
#define WLAN_SSID       "SSID"
#define WLAN_PASS       "password"

/************************* Adafruit.io Setup *********************************/
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883            //use 8883 for SSL
#define AIO_USERNAME    "YOUR_AIO_USERNAME"
#define AIO_KEY         "YOUR_AIO_KEY"

/************ Global State (you don't need to change this!) ******************/
//create an ESP8266 WiFiClient class to connect to the MQTT server
WiFiClient client;
//or use WiFiFlientSecure for SSL -> WiFiClientSecure client

//setup the MQTT client class by passing in the WiFi client, MQTT server and login details
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
```

Adafruit IO + Arduino


```
/****************************** Feeds ***************************************/
Adafruit_MQTT_Publish sensorfeed_publish = Adafruit_MQTT_Publish(&mqtt,
                                                    AIO_USERNAME "/feeds/sensorfeed");
Adafruit_MQTT_Subscribe sensorfeed_subscribe = Adafruit_MQTT_Subscribe(&mqtt,
                                                    AIO_USERNAME "/feeds/sensorfeed");
/***************************** Sketch Code ************************************/
void MQTT_connect();

void setup() {
  Serial.begin(115200); //set the data rate in bits per second (baud) for serial data transmission
  Serial.println(F("*** Adafruit MQTT demo for SensorFeed ***")); //write to serial

  Serial.print("Connecting to "); Serial.println(WLAN_SSID);
  WiFi.begin(WLAN_SSID, WLAN_PASS); //connect to WiFi access point
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.println("WiFi connected"); Serial.print("IP address: "); Serial.println(WiFi.localIP());

  mqtt.subscribe(&sensorfeed_subscribe); //setup MQTT subscription for SensorFeed feed
}
```

Adafruit IO + Arduino


```
uint32_t x=0;

void loop() {

  MQTT_connect(); //ensure the connection to the MQTT server is alive. See MQTT_connect() definition

  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {  //wait for incoming subscrip packets subloop
    if (subscription == &sensorfeed_subscribe) {
      Serial.print(F("Got: ")); Serial.println((char *)sensorfeed_subscribe.lastread);
    }
  }

  Serial.print(F("\nSending sensor x val ")); Serial.print(x); Serial.println("...");
  if (! sensorfeed_publish.publish(x++)) {  //lets publish stuff to the SensorFeed
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }

}
```

Adafruit IO + Arduino


```
//function to connect and reconnect as necessary to the MQTT server
void MQTT_connect() {
  if (mqtt.connected()) {  //return if already connected
    return;
  }

  Serial.println("Connecting to MQTT... ");

  int8_t ret;
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { //connect will return 0 for connected
    Serial.println(mqtt.connectErrorString(ret)); Serial.println("Retrying MQTT connection in 5 seconds");
    mqtt.disconnect();
    delay(5000);  //wait 5 seconds to retry
    retries--;
    if (retries == 0) {
      while (1);  //basically die and wait for the Watchdog Timer to reset us
    }
  }
  Serial.println("MQTT Connected!!!");
}
```

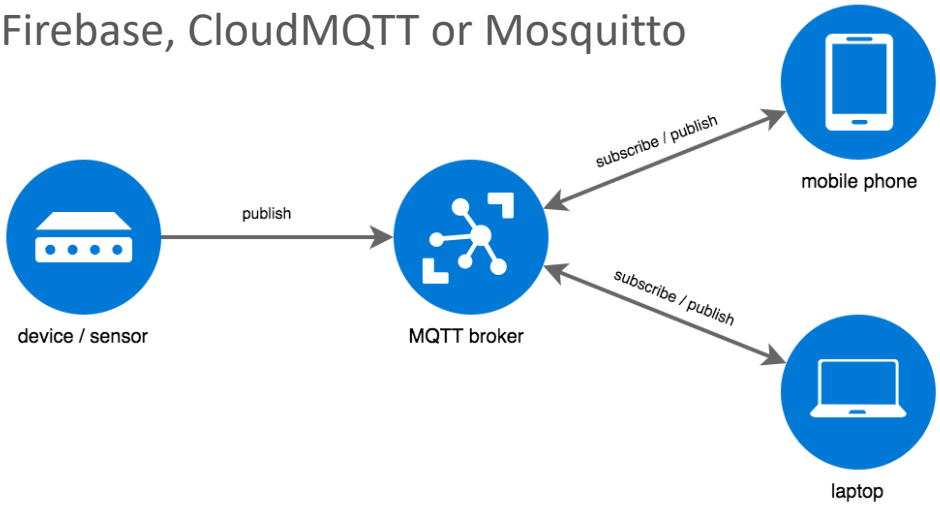
Final Notes

Final Notes

- Using the MQTT protocol we have established a publish/subscribe service using an extremely simple and lightweight messaging protocol

- MQTT libraries can be found for many programming languages

- We need a MQTT broker/server to store our messages
  - We used Adafruit IO as a MQTT broker
  - There are alternatives such as Firebase, CloudMQTT or Mosquitto



Final Notes

Now, you can:

- Integrate your sensors and publish their values in an Adafruit IO feed, for example
- Create an IFTTT applet to react to different contexts based on your sensor values and send notifications/emails/sms/… You choose!!
  - Send a notification when the temperature is uncomfortable
  - Send an email when movement is detected
  - Send an email and notification when a flame is detected

Final Notes

We could also have used Firebase databases instead of Adafruit IO:

- FirebaseArduino library helps you with that
- It uses Firebase API
  - a full abstraction of Firebase's REST API exposed through C++ calls
- References
  - https://firebase-arduino.readthedocs.io/en/latest/
  - https://github.com/FirebaseExtended/firebase-arduino/blob/master/examples/FirebaseDemo_ESP8266/FirebaseDemo_ESP8266.ino
  - https://github.com/FirebaseExtended/firebase-arduino

Hands On