



University of Minho
School of Engineering



Ambient Intelligence

Autonomous Systems

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

Bruno Fernandes, Cesar Analide, Fábio Silva

18/03/2019

Contents

2

Concepts

How To

Hands On

- Concepts
 - Adafruit IO
 - Firebase
 - IFTTT
 - MQTT Protocol
- How To
 - Adafruit IO + IFTTT
 - Adafruit IO + Java
 - Adafruit IO + JavaScript
 - Adafruit IO + Arduino
- Hands On

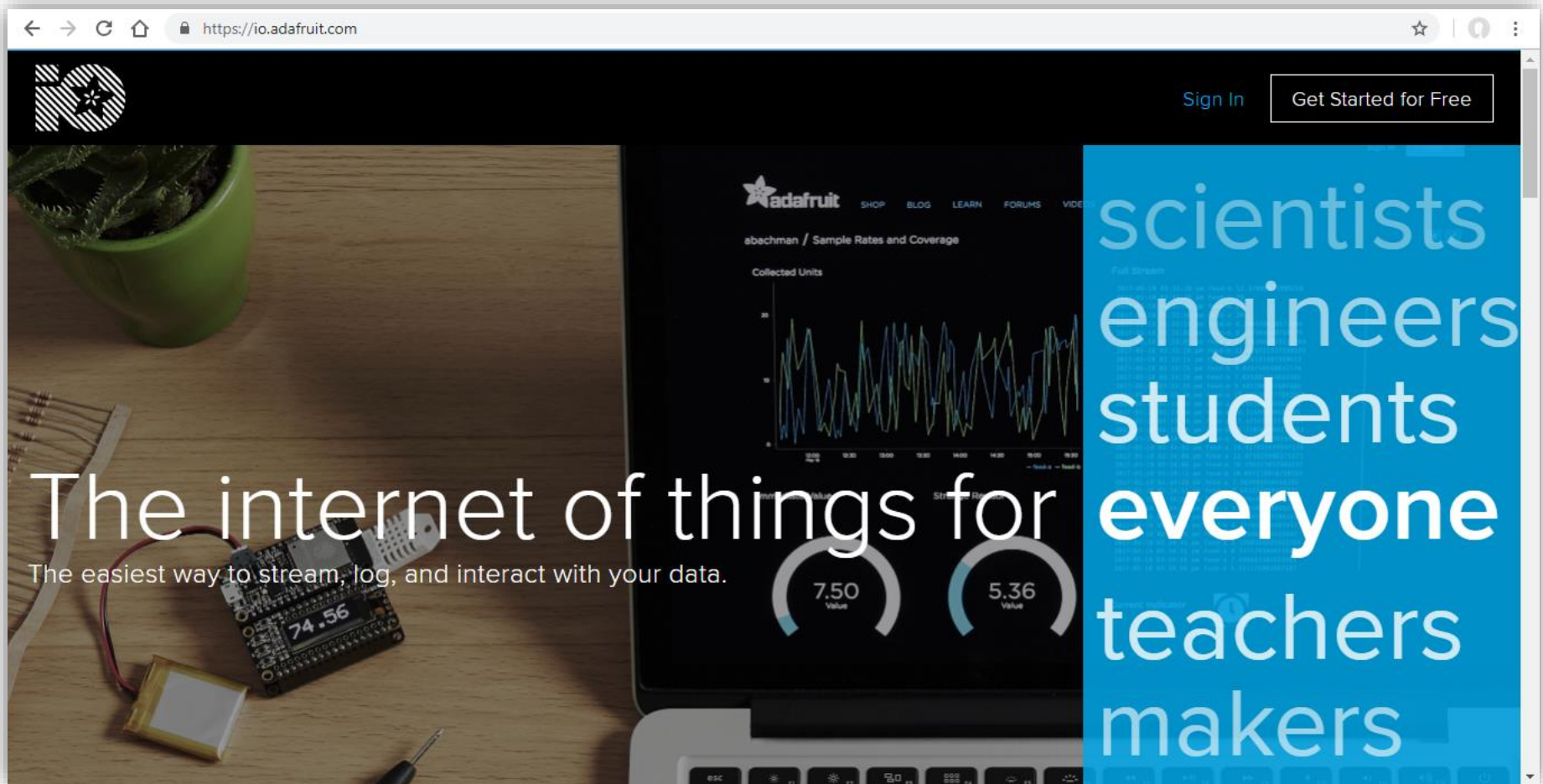
Adafruit IO

3

CONCEPTS

How To

Hands On



The screenshot shows the Adafruit IO web interface. The browser address bar displays <https://io.adafruit.com>. The page features the Adafruit logo, navigation links (SHOP, BLOG, LEARN, FORUMS, VIDEOS), and buttons for "Sign In" and "Get Started for Free". The main content area shows a dashboard for a user named "abechman" with the title "Sample Rates and Coverage". It includes a line graph titled "Collected Units" and two circular gauges displaying "74.56" and "5.36". A blue sidebar on the right lists user roles: "scientists", "engineers", "students", "everyone", "teachers", and "makers". The background of the website features a photograph of an Adafruit IO board connected to a small green potted plant and a battery.

The internet of things for

The easiest way to stream, log, and interact with your data.

scientists
engineers
students
everyone
teachers
makers

Adafruit IO

4

CONCEPTS

How To

Hands On

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

5

CONCEPTS

How To

Hands On

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

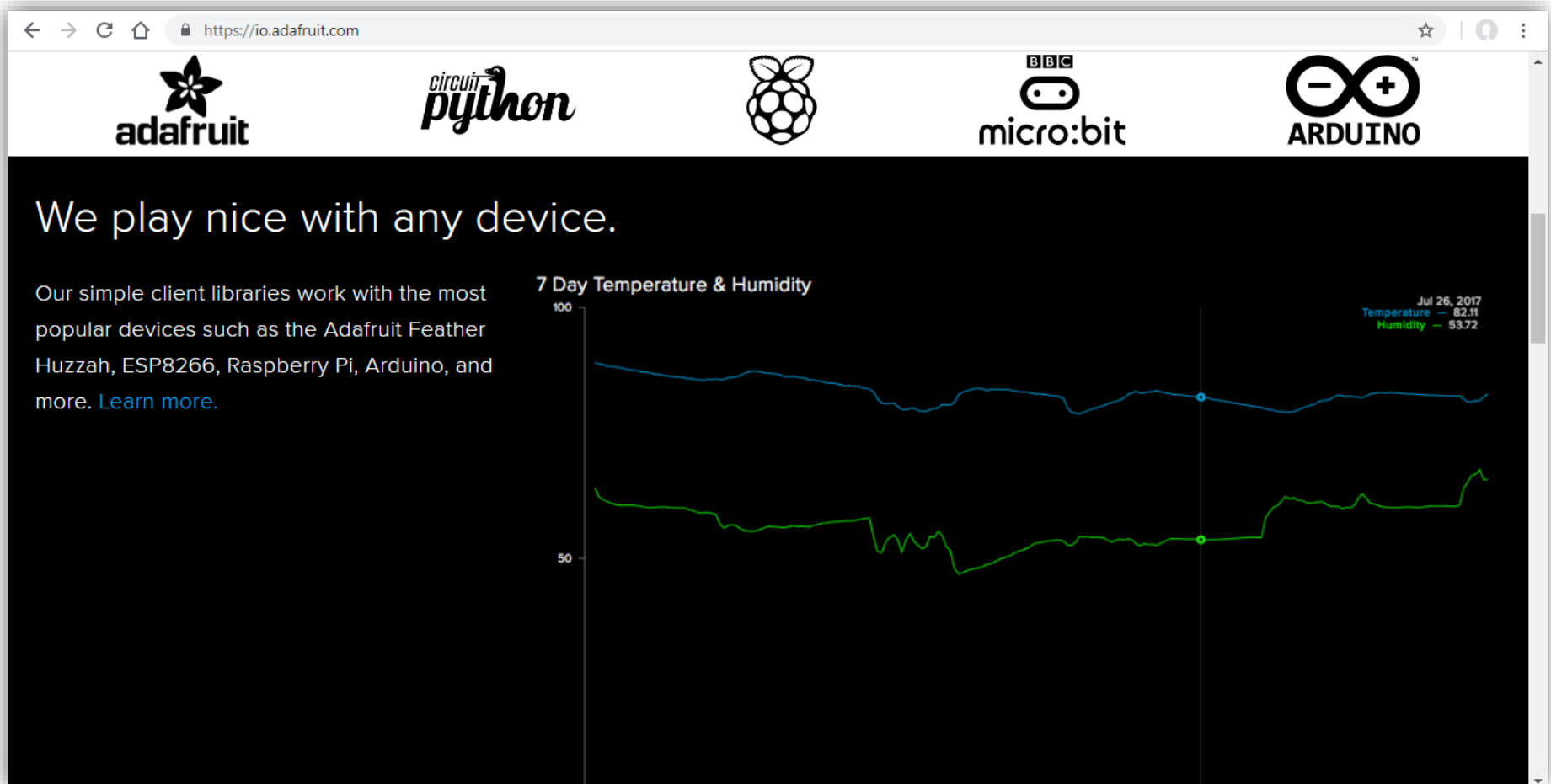
Adafruit IO

6

CONCEPTS

How To

Hands On



Adafruit IO

7

CONCEPTS

How To

Hands On

The screenshot displays the Adafruit IO web interface in a browser window. The address bar shows the URL `https://io.adafruit.com/.../feeds/sensorfeed`. The page title is `/Feeds/SensorFeed`. On the left, a sidebar contains navigation links: Home, Feeds, Dashboards, Triggers, Services, View AIO Key, API Docs, FAQ, IO Plus, Learn, News, Support, Terms of Service, Get Help, and Send Feedback. At the bottom left, a 'Free Usage' section shows: Feeds: 1 of 10, Dashboards: 0 of 5, Rate: 30 / minute, Current Usage: 0 / min, and Storage: 30 days. The main content area features a graph with a y-axis from -1.0 to 1.0 and an x-axis from Mar 14 to Mar 15. A white 'Add Data' modal is centered over the graph, containing a 'VALUE' input field with a cursor, and 'Cancel' and 'Create' buttons. On the right, a sidebar lists settings for the feed: 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.), and Webhooks (Webhooks let you connect). At the bottom of the main area, there are buttons for '+ Add Data', 'Download All Data', and 'Filter'.

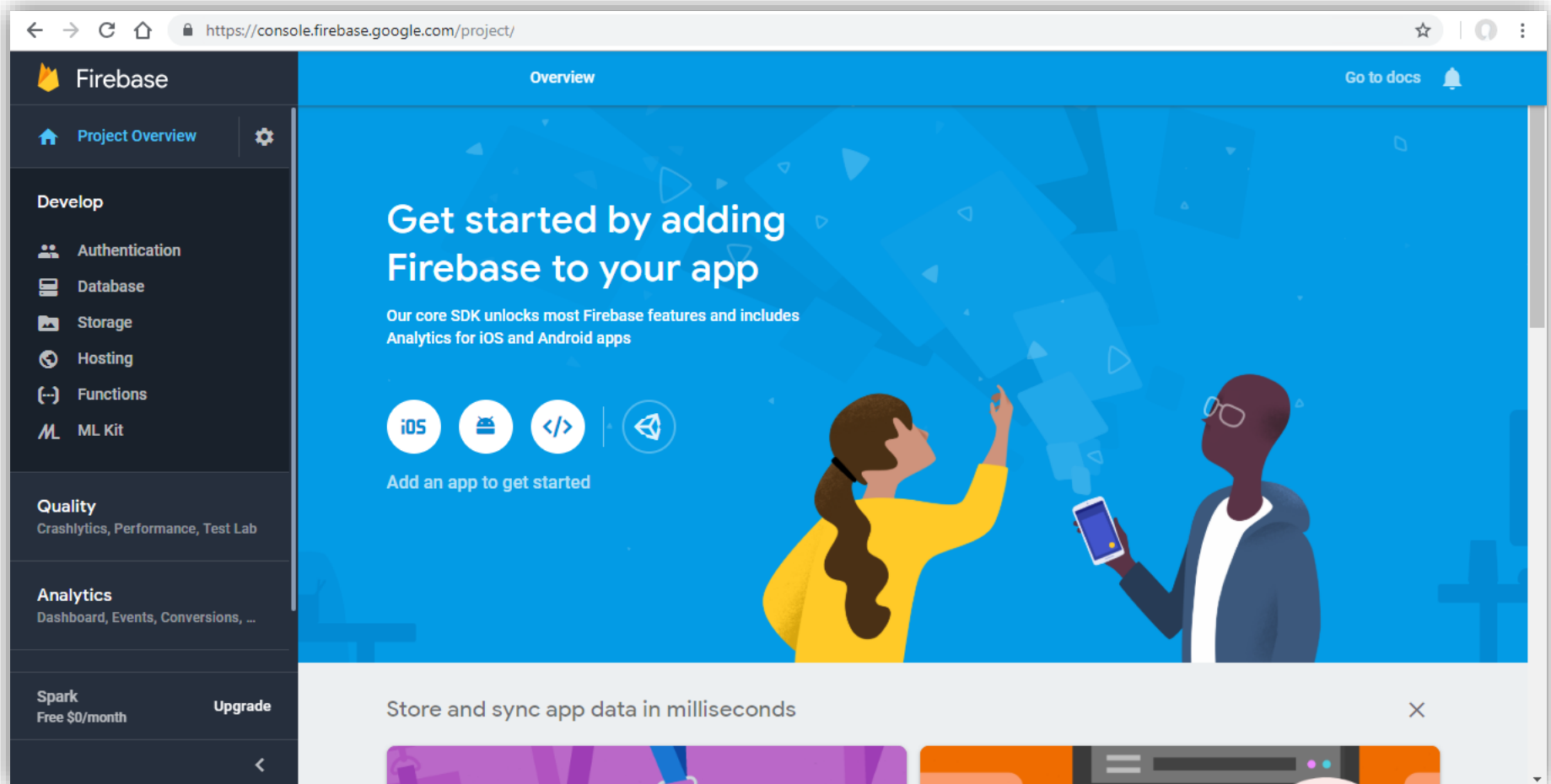
Firebase

8

CONCEPTS

How To

Hands On



Firebase

9

CONCEPTS

How To

Hands On

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

10

CONCEPTS

How To

Hands On

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

Firestore

11

CONCEPTS

How To

Hands On

The screenshot displays the Firebase console interface. On the left is a dark sidebar with navigation links: Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab), Analytics (Dashboard, Events, Conversions, ...), and Spark (Free \$0/month, Upgrade). The main content area has a blue header with 'Database' and a 'Realtime Database' dropdown. Below the header are tabs for Data, Rules, Backups, and Usage. The 'Data' tab is active, showing a tree view of the database structure. The tree shows a root node 'ProbeData1' with a child 'ProbeData10', which has a child 'probes'. The 'probes' node contains three indexed items (0, 1, 2). Item 0 has fields: mac: 'da:a1:19:67:08:61', previousMillisDetected: 455876, and rssi: '-96'. Item 1 has fields: mac: 'e8:93:09:03:0a:3', previousMillisDetected: 456239, and rssi: '-83'. Item 2 has field: mac: '7c:2e:dd:c1:c1:4'.

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

Go to docs

Database

Realtime Database

Data Rules Backups Usage

ProbeData1

ProbeData10

probes

0

mac: "da:a1:19:67:08:61"

previousMillisDetected: 455876

rssi: "-96"

1

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

previousMillisDetected: 456239

rssi: "-83"

2

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

IFTTT - If This Then That

12

CONCEPTS

How To

Hands On

The screenshot shows the IFTTT website interface. At the top, there's a navigation bar with the IFTTT logo, a link to 'Build new service', and buttons for 'Sign in' and 'Sign up'. The main heading reads 'A world that works for you'. Below this, a paragraph states: 'IFTTT is the free way to get all your apps and devices talking to each other. Not everything on the internet plays nice, so we're on a mission to build a more connected world.' A sign-up section includes an email input field, a 'Get started' button, and options to 'Continue with Google' or 'Continue with Facebook'. On the right side, a diagram illustrates various smart devices and services connected by arrows, representing the IFTTT ecosystem. The devices include a lightbulb, a smartphone, a smart speaker, a smartwatch, a calendar, a calendar event, a music player, a cloud storage icon, and a smart home hub. Arrows indicate the flow of data and actions between these devices, such as a calendar event triggering a smart speaker to play music or a smartwatch sending data to a cloud storage icon.

IFTTT Build new service Sign in Sign up

A world that works for you

IFTTT is the free way to get all your apps and devices talking to each other. Not everything on the internet plays nice, so we're on a mission to build a more connected world.

Enter your email Get started

or

Continue with Google Continue with Facebook

Diagram illustrating various smart devices and services connected by arrows, representing the IFTTT ecosystem. The devices include a lightbulb, a smartphone, a smart speaker, a smartwatch, a calendar, a calendar event, a music player, a cloud storage icon, and a smart home hub. Arrows indicate the flow of data and actions between these devices, such as a calendar event triggering a smart speaker to play music or a smartwatch sending data to a cloud storage icon.

IFTTT

13

CONCEPTS

How To

Hands On

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**.

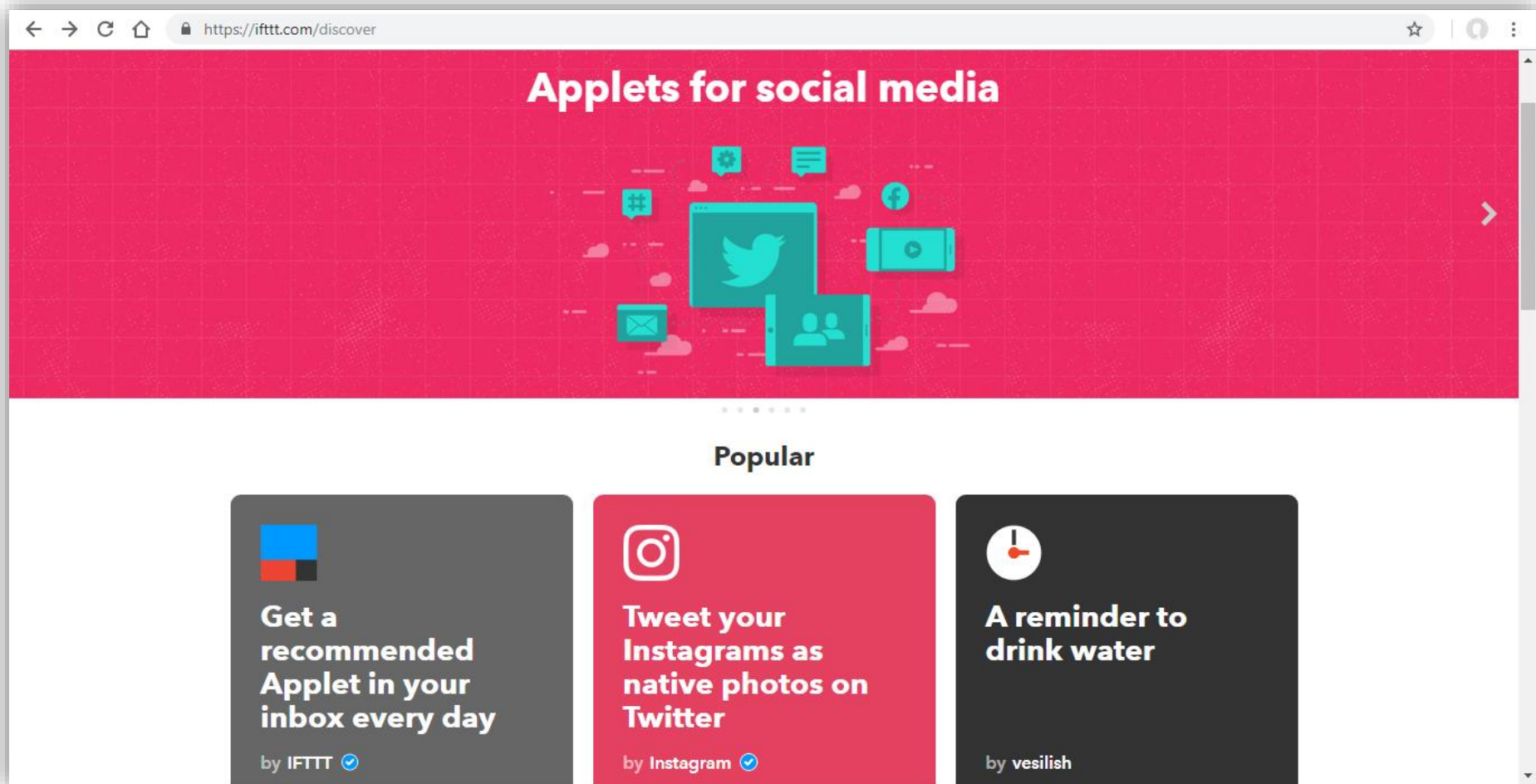
IFTTT

14

CONCEPTS

How To

Hands On



IFTTT

15

CONCEPTS

How To



Hands On

The screenshot shows a web browser window with the IFTTT website. The address bar displays the URL <https://ifttt.com/applets/yZHQBC5f-a-reminder-to-drink-water>. The IFTTT logo is in the top left, followed by a search bar. In the top right, there are links for 'Sign in' and a 'Sign up' button. Below the navigation bar, the breadcrumb 'My Applets > Date & Time' is visible. The main content area features a dark-themed card for an applet. At the top of the card is a clock icon with a red hand and a settings gear icon. The title 'A reminder to drink water' is prominently displayed. Below the title, a description states: 'This applet sends you a notification "Drink a glass of water!" every 2 hours between 8 am and 8 pm :)'. The creator is listed as 'by vesilish'. A large 'Get Started' button is centered at the bottom of the card. At the very bottom of the card, it shows '46k' users and 'works with' a notification bell icon.

← → ↻ 🏠 <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

16

CONCEPTS

How To

Hands On

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

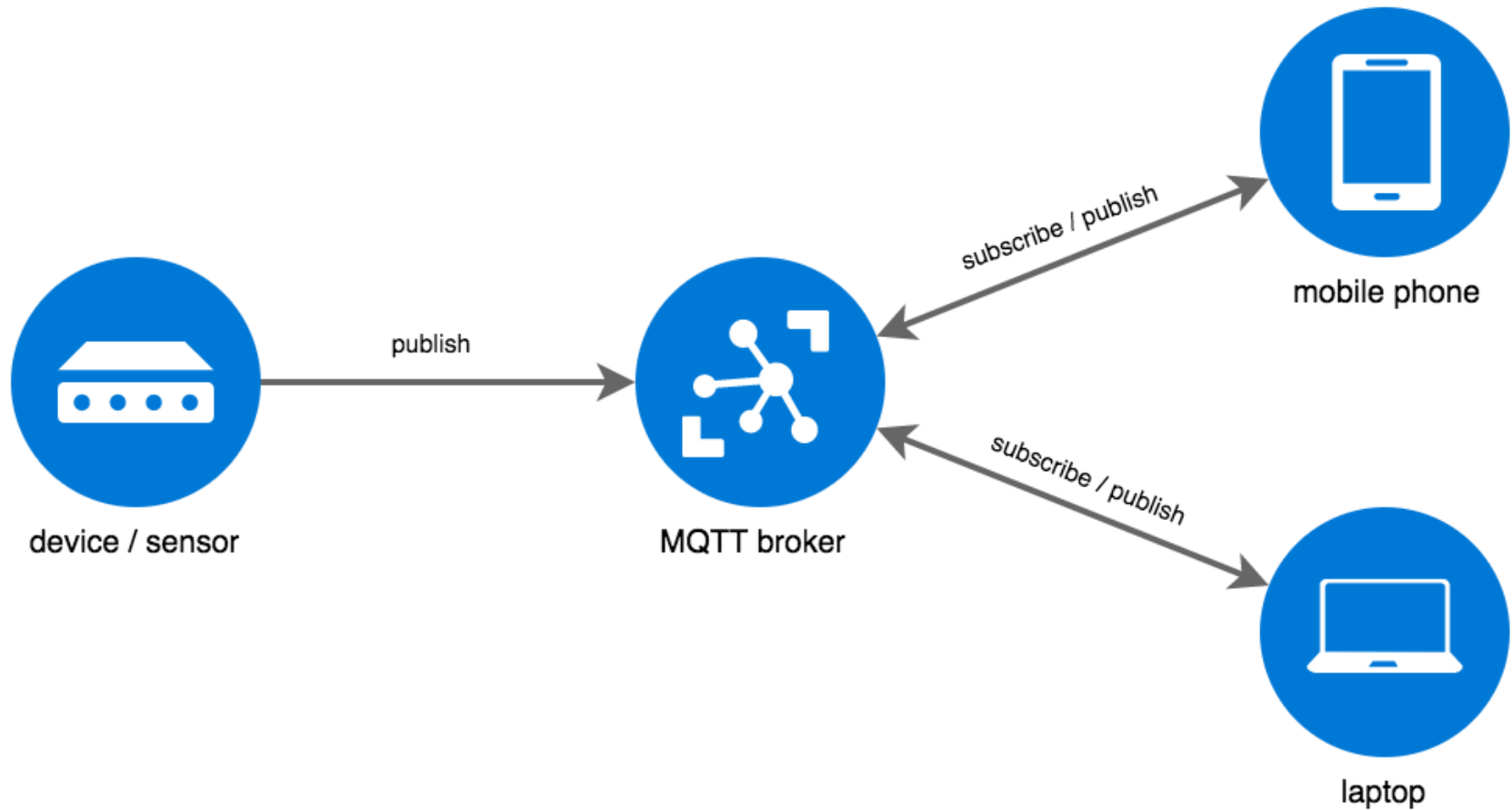
MQTT

17

CONCEPTS

How To

Hands On



Adafruit IO + IFTTT

18

Concepts

HOW TO

Hands On



Adafruit IO + IFTTT



19

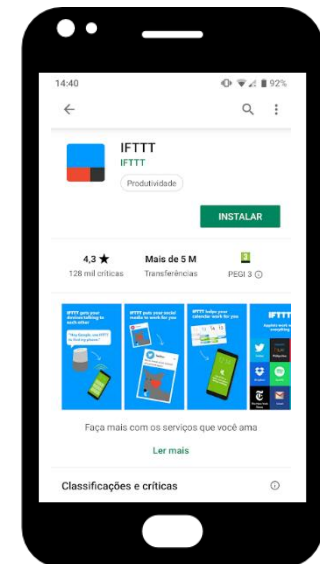
Concepts

HOW TO

Hands On

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



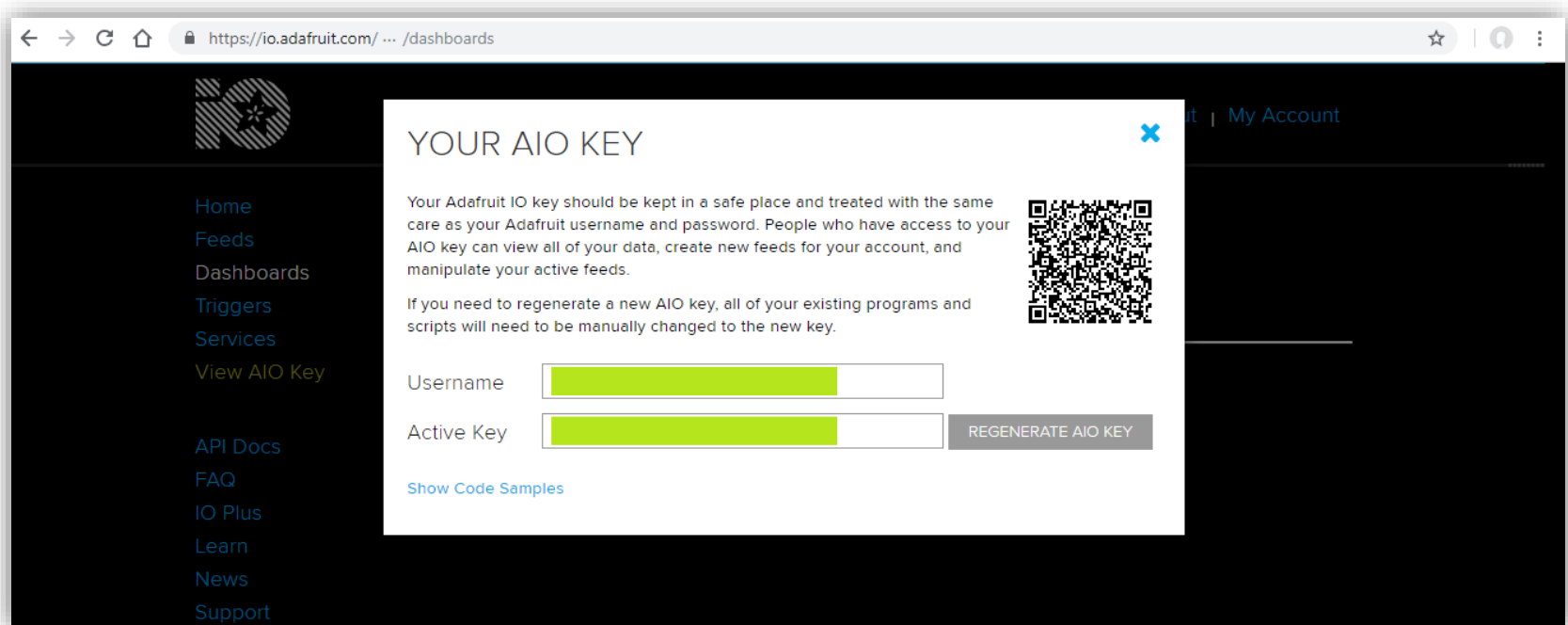
20

Concepts

HOW TO

Hands On

Take note of your **username** and **AIO Key**



Adafruit IO + IFTTT



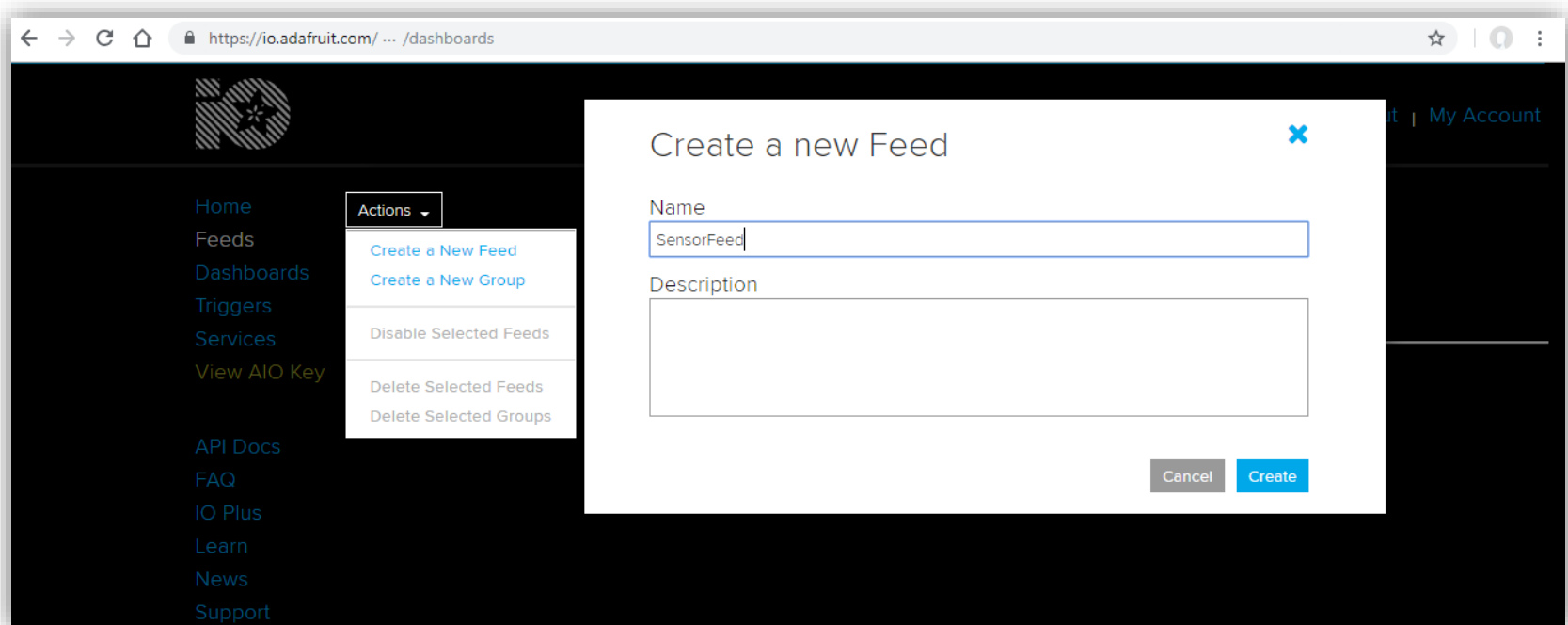
21

Concepts

HOW TO

Hands On

Create a **new feed** named **SensorFeed** in Adafruit IO



Adafruit IO + IFTTT



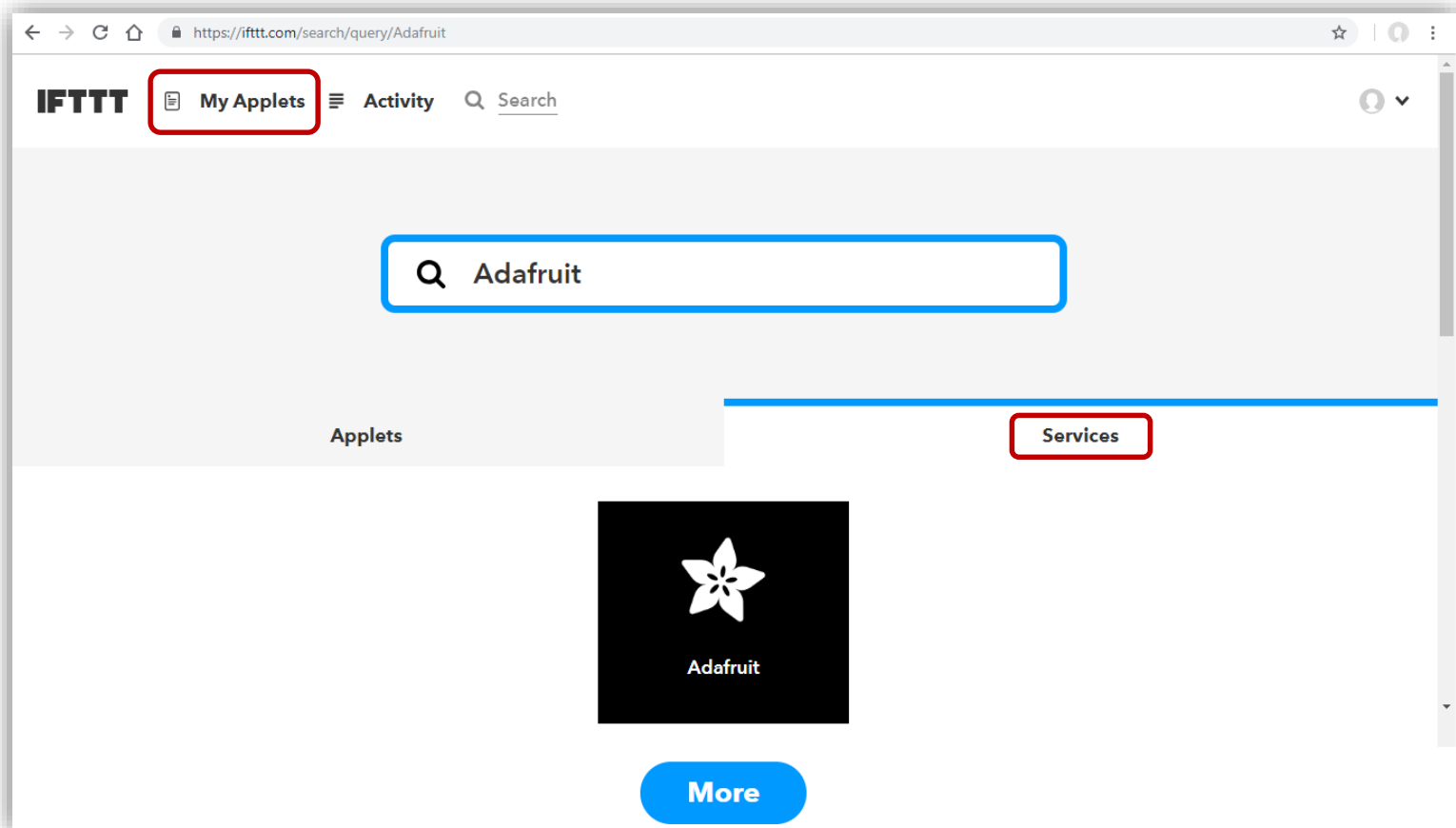
22

Concepts

HOW TO

Hands On

Login into IFTTT and **search for the Adafruit** service



Adafruit IO + IFTTT



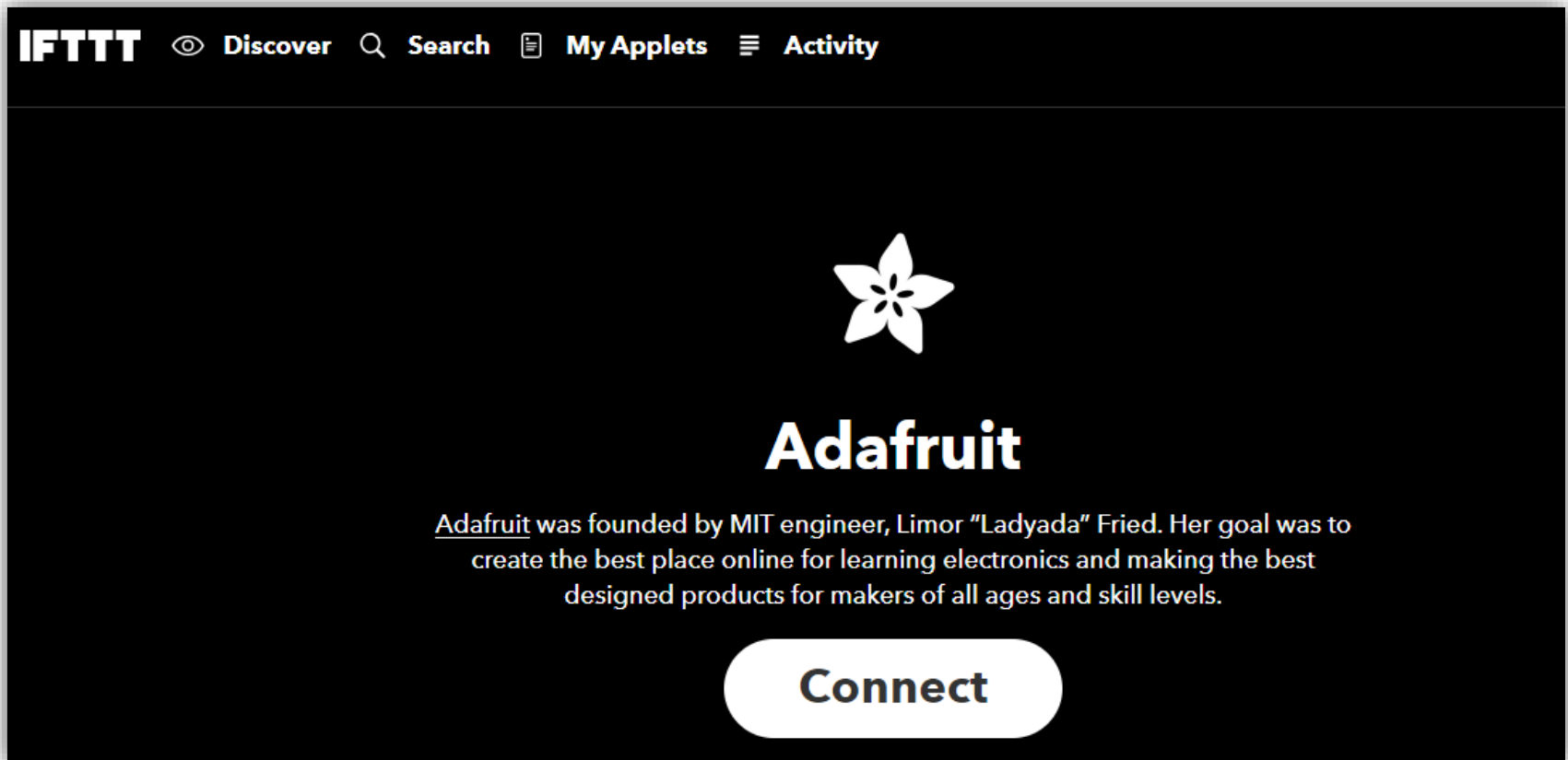
23

Concepts

HOW TO

Hands On

Connect IFTTT to the Adafruit platform



Adafruit IO + IFTTT



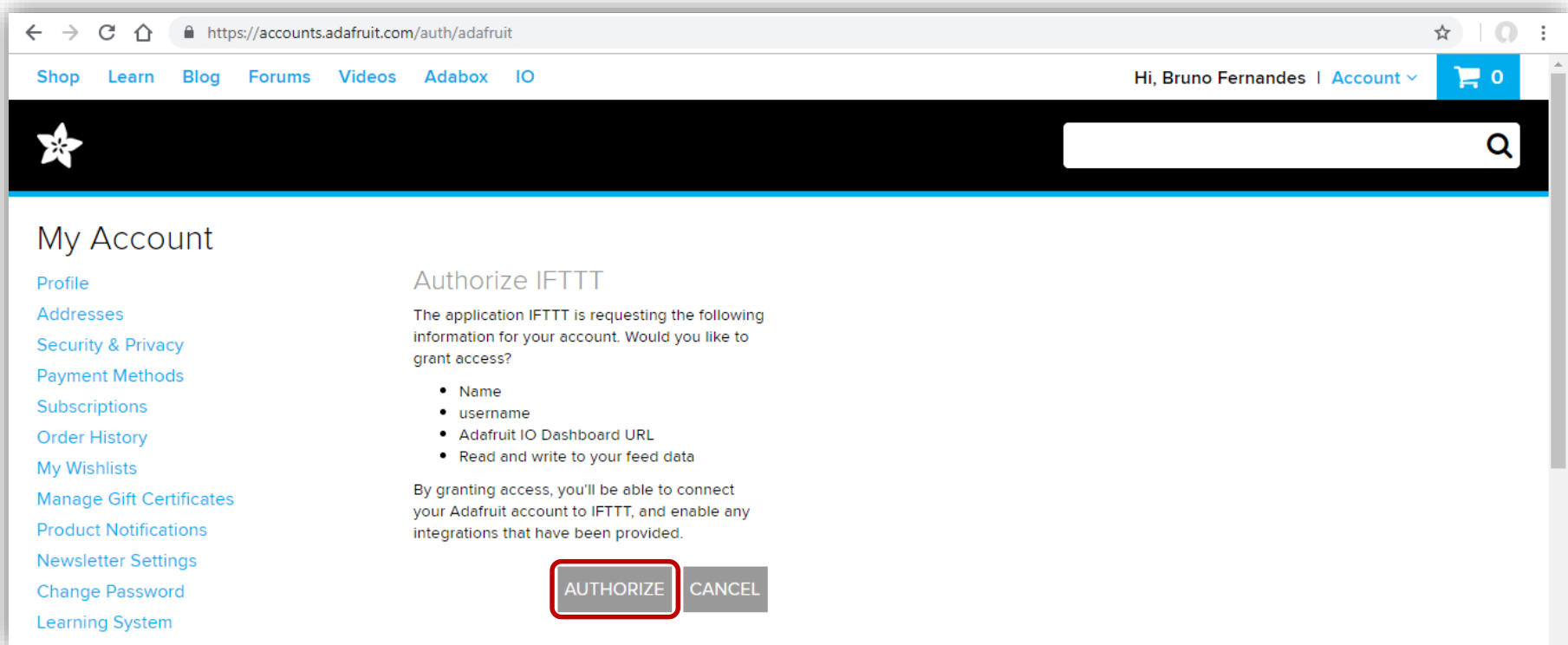
24

Concepts

HOW TO

Hands On

Authorize IFTTT access



Adafruit IO + IFTTT



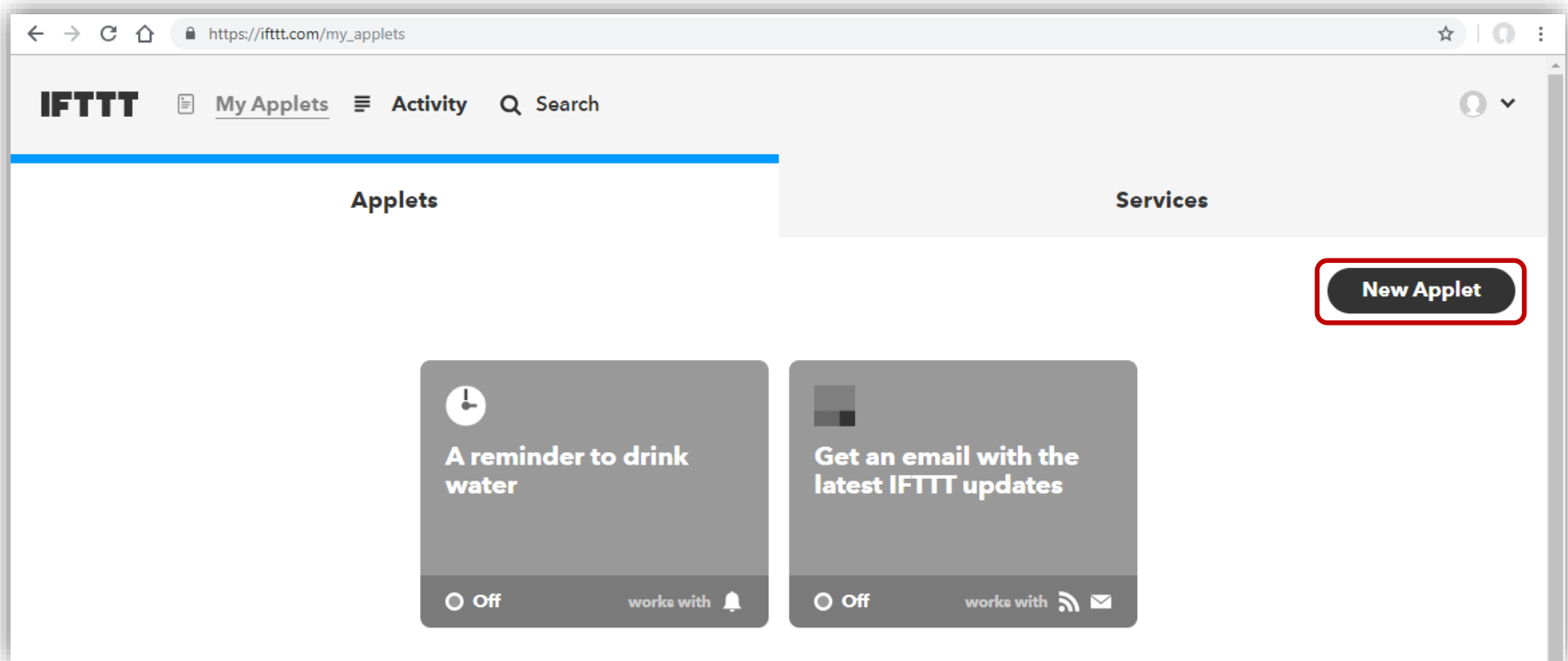
25

Concepts

HOW TO

Hands On

On the IFTTT platform lets **create a new applet**



Adafruit IO + IFTTT



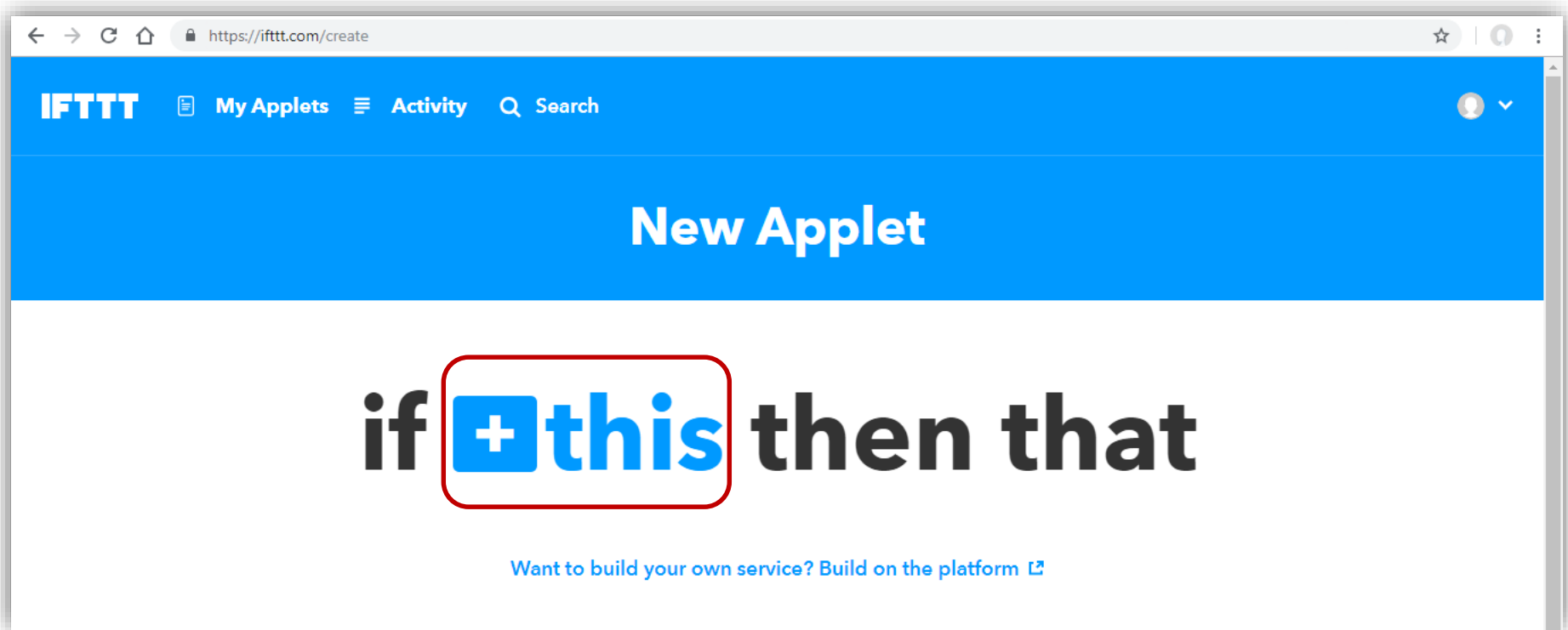
26

Concepts

HOW TO

Hands On

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



Adafruit IO + IFTTT



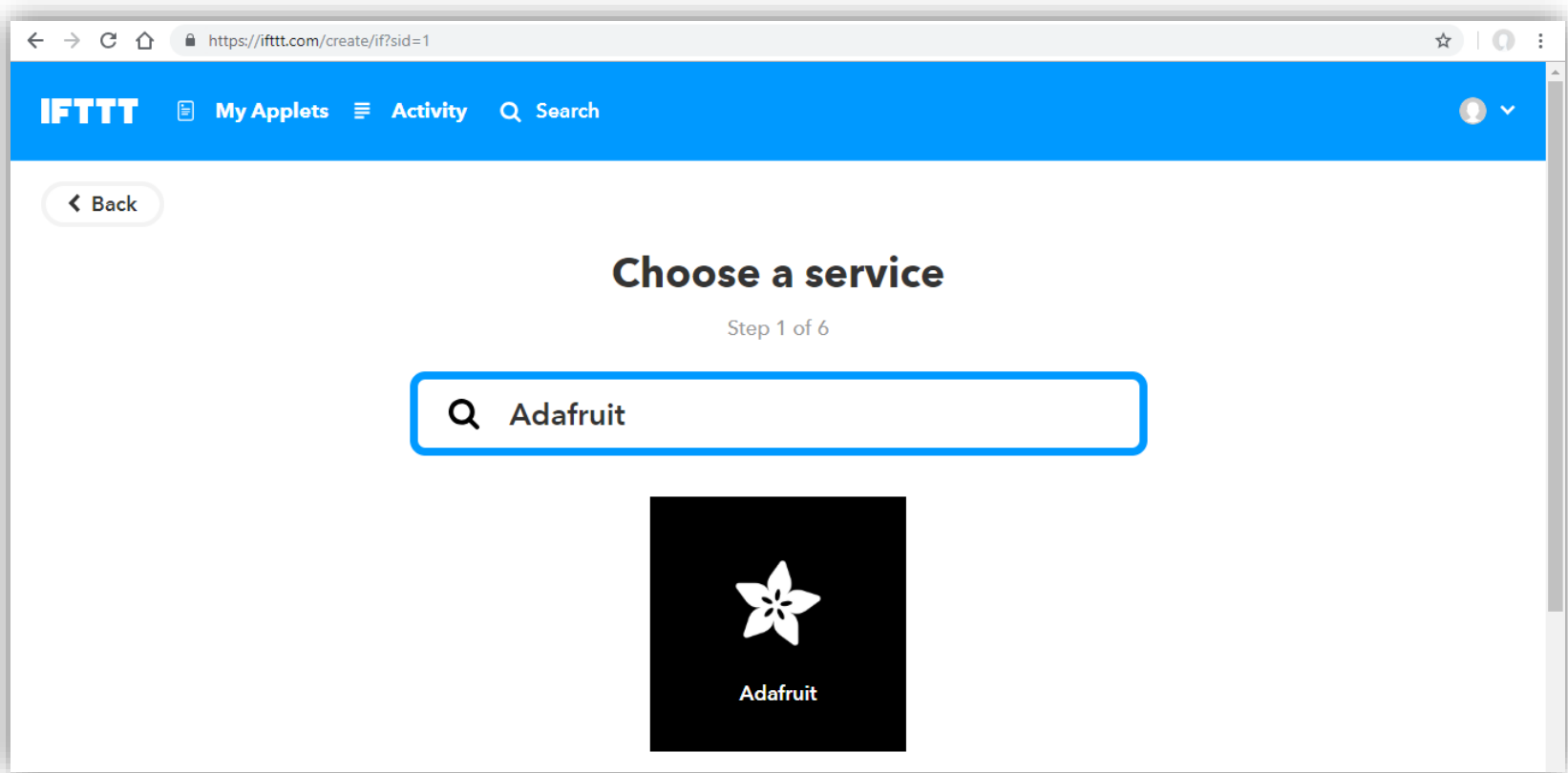
27

Concepts

HOW TO

Hands On

Search for the **Adafruit** service



Adafruit IO + IFTTT



28

Concepts

HOW TO


Hands On

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

A screenshot of a web browser showing the IFTTT 'Choose trigger' step. The browser address bar shows 'https://ifttt.com/create/if?sid=1'. The IFTTT logo and navigation links are at the top. The main heading is 'Choose trigger' with a star icon and 'Step 2 of 6'. Two trigger options are shown: 'Monitor a feed on Adafruit IO' (highlighted with a red border) and 'Any new data'.

IFTTT My Applets Activity Search

< Back

 **Choose trigger**
Step 2 of 6

Monitor a feed on Adafruit IO
This Trigger fires anytime it validates the data that you send to your feed. Example: If Feed Temperature > 80, fire Trigger.

Any new data
This Trigger fires any time there is new data in your feed.

Adafruit IO + IFTTT



29

Concepts

HOW TO

Hands On

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

A screenshot of the IFTTT web interface showing the 'Complete trigger fields' step (Step 2 of 6) for creating a trigger. The interface is in a browser window with the URL 'https://ifttt.com/create/iftt?sid=1'. The top navigation bar is blue with the IFTTT logo and links for 'My Applets', 'Activity', and 'Search'. A 'Back' button is visible. The main content area has a black star icon and the title 'Complete trigger fields'. Below this, a black box contains the trigger configuration: 'Monitor a feed on Adafruit IO'. A description states: 'This Trigger fires anytime it validates the data that you send to your feed. Example: If Feed Temperature > 80, fire Trigger.' The configuration fields are: 'Feed' (dropdown menu with 'SensorFeed' selected), 'Relationship' (dropdown menu with 'equal to' selected), and 'Value' (text input with '1'). Each field has a small explanatory text below it. A 'Create trigger' button is at the bottom of the black box.

Adafruit IO + IFTTT



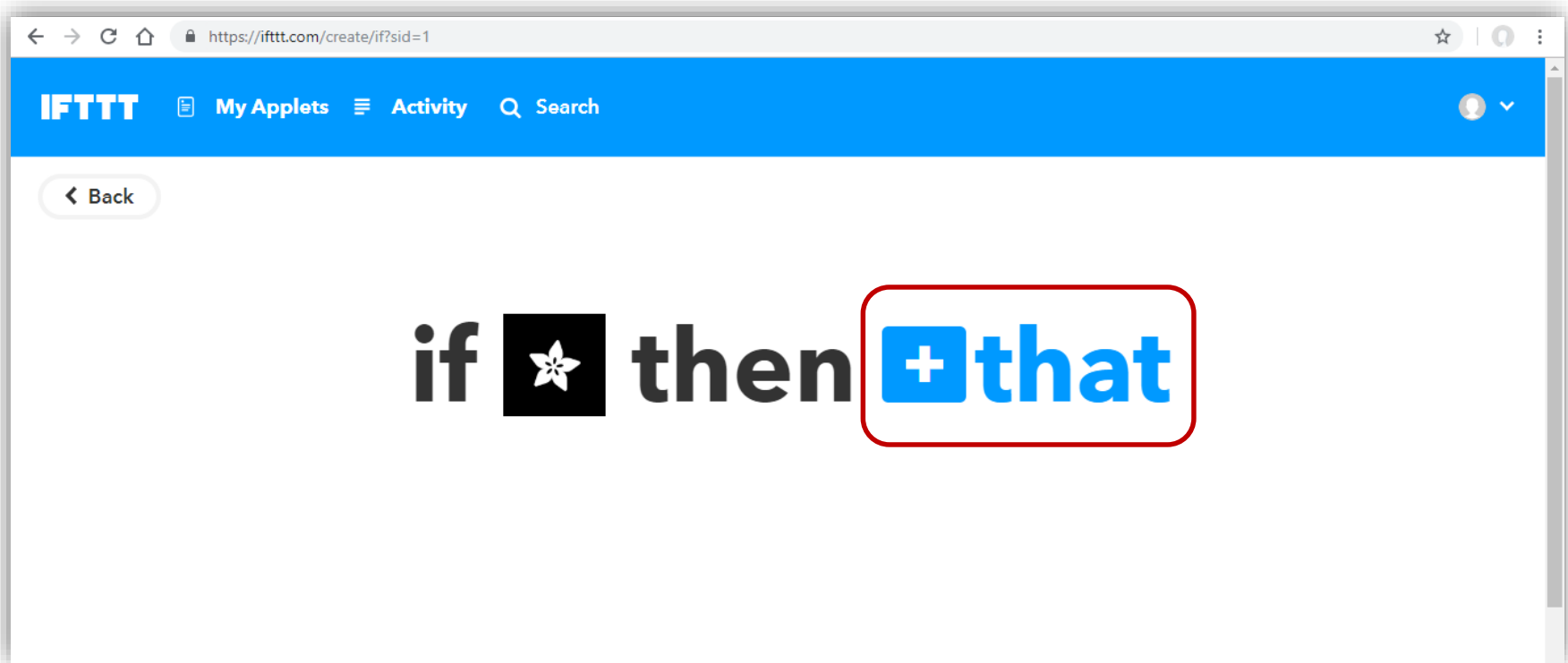
30

Concepts

HOW TO

Hands On

If ... Then **That**



Adafruit IO + IFTTT



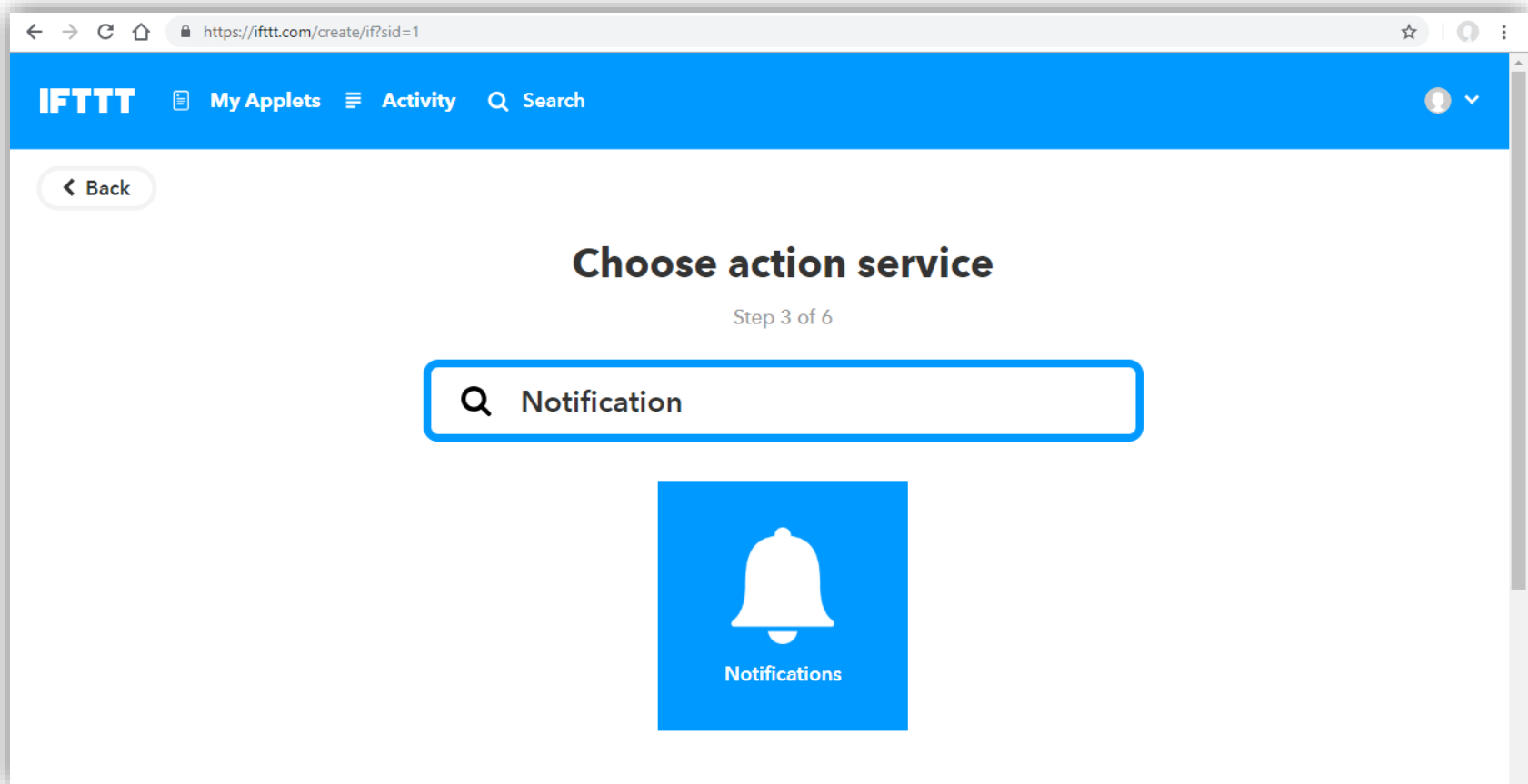
31

Concepts

HOW TO

Hands On

Search for the **Notifications** service



Adafruit IO + IFTTT



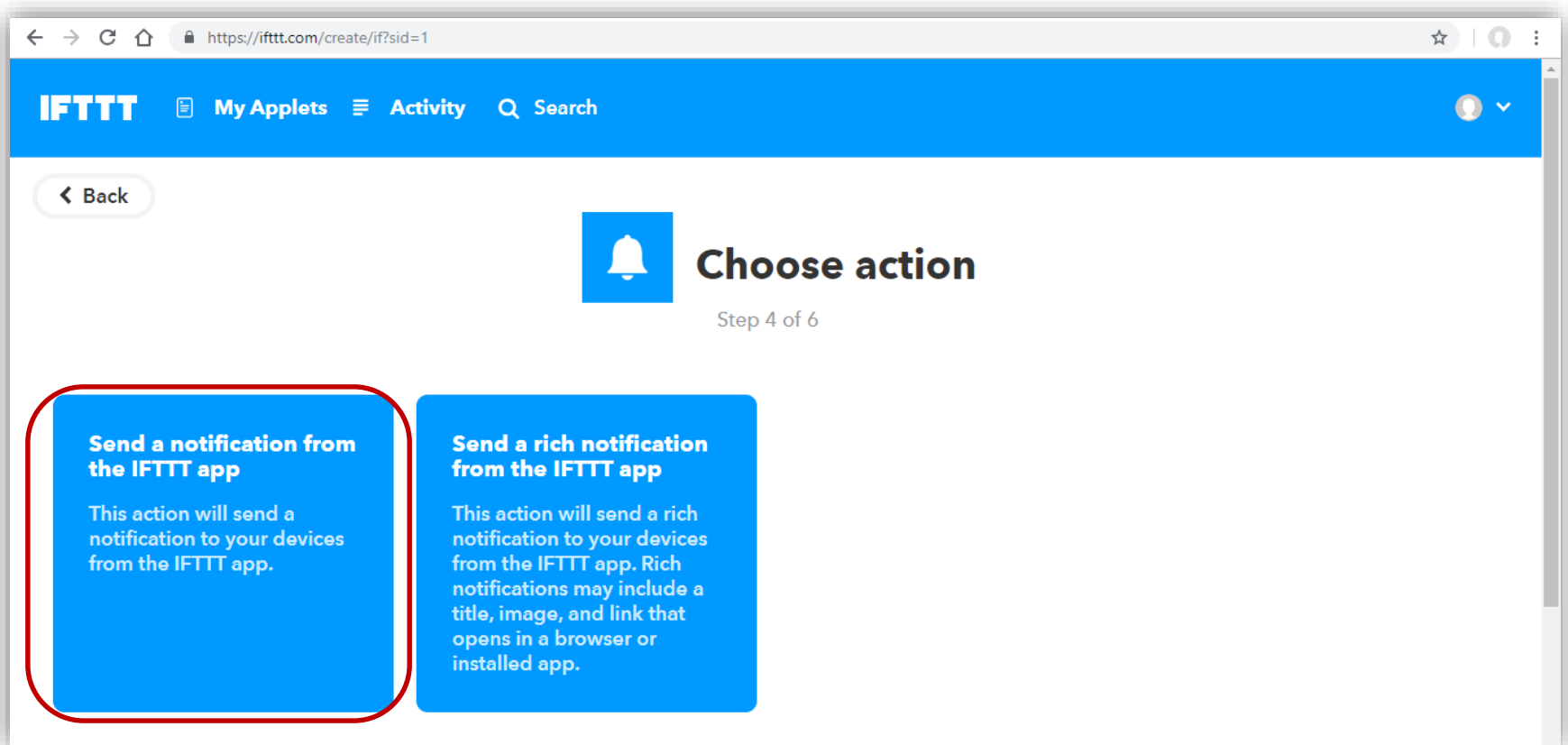
32

Concepts

HOW TO

Hands On

Let us **send a notification** from the **IFTTT** app



Adafruit IO + IFTTT



33

Concepts

HOW TO

Hands On


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**!

A screenshot of a web browser showing the IFTTT 'Complete action fields' screen. The browser address bar shows 'https://ifttt.com/create/if?sid=1'. The IFTTT logo and navigation links are at the top. The main heading is 'Complete action fields' with a bell icon and 'Step 5 of 6'. A blue box contains the title 'Send a notification from the IFTTT app' and a description. Below is a 'Message' field with a text template: 'FeedName FeedValue is Operator TriggerValue ! Created at CreatedAt'. An 'Add ingredient' button is next to the template. At the bottom is a 'Create action' button.

← → ↻ 🏠 <https://ifttt.com/create/if?sid=1> ☆ | ⓘ | ⋮

IFTTT 📄 My Applets ☰ Activity 🔍 Search 👤 ▾

⬅ Back

 **Complete action fields**
Step 5 of 6

Send a notification from the IFTTT app
This action will send a notification to your devices from the IFTTT app.

Message

FeedName FeedValue is
Operator TriggerValue !
Created at CreatedAt

Add ingredient

Create action

Adafruit IO + IFTTT



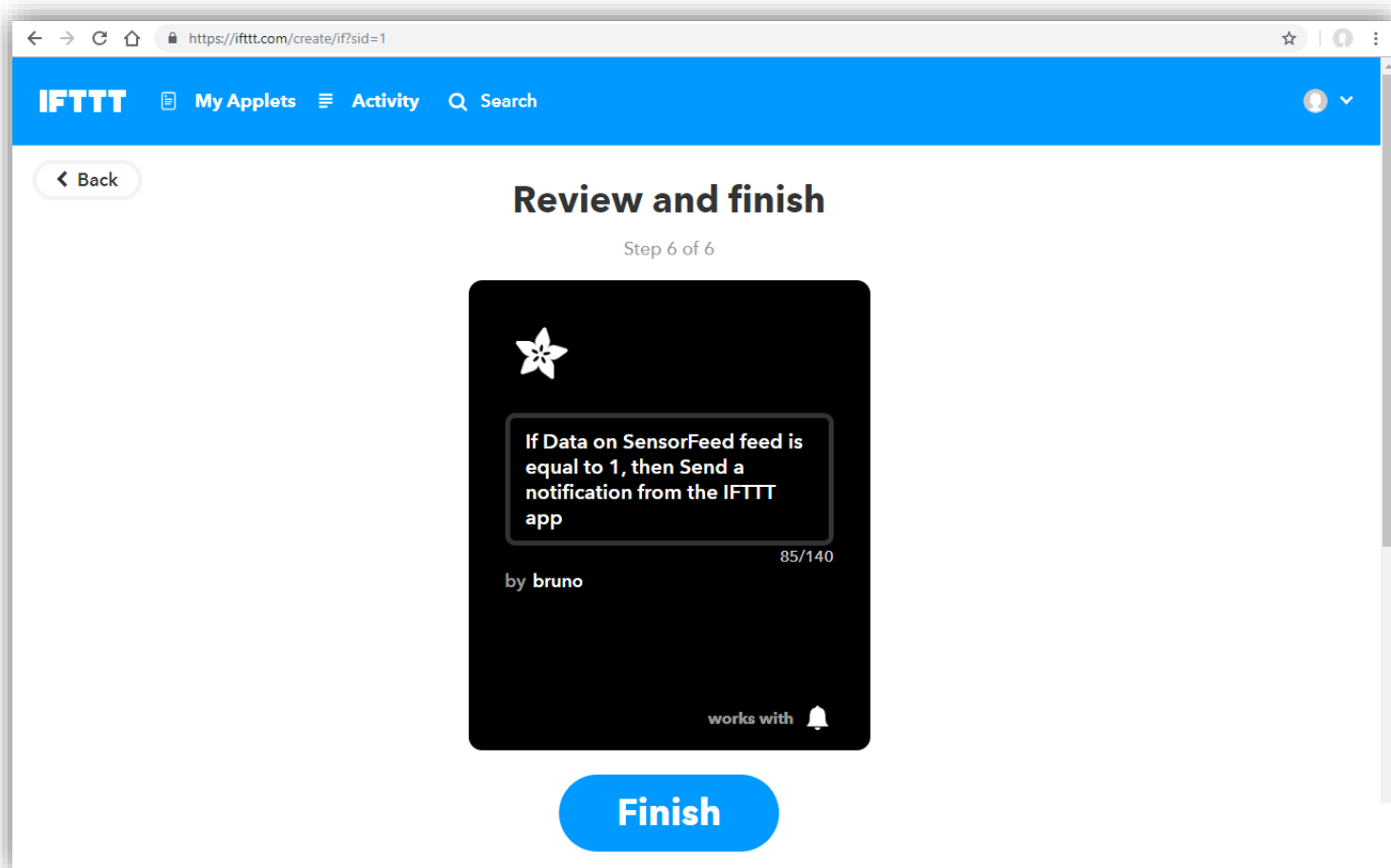
34

Concepts

HOW TO

Hands On

Review the created rule and **confirm** it!



Adafruit IO + IFTTT



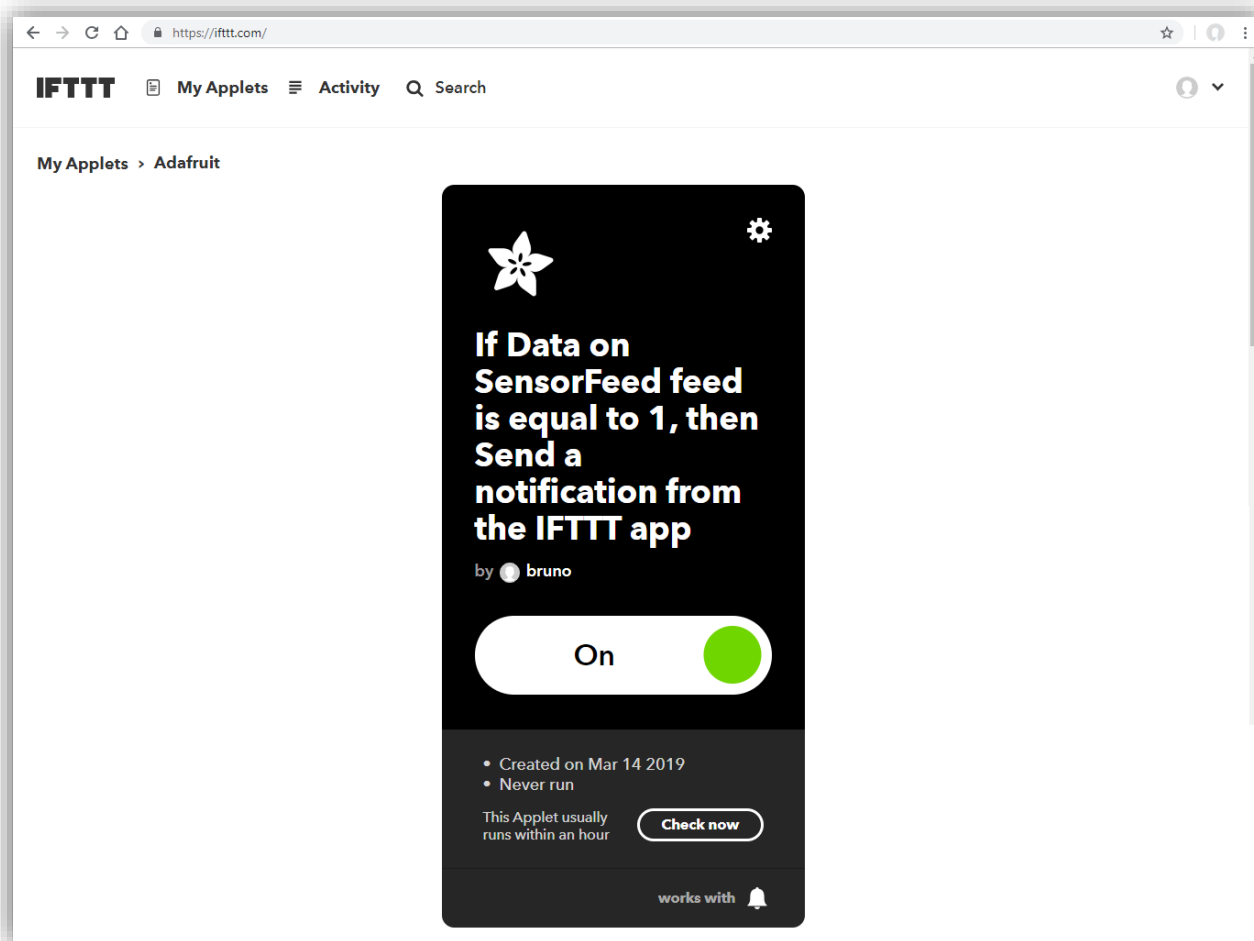
35

Concepts

HOW TO

Hands On

Activate it!



Adafruit IO + IFTTT



36

Concepts

HOW TO

Hands On

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

The screenshot shows the Adafruit IO web interface for the 'SensorFeed' feed. A modal window titled 'Add Data' is open, prompting the user to enter a 'VALUE'. The background shows a graph of the feed's data points over time, with the x-axis labeled 'Mar 14' and 'Mar 15'. The y-axis ranges from -1.0 to 1.0. On the right side, there are settings for the feed, including 'Feed Info', 'Privacy' (set to private), 'Sharing' (not shared yet), 'Feed History' (ON), 'Notifications' (Online), and 'Webhooks'. At the bottom left, there is a 'Free Usage' section showing 'Feeds: 1 of 10', 'Dashboards: 0 of 5', 'Rate: 30 / minute', 'Current Usage: 0 / min', and 'Storage: 30 days'. A red box highlights the '+ Add Data' button at the bottom left of the interface.

Adafruit IO + IFTTT



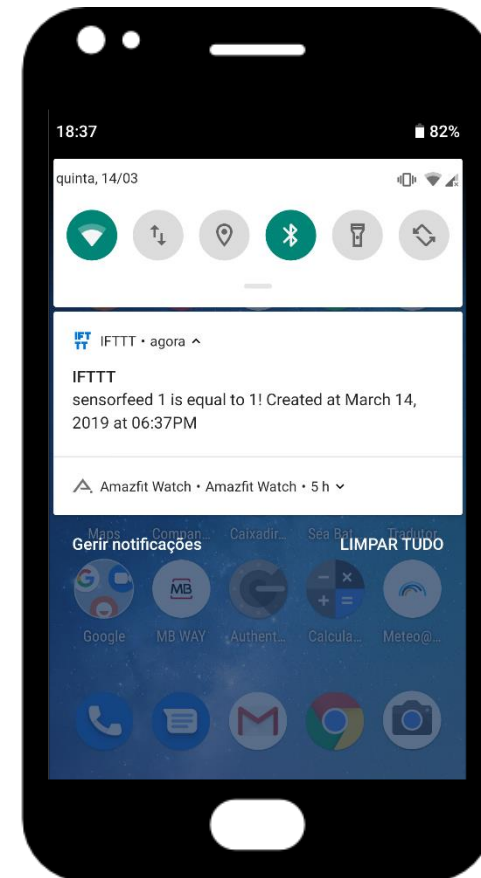
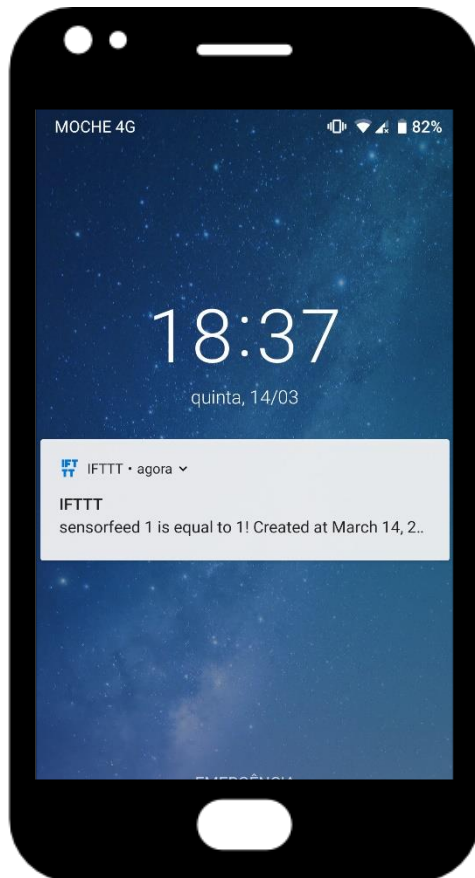
37

Concepts

HOW TO

Hands On

Notification received in the Smartphone!



Adafruit IO + IFTTT



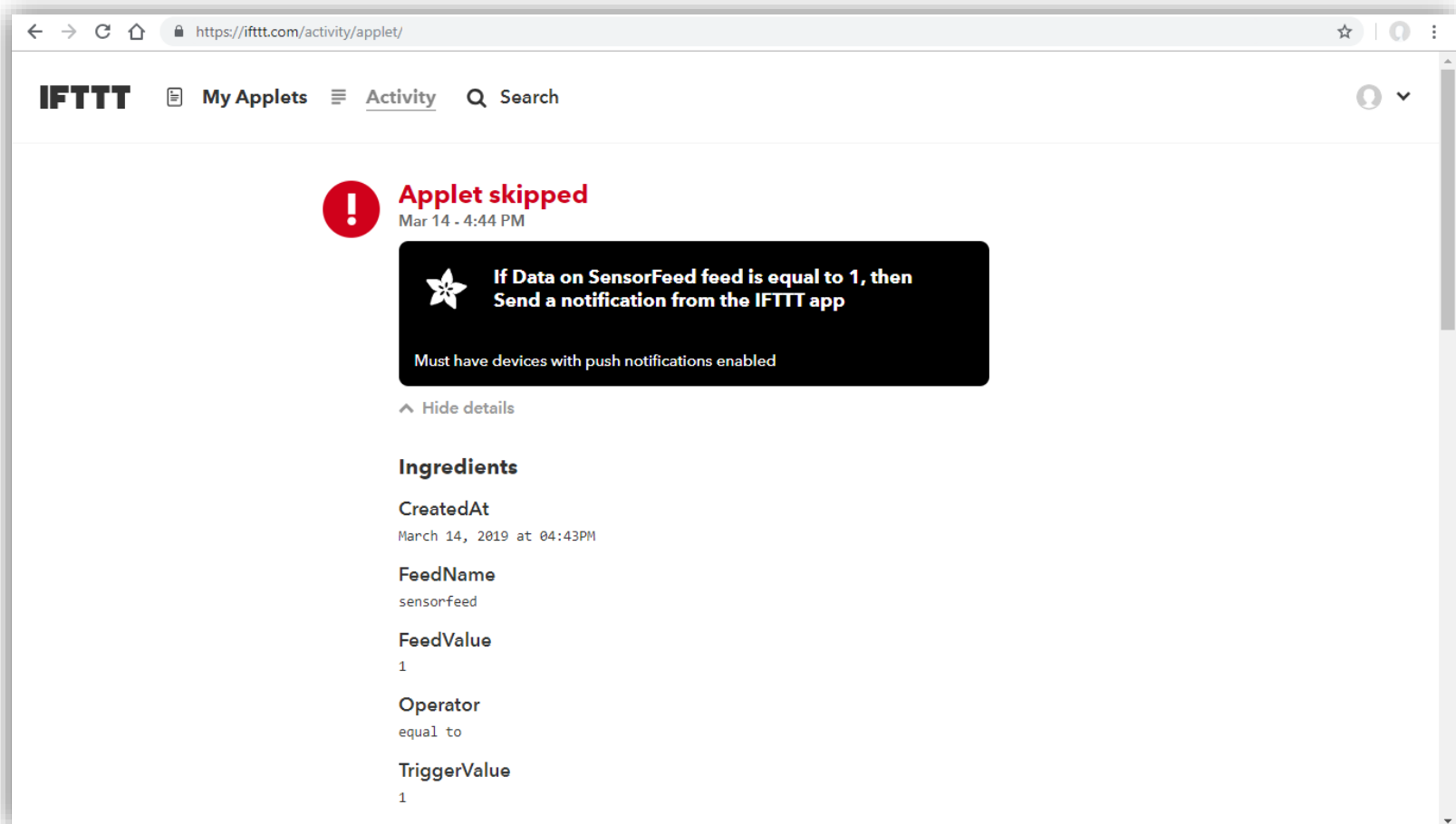
38

Concepts

HOW TO

Hands On

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



Adafruit IO + Java

39

Concepts

HOW TO

Hands On



Adafruit IO + Java



40

Concepts

HOW TO

Hands On

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



41

Concepts

HOW TO

Hands On

```
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



42

Concepts

HOW TO

Hands On

```
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



43

Concepts

HOW TO

Hands On

```
...
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

44

Concepts

HOW TO

Hands On



JavaScript

Adafruit IO + JavaScript



45

Concepts

HOW TO

Hands On

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



46

Concepts

HOW TO

Hands On

```
<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



47

Concepts

HOW TO

Hands On

```
//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



48

Concepts

HOW TO

Hands On

```
//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



49

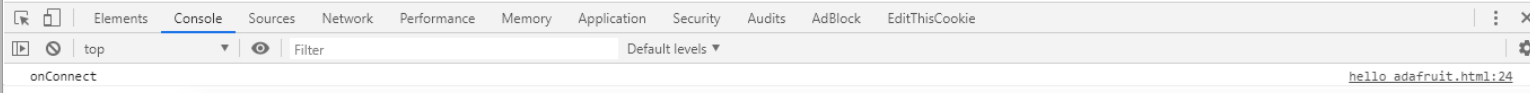
Concepts

HOW TO

Hands On

Test it!

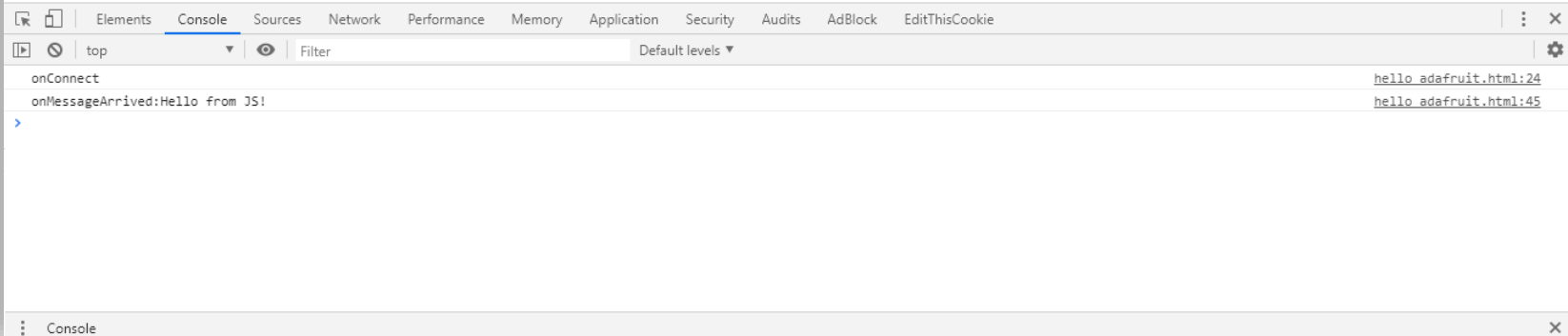
Publish



Test it!

Publish

Hello from JS!



Adafruit IO + Arduino

50

Concepts

HOW TO

Hands On



Adafruit IO + Arduino



51

Concepts

HOW TO

Hands On

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



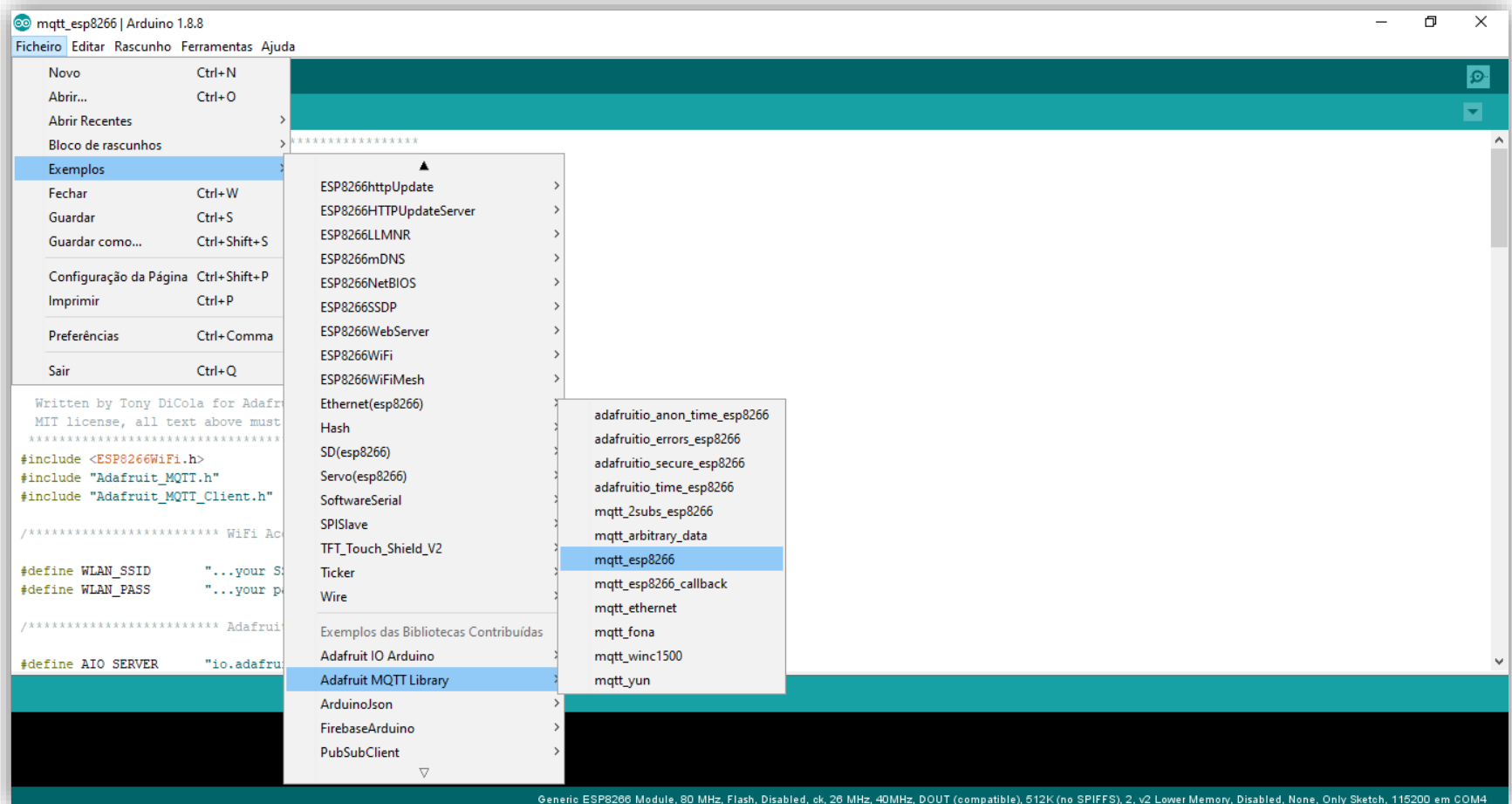
52

Concepts

HOW TO

Hands On

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



Adafruit IO + Arduino



53

Concepts

HOW TO

Hands On

```
#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 WiFiClientSecure 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



54

Concepts

HOW TO

Hands On

```
/****** 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



55

Concepts

HOW TO

Hands On

```
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



56

Concepts

HOW TO

Hands On

```
//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

57

Concepts

HOW TO

Hands On



Final Notes

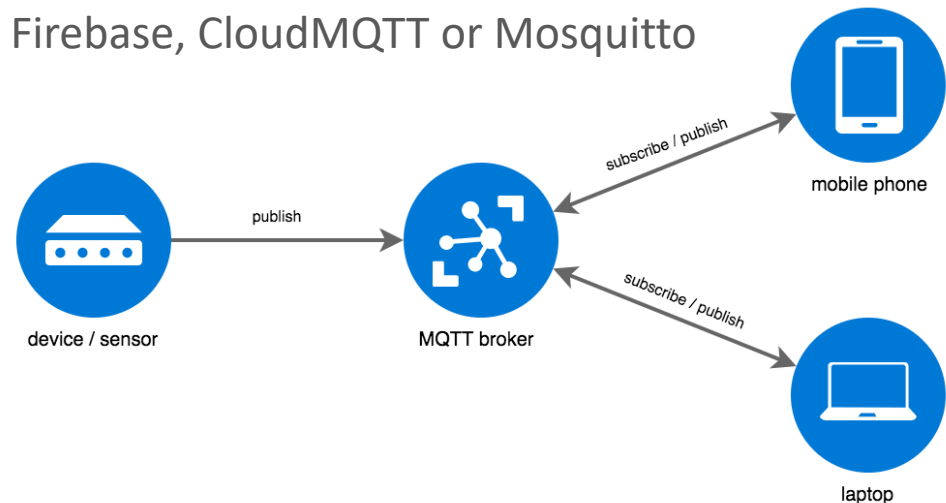
58

Concepts

HOW TO

Hands On

- 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

59

Concepts

HOW TO

Hands On

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



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>

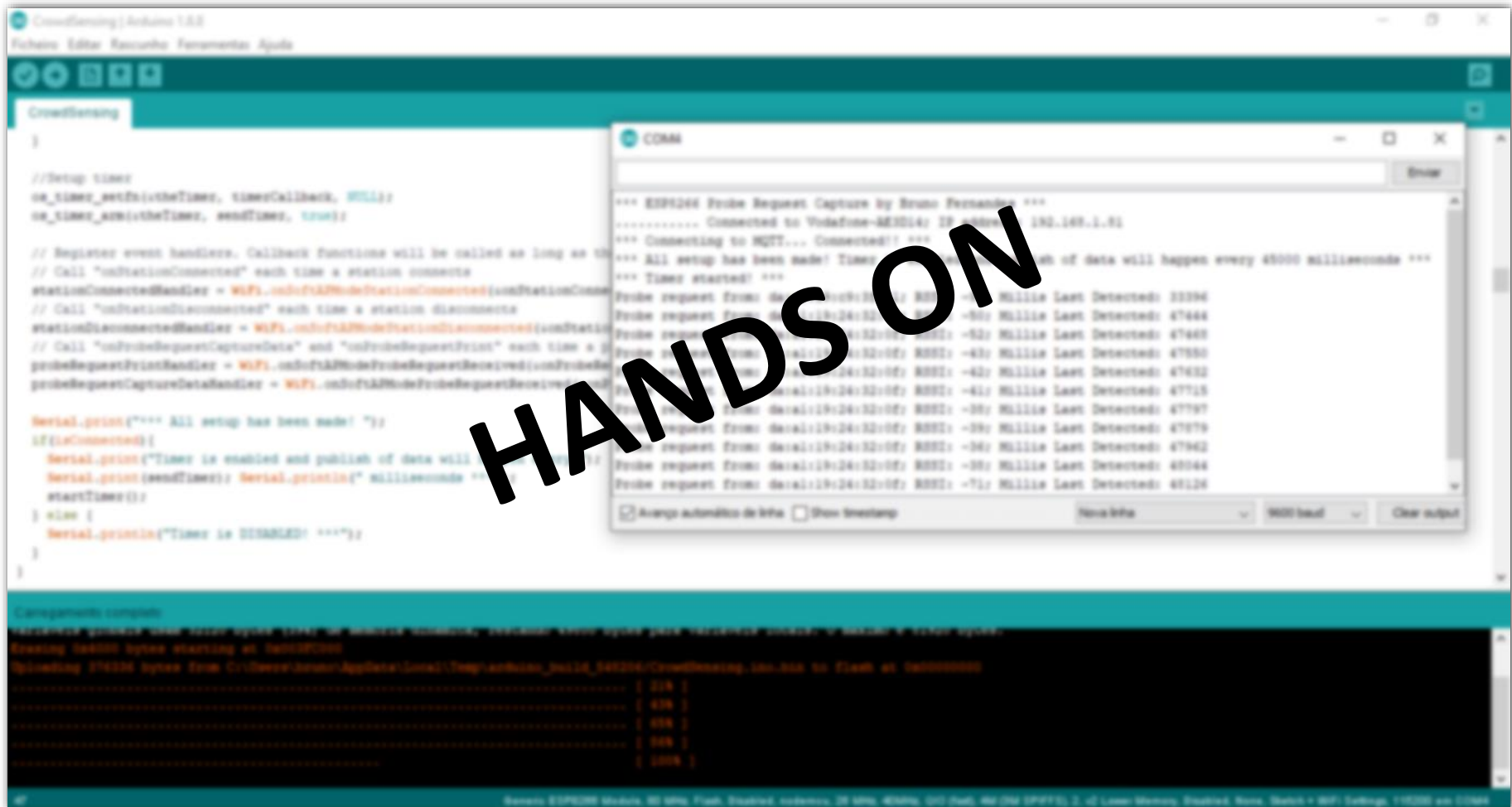
Hands On

61

Concepts

How To

HANDS ON



The screenshot shows the Arduino IDE interface with a project named "Crowdfensing". The code in the main editor includes setup functions for a timer and event handlers for station connections and probe requests. A large "HANDS ON" watermark is overlaid on the code. A serial monitor window is open, displaying the following output:

```
*** ESP8266 Probe Request Capture by Bruno Fernandez ***  
..... Connected to Vodafone-ME3D14: IP address: 192.168.1.81  
*** Connecting to MQTT... Connected! ***  
*** All setup has been made! Timer of data will happen every 45000 milliseconds ***  
*** Timer started! ***  
Probe request from dea1:19:24:32:0f: RSSI: -41; Millie Last Detected: 33396  
Probe request from dea1:19:24:32:0f: RSSI: -51; Millie Last Detected: 47444  
Probe request from dea1:19:24:32:0f: RSSI: -52; Millie Last Detected: 47449  
Probe request from dea1:19:24:32:0f: RSSI: -43; Millie Last Detected: 47550  
Probe request from dea1:19:24:32:0f: RSSI: -42; Millie Last Detected: 47632  
Probe request from dea1:19:24:32:0f: RSSI: -41; Millie Last Detected: 47715  
Probe request from dea1:19:24:32:0f: RSSI: -39; Millie Last Detected: 47797  
Probe request from dea1:19:24:32:0f: RSSI: -39; Millie Last Detected: 47879  
Probe request from dea1:19:24:32:0f: RSSI: -36; Millie Last Detected: 47942  
Probe request from dea1:19:24:32:0f: RSSI: -39; Millie Last Detected: 48044  
Probe request from dea1:19:24:32:0f: RSSI: -71; Millie Last Detected: 48126
```

At the bottom of the IDE, a progress bar shows the upload status: "Uploading 174334 bytes from C:\Users\bruno\AppData\Local\Temp\arduino_build_549296\Crowdfensing.ino.bin to Flash at 115200 baud".