

# Autonomous Systems

---

INTERNET OF THINGS

Fábio Silva  
fabiosilva@di.uminho.pt

# Index

---

Platforms

IFTTT

AdafruitIO

MQTT

AdafruitIO and IFTTT Tutorial

Java and AdafruitIO

Arduino and AdafruitIO

Exercises

# Platforms

---

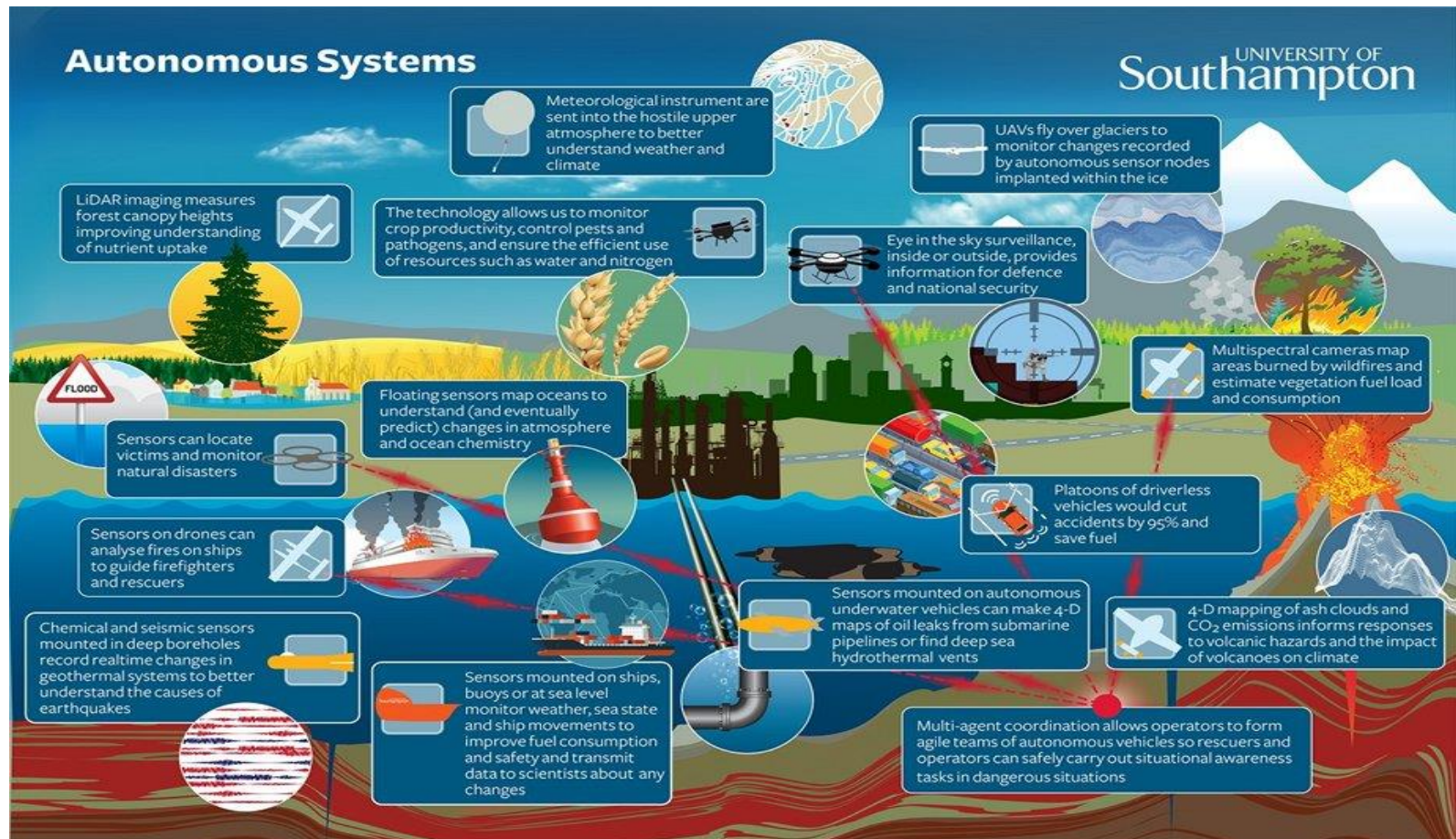
Actuator platform perform actions on the behalf of the user

They decide each action based on triggers, conditions that define a context, observed through services, sensors and inputs

Traditional platforms are reactive in nature and automate tasks.

Better implementations should make use of deliberative and hybrid processes

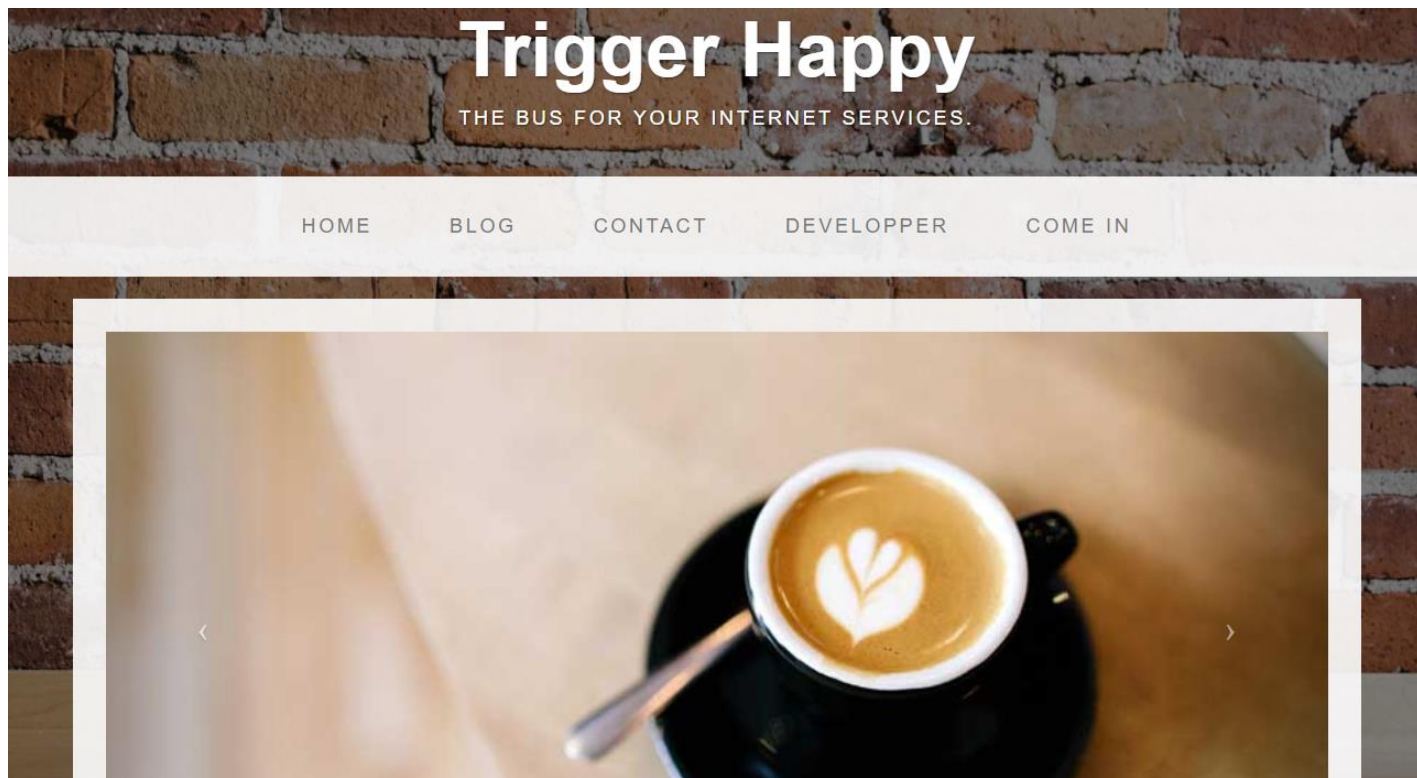
# Platforms



Source: <https://www.southampton.ac.uk/autonomous-systems/index.page>

# Platforms

---



<https://trigger-happy.eu/>

# Platforms

The screenshot shows the Microsoft Flow website homepage. At the top, there is a navigation bar with the Microsoft logo, links for 'Flow', 'Templates', 'Connectors', and 'Learn', a search bar for templates, and 'Sign in' and 'Sign up free' buttons. The main heading is 'Work less, do more' in large white text on a blue background. Below this, a subheading reads: 'Create automated workflows between your favorite apps and services to get notifications, synchronize files, collect data, and more'. A button labeled 'See how it works' is positioned below the subheading. A search bar with the placeholder text 'Find a template or connector to start with' is located below the button. In the bottom left, there is a smartphone displaying a list of flows, including 'Get a notification when #Contoso gets tweeted', 'When a file is added in Dropbox, upload it to SharePoint and notify team via Slack', and 'Set up project essentials'. To the right of the phone is a laptop displaying a workflow editor. The workflow is titled 'Get a Slack message and email for new Dropbox item'. It consists of two steps: 'Dropbox - When a file is created' and 'Slack - Post message'. The 'Dropbox' step has a 'Choose a folder' dropdown set to 'Motion'. The 'Slack' step has a 'Channel name' dropdown set to 'FromDropbox' and a 'Message text' field containing 'File name is added to Dropbox'.

Microsoft | Flow | Templates | Connectors | Learn | Search templates ... | Sign in | Sign up free

## Work less, do more

Create automated workflows between your favorite apps and services to get notifications, synchronize files, collect data, and more

See how it works

Find a template or connector to start with

Flow | My flows | Activity | Browse | Learn | Search templates

Get a Slack message and email for new Dropbox item

Dropbox - When a file is created

Choose a folder

Motion

Slack - Post message

Channel name

FromDropbox

Message text

File name is added to Dropbox

<https://flow.microsoft.com/en-us/>

# Platforms

IFTTT

## 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.

**Get Started**



<https://ifttt.com/>

# IFTTT

---

## IFTTT - If This Then That

- Service to create chains of simple conditional statements, called applets
- Is triggered by changes that occur within other web services
- After a trigger executes an actionable service in the platform





# AdafruitIO

---



# MQTT

---

- MQTT (Message Queue Telemetry Transport) was originally developed out of IBM's pervasive computing team and their work with partners in the industrial sector.
- Lightweight message protocol:
  - Connecting to a server only takes about 80 bytes
  - Push data from server to device is about 20 bytes
- MQTT messages are sent to feeds in an MQTT broker such as AdafruitIO which then distributes them through the devices which subscribed feeds
- <http://mqtt.org/>



# AdafruitIO and IFTTT Tutorial

---

In this tutorial we will develop an IFTTT applet that reacts to values in AdafruitIO feeds

The objective is to monitor sensor values sent to AdafruitIO feeds and take actions in some contexts

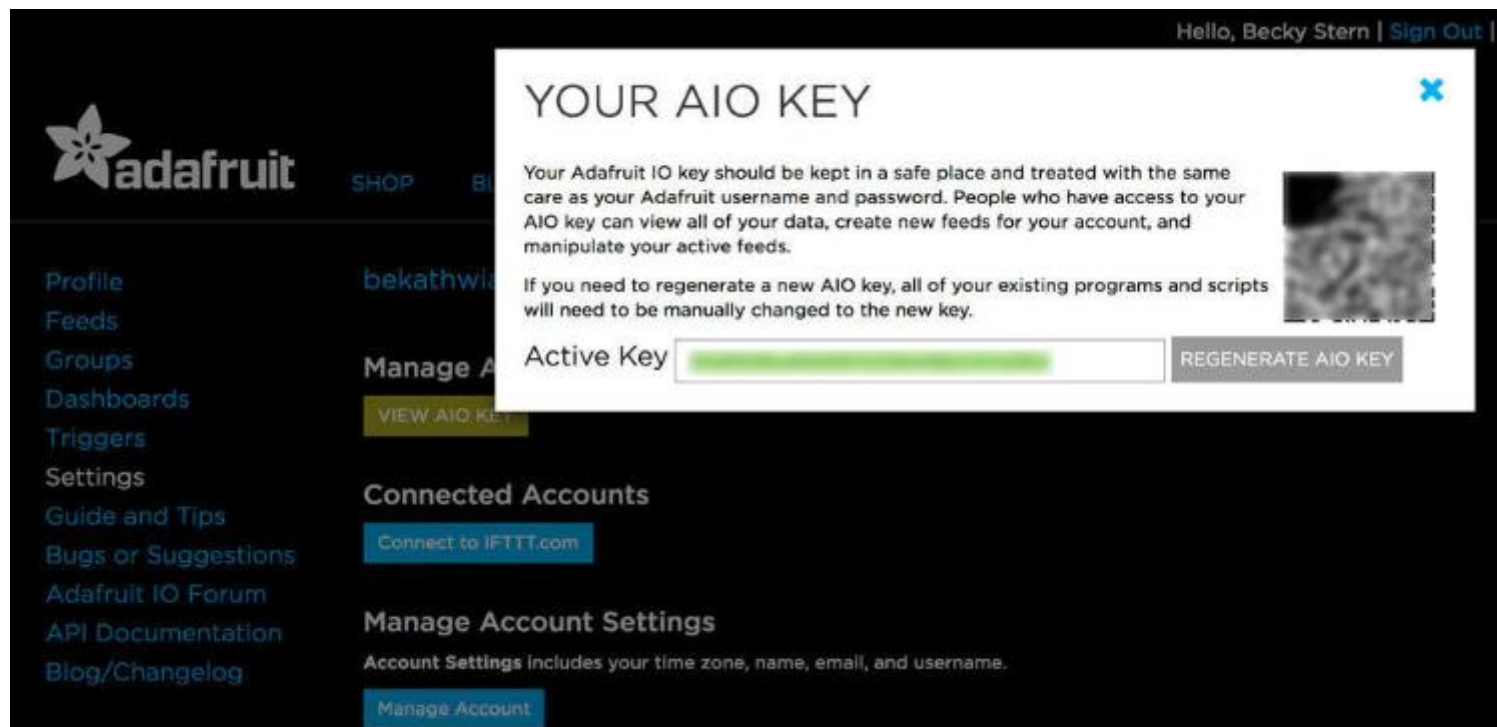
In order to complete this tutorial it is needed:

- An IFTTT account
- An AdafruitIO account
- A smartphone with the IFTTT application installed

# AdafruitIO and IFTTT Tutorial

---

Take note of your username and AIO Key

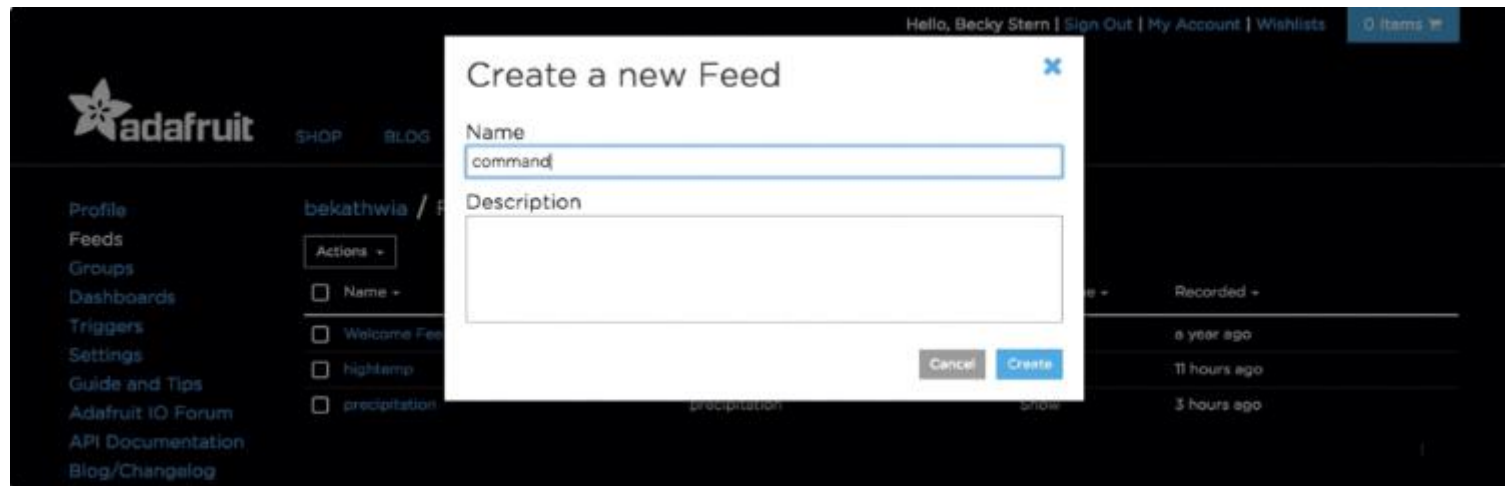


The screenshot displays the AdafruitIO web interface. On the left is a dark sidebar with the Adafruit logo and a list of navigation links: Profile, Feeds, Groups, Dashboards, Triggers, Settings, Guide and Tips, Bugs or Suggestions, Adafruit IO Forum, API Documentation, and Blog/Changelog. The main content area has a dark background. At the top right, it says 'Hello, Becky Stern | Sign Out |'. A white modal window titled 'YOUR AIO KEY' is centered on the screen. Inside the modal, there is a warning: 'Your Adafruit IO key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your AIO key can view all of your data, create new feeds for your account, and manipulate your active feeds.' Below this, it states: 'If you need to regenerate a new AIO key, all of your existing programs and scripts will need to be manually changed to the new key.' To the right of the text is a small image of a QR code. At the bottom of the modal, there is a label 'Active Key' followed by a green rectangular field containing a blurred key. To the right of this field is a button labeled 'REGENERATE AIO KEY'. In the background, behind the modal, several sections are visible: 'Manage Account' with a 'VIEW AIO KEY' button, 'Connected Accounts' with a 'Connect to IFTTT.com' button, and 'Manage Account Settings' with a 'Manage Account' button.

# AdafruitIO and IFTTT Tutorial

---

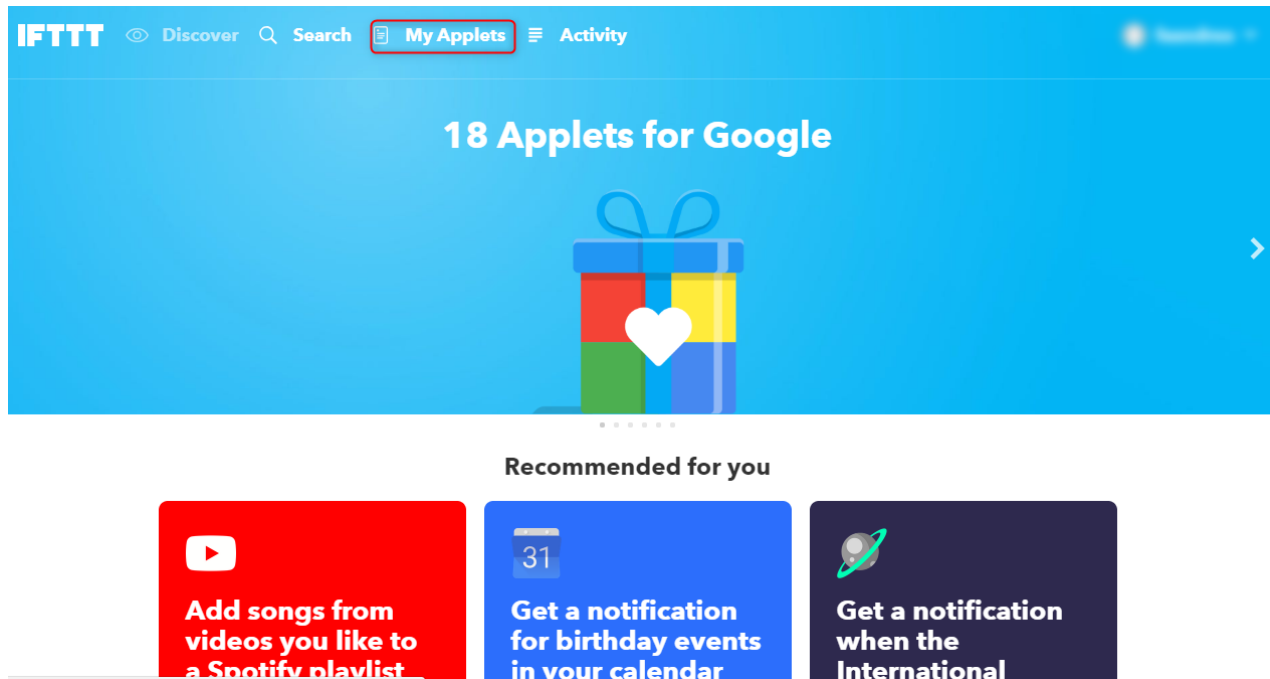
Create a new feed named command in adafruitIO



# AdafruitIO and IFTTT Tutorial

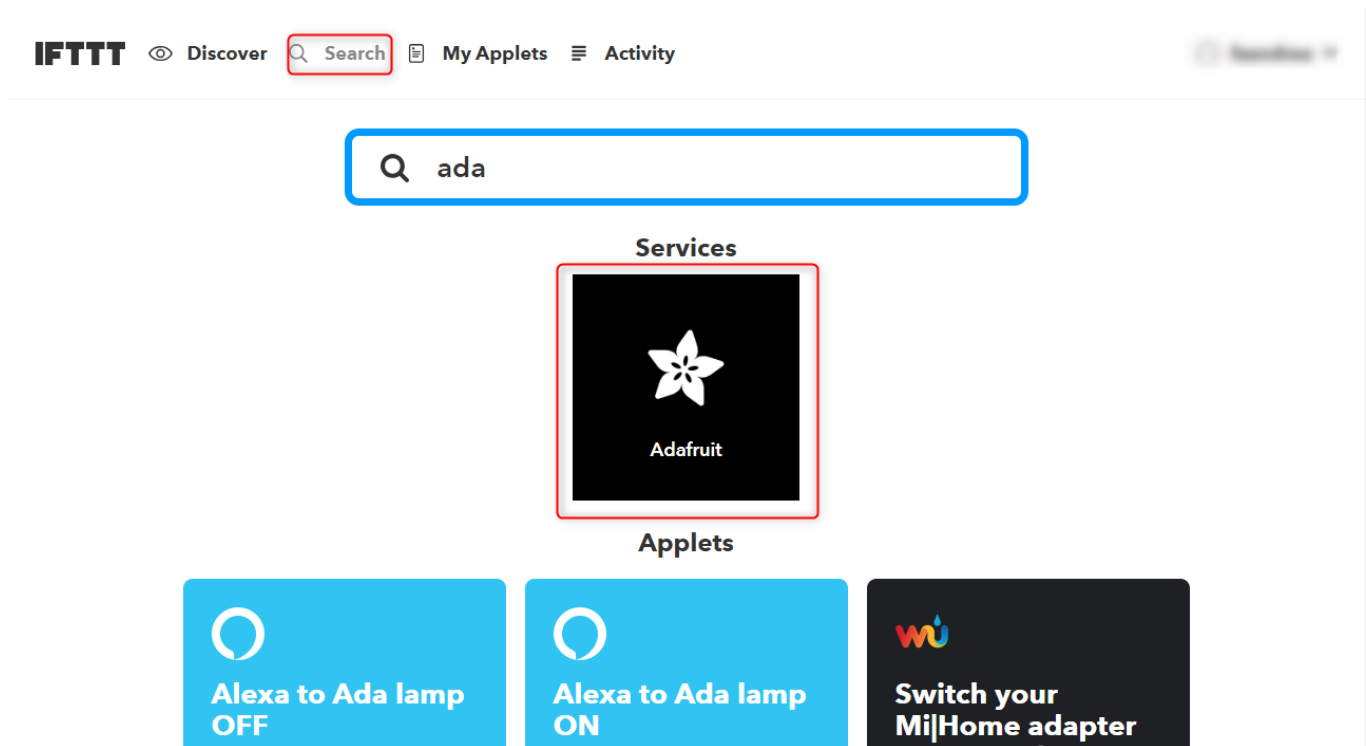
---

Log in to the IFTTT platform



# AdafruitIO and IFTTT Tutorial

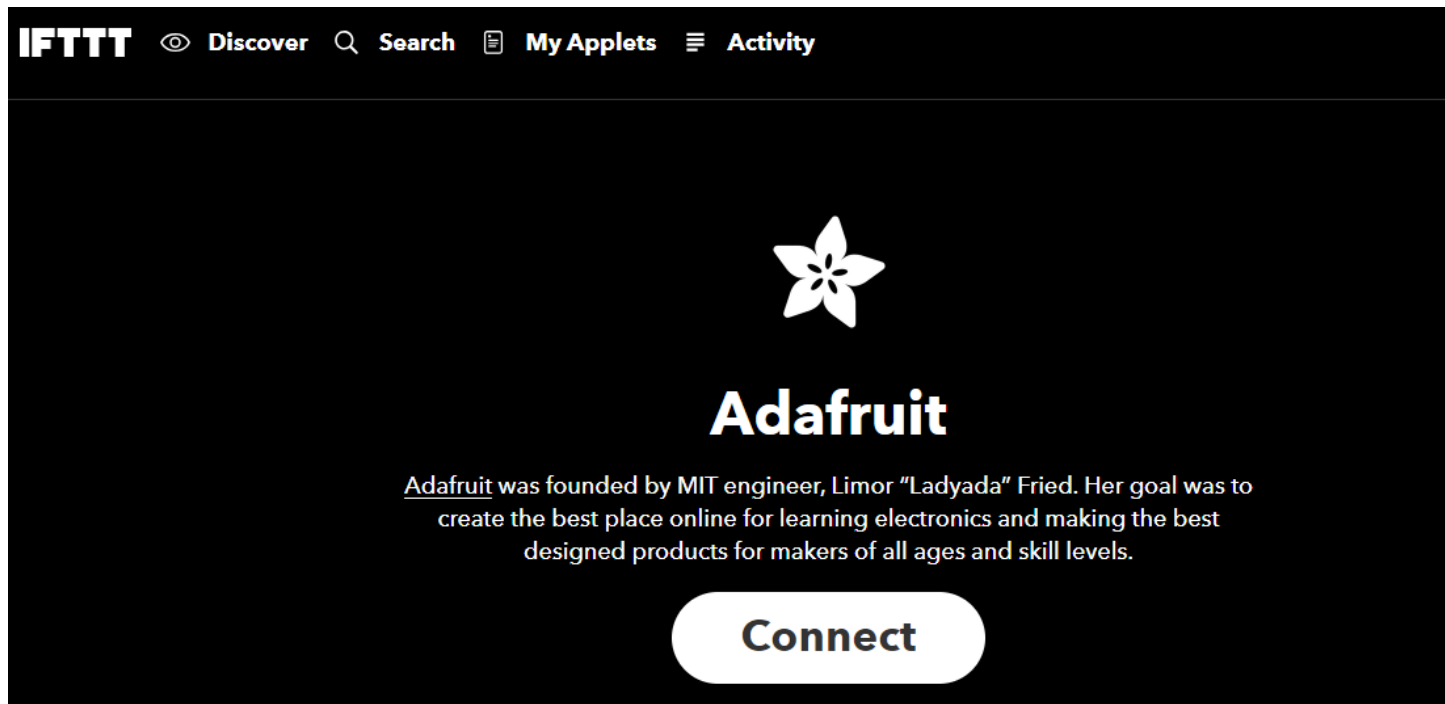
Search for the Adafruit service



# AdafruitIO and IFTTT Tutorial

---

Connect IFTTT to the Adafruit platform

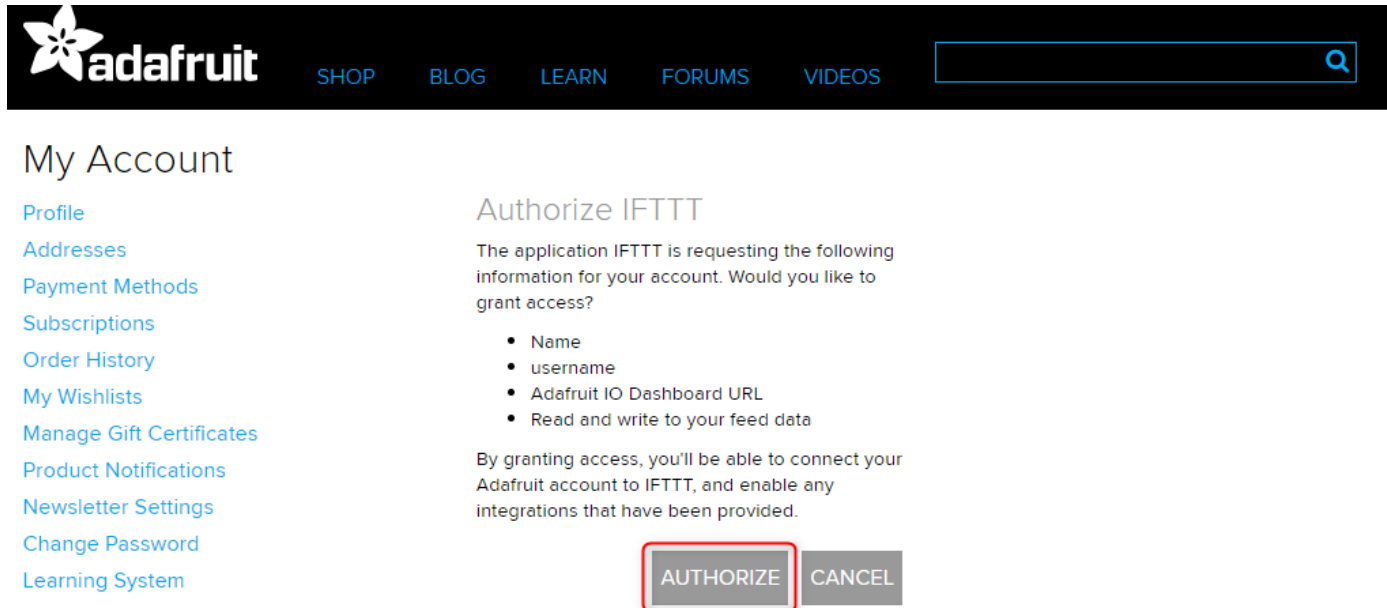




# AdafruitIO and IFTTT Tutorial

---

## Authorize IFTTT access



The screenshot shows the Adafruit website's 'My Account' page. The top navigation bar is black with the Adafruit logo and links for SHOP, BLOG, LEARN, FORUMS, and VIDEOS. A search bar is on the right. The 'My Account' section on the left lists various account management options. The main content area displays the 'Authorize IFTTT' dialog, which explains that the application is requesting access to the user's account information. A list of requested permissions is shown, and a note states that granting access will allow the user to connect their Adafruit account to IFTTT. At the bottom, there are two buttons: 'AUTHORIZE' (highlighted with a red box) and 'CANCEL'.

**adafruit** SHOP BLOG LEARN FORUMS VIDEOS

### My Account

- [Profile](#)
- [Addresses](#)
- [Payment Methods](#)
- [Subscriptions](#)
- [Order History](#)
- [My Wishlists](#)
- [Manage Gift Certificates](#)
- [Product Notifications](#)
- [Newsletter Settings](#)
- [Change Password](#)
- [Learning System](#)

### Authorize IFTTT

The application IFTTT is requesting the following information for your account. Would you like to grant access?

- Name
- username
- Adafruit IO Dashboard URL
- Read and write to your feed data

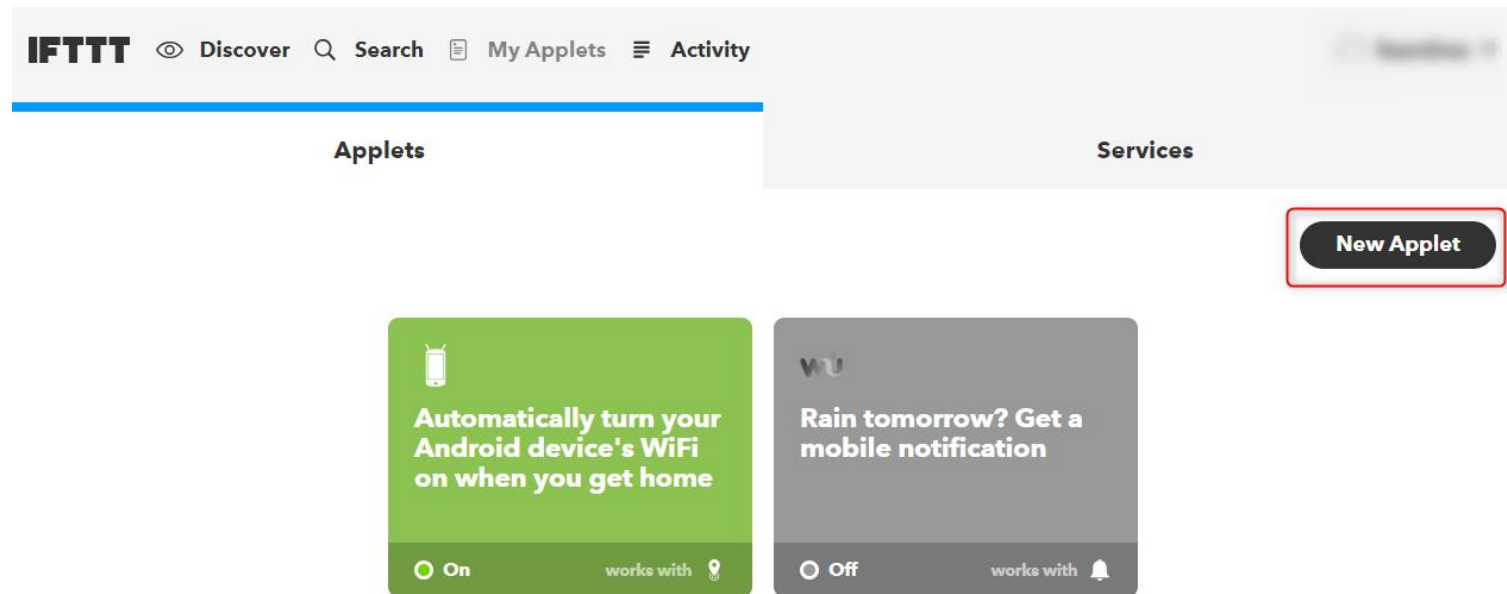
By granting access, you'll be able to connect your Adafruit account to IFTTT, and enable any integrations that have been provided.

**AUTHORIZE** CANCEL

# AdafruitIO and IFTTT Tutorial

---

On the IFTTT platform lets create a new applet



# AdafruitIO and IFTTT Tutorial

---

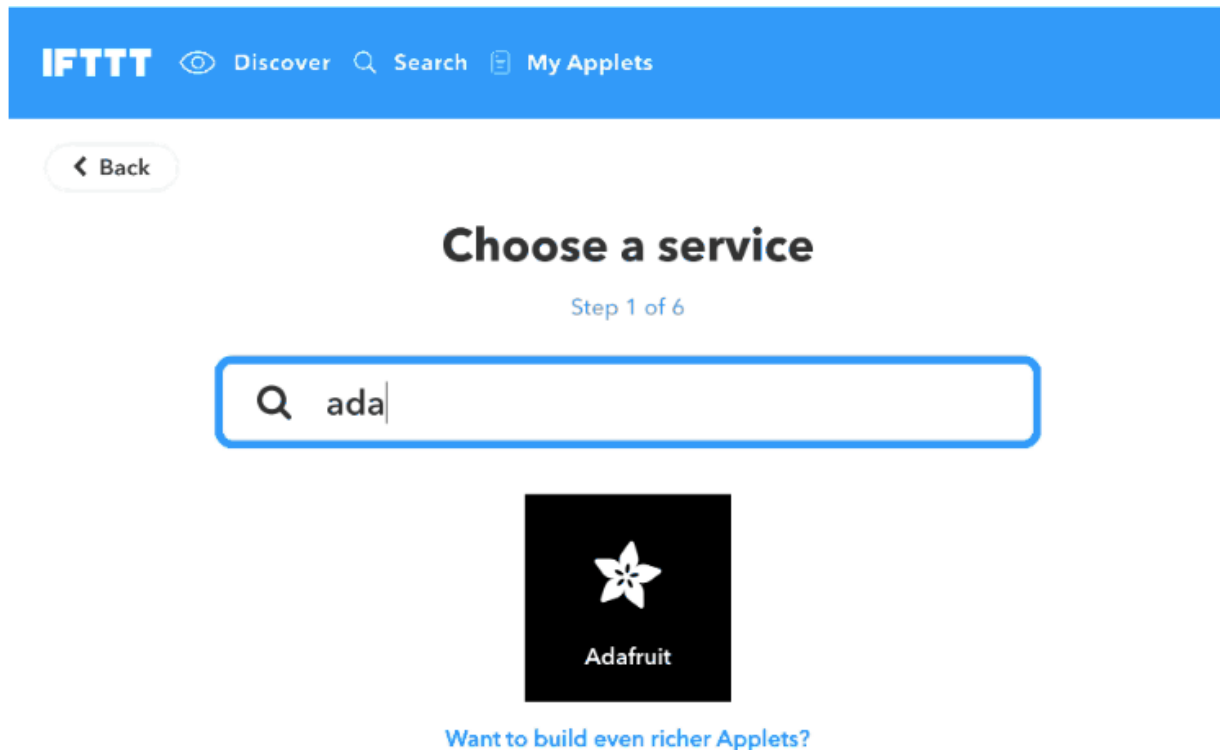
**New Applet**

**if**  **this** **then that**

# AdafruitIO and IFTTT Tutorial

---

Choose the adafruit service



The screenshot shows the IFTTT interface for selecting a service. At the top is a blue navigation bar with the IFTTT logo, an eye icon, and links for 'Discover', 'Search', and 'My Applets'. Below this is a 'Back' button. The main heading is 'Choose a service', followed by 'Step 1 of 6'. A search bar contains the text 'ada'. Below the search bar is a black square button with a white Adafruit logo (a stylized flower) and the text 'Adafruit' underneath. At the bottom, there is a link that says 'Want to build even richer Applets?'.

# AdafruitIO and IFTTT Tutorial

**IFTTT** Discover Search My Applets

[← 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**

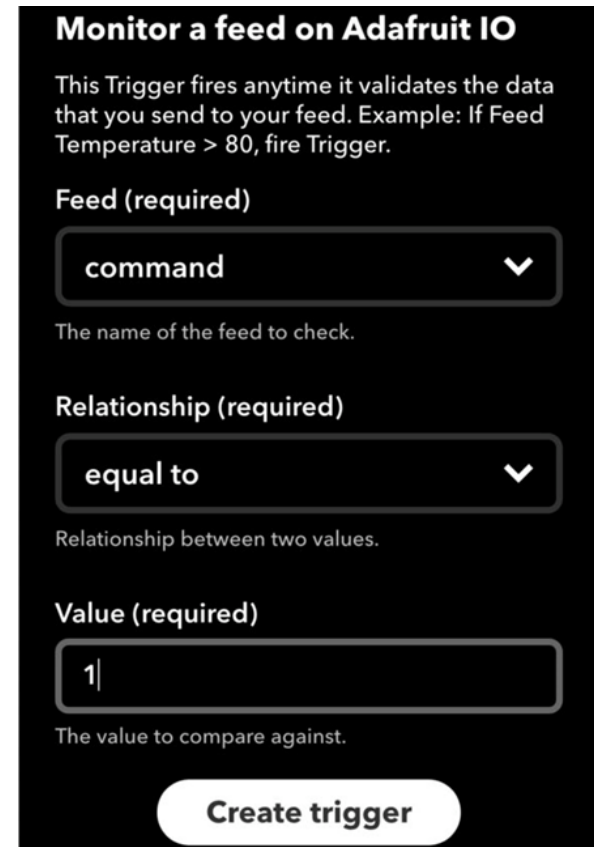
This Trigger fires any time there is new data in your feed.

# AdafruitIO and IFTTT Tutorial

---

Select the command feed created earlier

Create a trigger after the value 1 is sent to the command feed



**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.

**Feed (required)**

command

The name of the feed to check.

**Relationship (required)**

equal to

Relationship between two values.

**Value (required)**

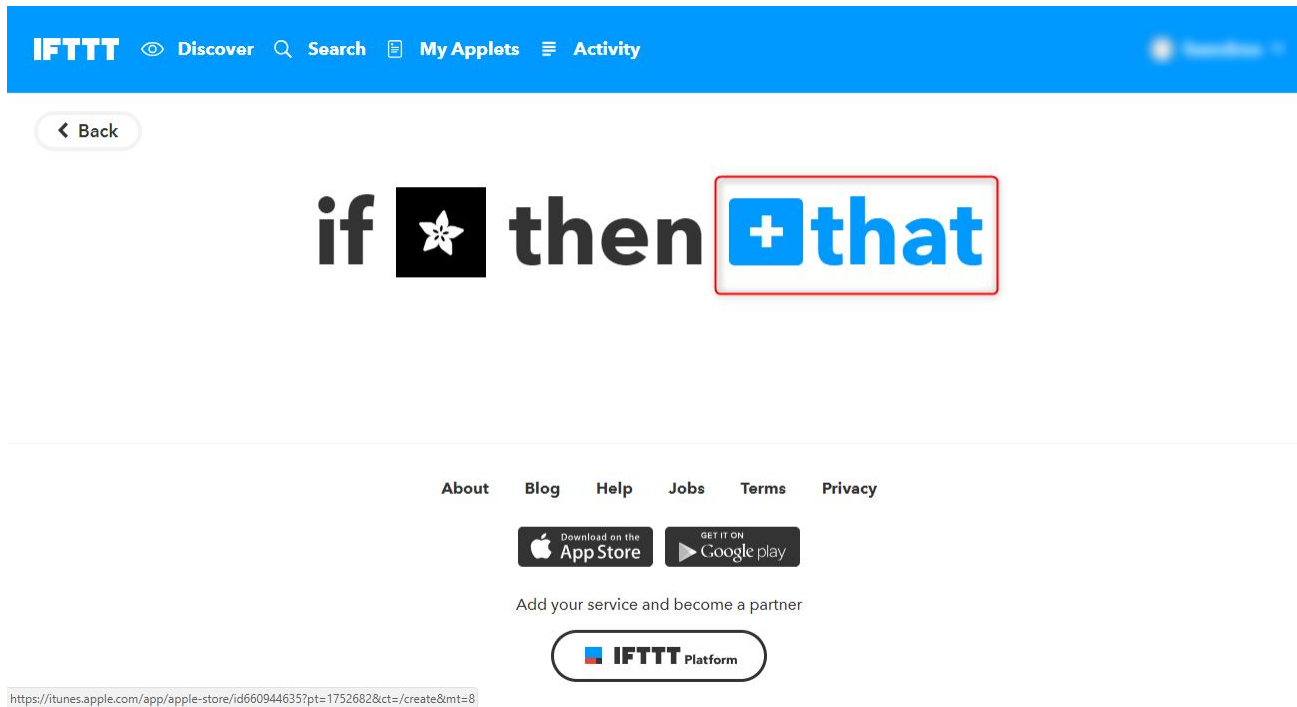
1

The value to compare against.

**Create trigger**

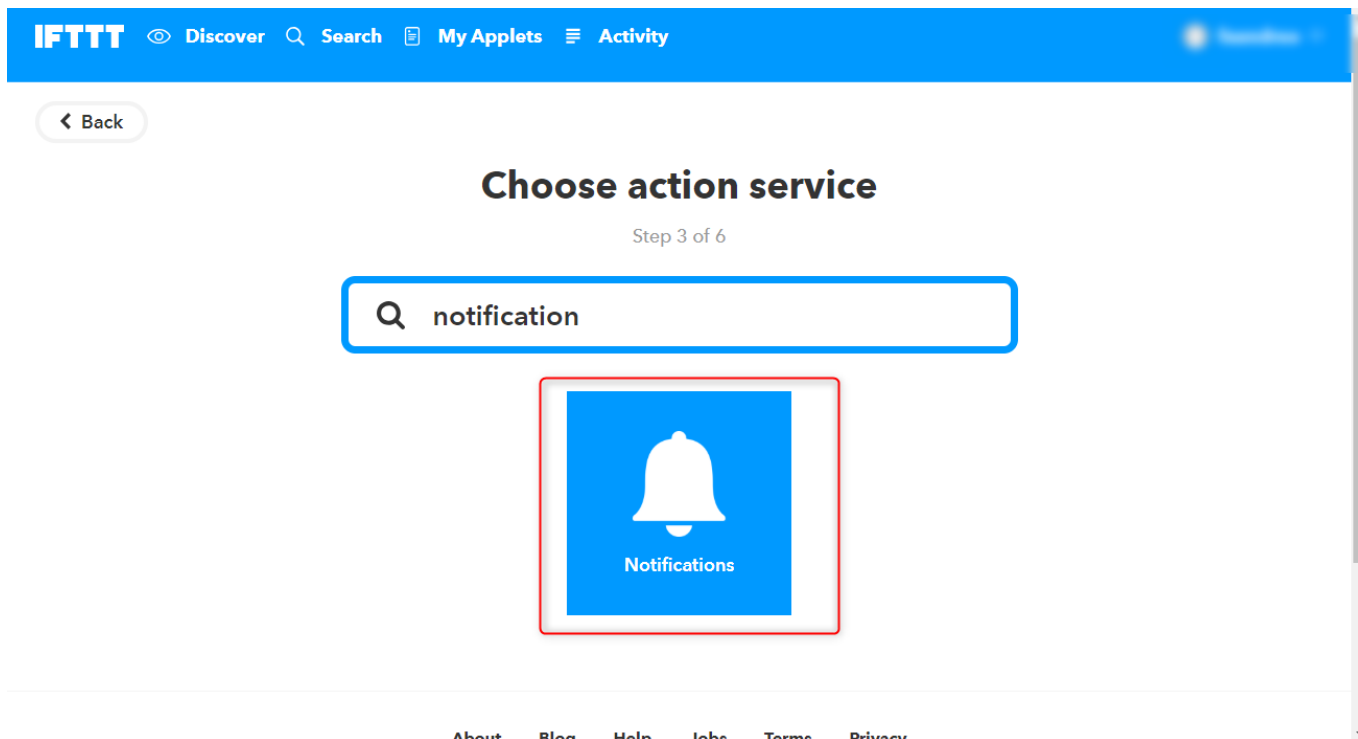
# AdafruitIO and IFTTT Tutorial

---



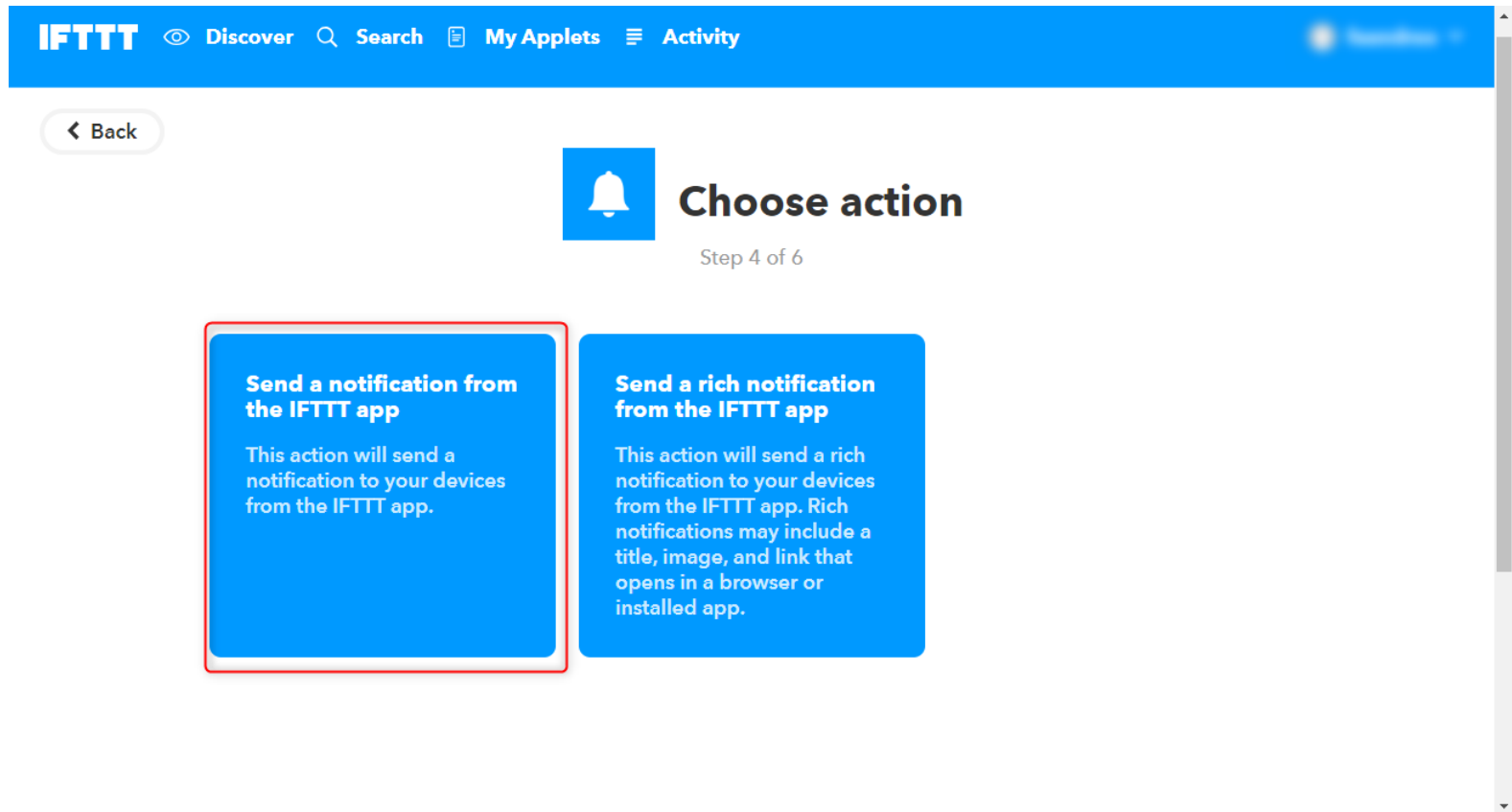
# AdafruitIO and IFTTT Tutorial

Choose a notification service for the action






# AdafruitIO and IFTTT Tutorial



The screenshot shows the IFTTT web interface. At the top is a blue navigation bar with the IFTTT logo, links for 'Discover', 'Search', 'My Applets', and 'Activity', and a user profile icon. Below the navigation bar is a 'Back' button. The main content area is titled 'Choose action' with a bell icon and 'Step 4 of 6'. There are two blue action cards. The first card, 'Send a notification from the IFTTT app', is highlighted with a red border. The second card is 'Send a rich notification from the IFTTT app'.

**IFTTT** Discover Search My Applets Activity

◀ Back

 **Choose action**  
Step 4 of 6

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

**Send a rich notification from the IFTTT app**  
This action will send a rich notification to your devices from the IFTTT app. Rich notifications may include a title, image, and link that opens in a browser or installed app.

# AdafruitIO and IFTTT Tutorial

---

Create a custom notification to display in device where the IFTTT app is installed

Inside the double brackets “{{ ... }}”, are the values obtained from the trigger we created earlier



## 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

Alerta!!  
{{FeedName}} {{FeedValue}}  
{{Operator}} {{TriggerValue}}!

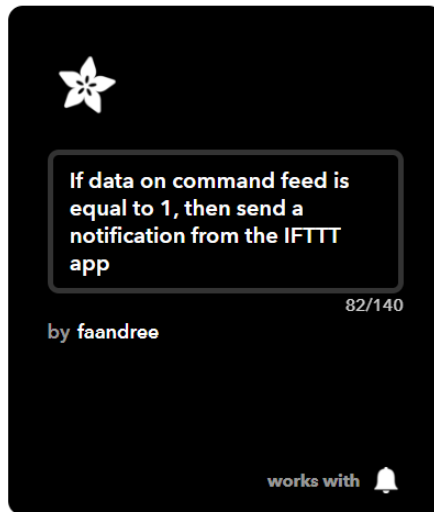
Add ingredient

Create action

# AdafruitIO and IFTTT Tutorial

---

Step 6 of 6



**Finish**

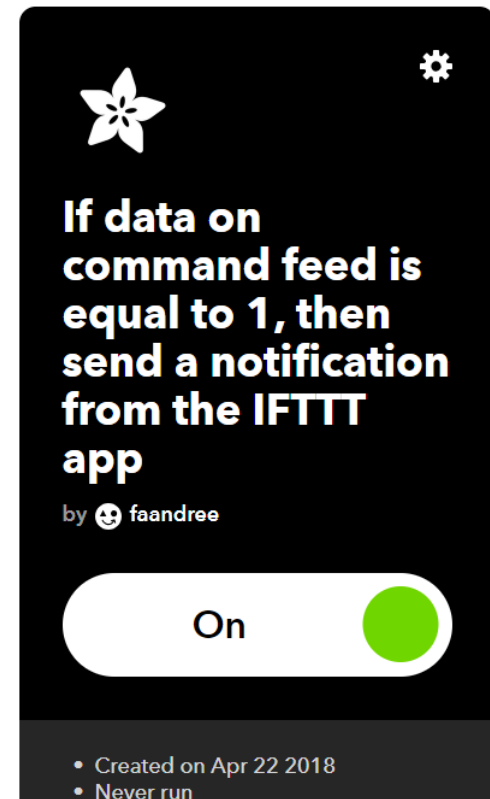
Review the rule created

If all is well we can finish the process

# AdafruitIO and IFTTT Tutorial

---

Active the rule created



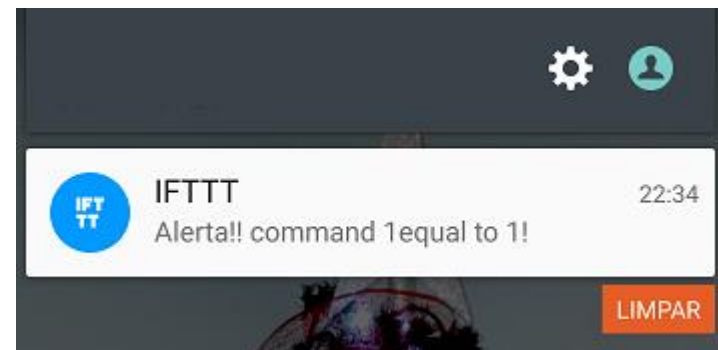
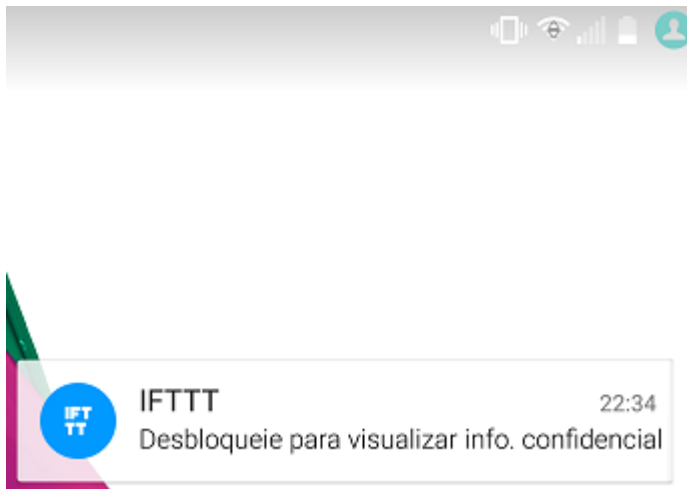
# AdafruitIO and IFTTT Tutorial



# AdafruitIO and IFTTT Tutorial

---

Notification on the smartphone



# Java and AdafruitIO

---



# Java and AdafruitIO

---

A sample Java Program to test interaction between AdafruitIO and a Java Program

```
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 {

    public static void main(String[] args) {
        String topic    = "topic";
        String content   = "Message from MqttPublishSample";
        int qos          = 2;
        String broker    = "tcp://io.adafruit.com:1883";
        String clientId  = "JavaSample";
        MemoryPersistence persistence = new MemoryPersistence();
        (...)
```



# Java and AdafruitIO

---

```
(...)  
  
try {  
    MqttClient sampleClient = new MqttClient(broker, clientId, persistence);  
    MqttConnectOptions connOpts = new MqttConnectOptions();  
    connOpts.setCleanSession(true);  
    connOpts.setUserName("username");  
    connOpts.setPassword("key".toCharArray());  
    connOpts.setSSLProperties(new Properties());  
    System.out.println("Connecting to broker: "+broker);  
    sampleClient.connect(connOpts);  
    System.out.println("Connected");  
    System.out.println("Publishing message: "+content);  
    MqttMessage message = new MqttMessage(content.getBytes());  
    message.setQos(qos);  
    sampleClient.publish(topic, message);  
    System.out.println("Message published");  
    sampleClient.disconnect();  
    System.out.println("Disconnected");  
    System.exit(0);  
}  
  
(...)
```

# Java and AdafruitIO

---

```
                                (...)  
    } 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();  
    }  
}
```

# Java and AdafruitIO

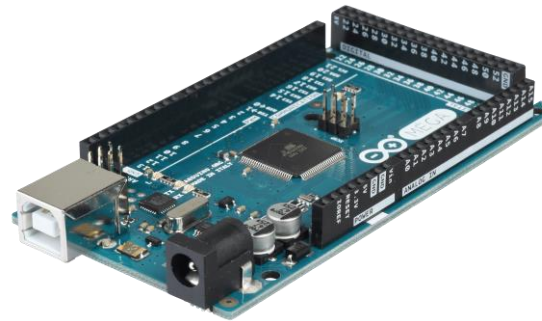
---

## References:

- <https://www.eclipse.org/paho>
- <http://wiki.eclipse.org/Paho>
- <http://www.eclipse.org/paho/clients/java>

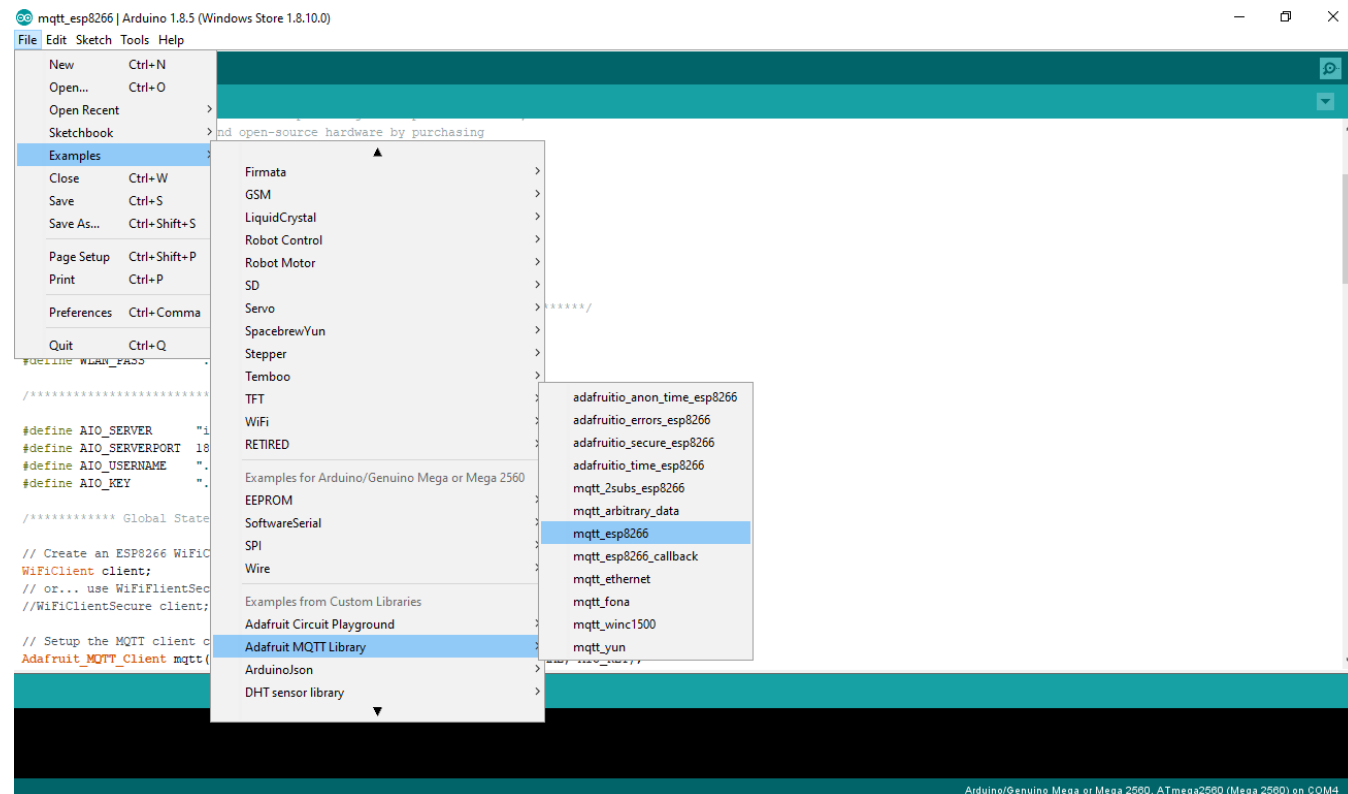
# Arduino and AdafruitIO

---



# Arduino and AdafruitIO

## Use an arduino studio IDE example to get started



# Arduino and AdafruitIO

---

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

/***** WiFi Access Point *****/

#define WLAN_SSID      "...your SSID..."
#define WLAN_PASS      "...your password..."

/***** Adafruit.io Setup *****/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT 1883           // use 8883 for SSL
#define AIO_USERNAME    "...your AIO username (see https://accounts.adafruit.com)..."
#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 and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
```

# Arduino and AdafruitIO

```
....  
/***** Feeds *****/  
Adafruit_MQTT_Publish commandpublish = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/command");  
Adafruit_MQTT_Subscribe commandsubscribe = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/command");  
/***** Sketch Code *****/  
// Bug workaround for Arduino 1.6.6, it seems to need a function declaration  
// for some reason (only affects ESP8266, likely an arduino-builder bug).  
void MQTT_connect();  
  
void setup() {  
  Serial.begin(115200);  
  delay(10);  
  Serial.println(F("Adafruit MQTT demo"));  
  // Connect to WiFi access point.  
  Serial.println(); Serial.println();  
  Serial.print("Connecting to ");  
  Serial.println(WLAN_SSID);  
  WiFi.begin(WLAN_SSID, WLAN_PASS);  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }  
  Serial.println();  
  Serial.println("WiFi connected");  
  Serial.println("IP address: "); Serial.println(WiFi.localIP());  
  // Setup MQTT subscription for onoff feed.  
  mqtt.subscribe(&commandsubscribe);  
}  
....
```

# Arduino and AdafruitIO

---

```
....
uint32_t x=0;

void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected). See the MQTT_connect
  // function definition further below.
  MQTT_connect();

  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &commandsubscribe) {
      Serial.print(F("Got: "));
      Serial.println((char *)commandsubscribe.lastread);
    }
  }
  // Now we can publish stuff!
  Serial.print(F("\nSending val "));
  Serial.print(x);
  Serial.print("...");
  if (! commandpublish.publish(x++)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }
}
.....
```



# Arduino and AdafruitIO

---

```
....  
// Function to connect and reconnect as necessary to the MQTT server.  
// Should be called in the loop function and it will take care if connecting.  
void MQTT_connect() {  
  int8_t ret;  
  
  // Stop if already connected.  
  if (mqtt.connected()) {  
    return;  
  }  
  
  Serial.print("Connecting to MQTT... ");  
  
  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  
    retries--;  
    if (retries == 0) {  
      // basically die and wait for WDT to reset me  
      while (1);  
    }  
  }  
  Serial.println("MQTT Connected!");  
}
```

# Arduino and AdafruitIO

---

## References

- <https://learn.adafruit.com/mqtt-adafruit-io-and-you/intro-to-adafruit-mqtt>
- <https://learn.adafruit.com/mqtt-adafruit-io-and-you/overview>

# Javascript and AdafruitIO

---



# Javascript and AdafruitIO

---

A sample html web page with a javascript script to test interaction between AdafruitIO and javascript

```
<html>
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
  </head>
  <body>
    <h1> test </h1>
    <script>

      // Create a client instance
      client = new Paho.MQTT.Client("io.adafruit.com", Number(443), "www");

      // set callback handlers
      client.onConnectionLost = onConnectionLost;
      client.onMessageArrived = onMessageArrived;

      // connect the client
      client.connect({onSuccess:onConnect, userName:"username", password:"AIOKey", useSSL : true, mqttVersion: 4 });

    ....
  </script>
</body>
</html>
```

# Javascript and AdafruitIO

---

```
...  
  
// called when the client connects  
function onConnect() {  
  // Once a connection has been made, make a subscription and send a message.  
  console.log("onConnect");  
  client.subscribe("username/feeds/command");  
  message = new Paho.MQTT.Message("1");  
  message.destinationName = "username/feeds/command";  
  client.send(message);  
}  
  
// called when the client loses its connection  
function onConnectionLost(responseObject) {  
  if (responseObject.errorCode !== 0) {  
    console.log("onConnectionLost:"+responseObject.errorMessage);  
  }  
}  
  
// called when a message arrives  
function onMessageArrived(message) {  
  console.log("onMessageArrived:"+message.payloadString);  
}  
  
</script>  
</body>  
</html>
```

# Javascript and AdafruitIO

---

## References:

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

# Exercises

---

Integrate your sensors from last class and publish their values in an AdafruitIO feed

Create an IFTTT applet to react to different contexts based on your sensor values and send notifications to your smartphone.

Examples:

- 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

# Exercises

---

Subscribe another group sensor values and light a led on your arduino according to certain range conditions

Use exclusively the AdafruitIO platform



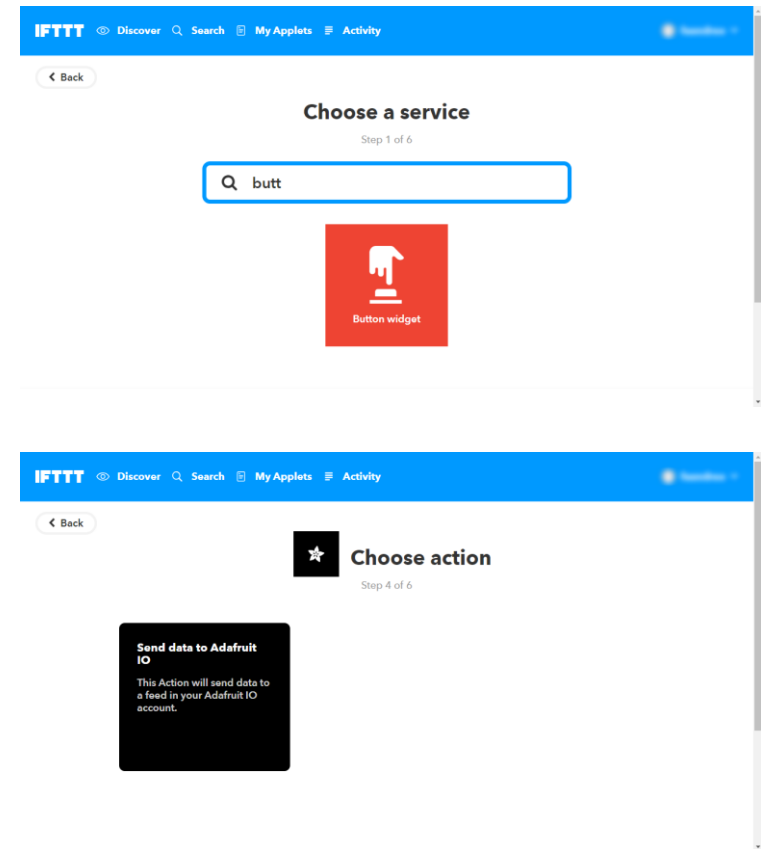
# Exercises

---

Create a button trigger and an AdafruitIO action within an IFTTT applet

When the button is clicked on the IFTTT application data is submitted adafruit feed

Use an AdafruitIO client react to changes on the feed (Ex: light a led, buzz a buzzer, etc)



# Exercises

---

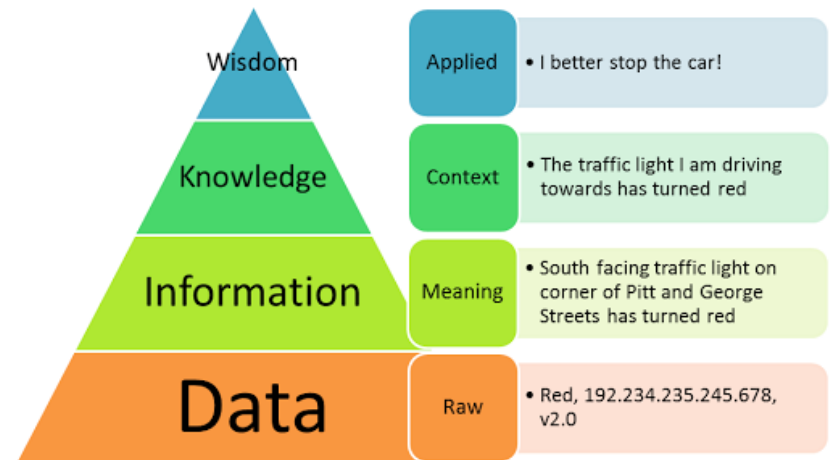
Remember the data pyramid

An efficient actuator perform its tasks after data and information processing

Need to get information about contexts for deploying actions in autonomous system

Actions can be:

- Reactive
- Deliberative
- Hybrid



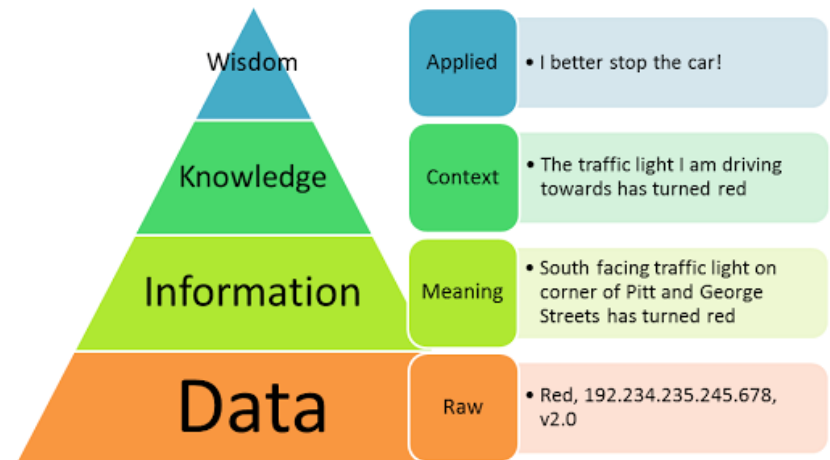
# Exercises

---

Create a Java program that assess sensor values and computes context based on data and information fusion

Examples:

- Thermal sensation (temperature, humidity, wind, solar radiation, clothing, etc )
- Noise sensation (exposure, noise level, etc)
- Safety indicator (exposure, level)
- Smart alert system (location, time, tasks, etc)



# Autonomous Systems

---

INTERNET OF THINGS

Fábio Silva  
fabiosilva@di.uminho.pt