



Using IFTTT with Adafruit IO to Make an IoT Door Detector

Created by Todd Treece



Last updated on 2017-09-12 03:10:35 PM UTC

Guide Contents

Guide Contents	2
Overview	3
Adafruit.io + IFTTT	3
Wiring	4
Low Power Usage	4
Battery tracking	4
Adafruit IO Setup	6
Creating the Feeds	6
Arduino Code	8
Arduino Sketch	8
Upload and Test!	11
IFTTT <-> Adafruit.io	15
IFTTT Connection	15
Creating the IFTTT Recipe	17

Overview

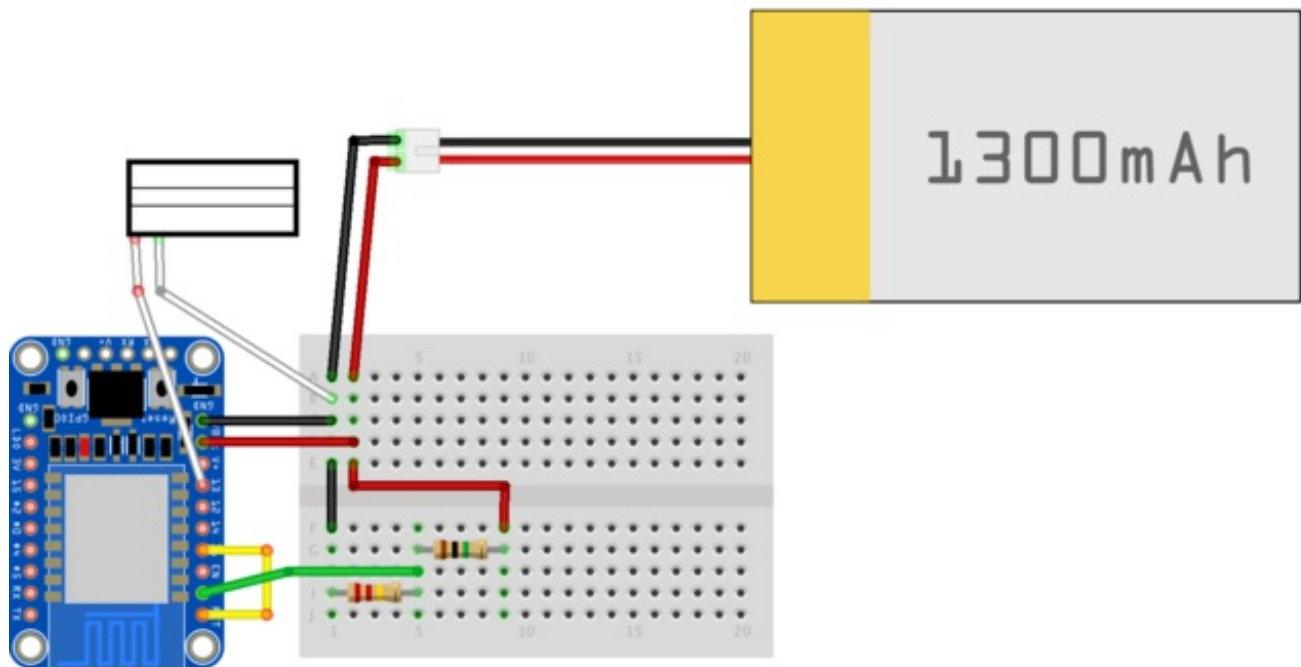


Way cheaper than a guard dog who can use an iPhone, this project will show you how you can use an Adafruit HUZZAH ESP8266 WiFi microcontroller board with a door sensor to email/tweet/text you when your door has opened!

Adafruit.io + IFTTT

This tutorial will show building a full IoT project using Adafruit.IO and a Huzzah board, adding sensors, and then connecting it to [IFTTT \(if-this-then-that\)](http://ifttt.com) (<http://adafru.it/iOe>) an API gateway that can communicate with adafruit.io to give your project tons of connectivity options

Wiring



fritzing

Our schematic is fairly simple, we will have one pin (GPIO #13) monitor the door sensor.

Low Power Usage

We can put the ESP8266 in low power mode and then 'wake up' every few seconds. This is way better for battery life than just keep the ESP8266 awake forever. To do that, tie pin GPIO #16 to the RST pin, that way the auto-wakeup will work.

Battery tracking

We will also keep track of the battery level by creating a high-resistor-divider on VBat to reduce the voltage of the battery from $220K\Omega/1220K\Omega = 1/5.5$ times. This means a max voltage of the battery (4.2V) is 0.75V, well within the range of the ESP8266's 1.0V-max ADC

Note that even with low-power sleeping, this setup draws about 20mA on average!

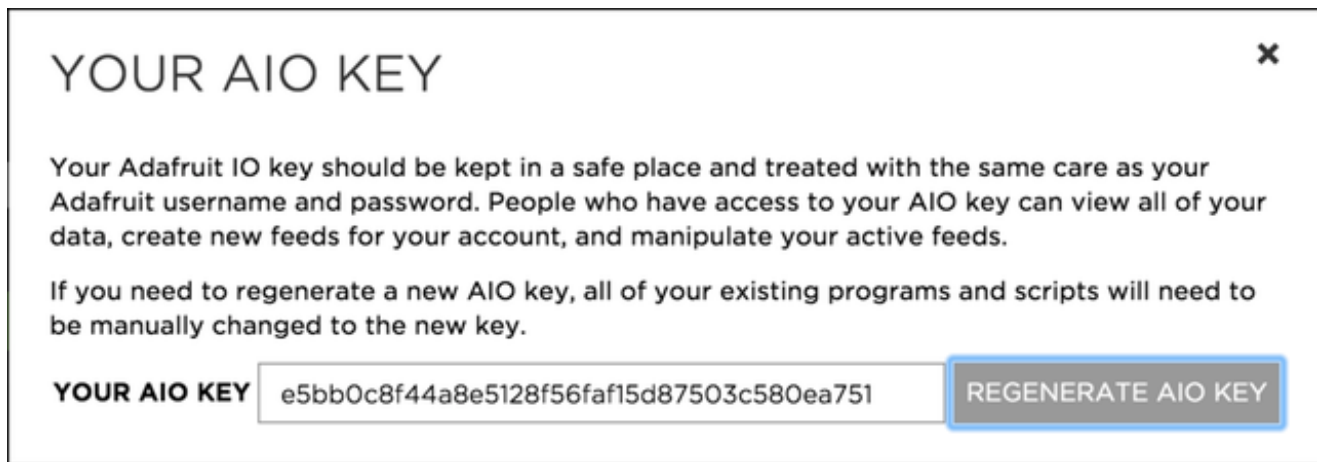
- **Pin 16 to RST** (this lets us use the low power mode)
- **Pin 13 to one side of door sensor**

- **GND** to opposite side of **door sensor**
- **VBat** to the **battery +**
- **GND** to **battery -**
- **GND** to one side of the **220k Ω resistor**
- **VBat** to one side of the **1M Ω resistor**
- **Pin A** to the opposite side of the **1M Ω resistor** and **220k Ω resistor**

Adafruit IO Setup

Before you even upload code to the arduino, you'll need to get your IO account going - that way you can properly test your code and connection!

The first thing you will need to do is to login to your [Adafruit IO account](http://adafru.it/eZ8) (<http://adafru.it/eZ8>) and get your Adafruit IO Key if you haven't already. Click the **AIO KEY** button on the right hand side of the window, to access your key.



A window will pop up with your Adafruit IO key. Keep a copy of this in a safe place. We'll need it later. (Don't use the one in the image above, its not a real key, its just to indicate what the key looks like)

Creating the Feeds

You will now need to create feeds called "*door*" and "*battery*". If you need help getting started with creating feeds on Adafruit IO, check out the [Adafruit IO Feed Basics guide](http://adafru.it/ioA) (<http://adafru.it/ioA>).

Hello, Todd Treece | [Sign Out](#) | [My Feeds](#)

CREATE A NEW FEED

NAME

door

DESCRIPTION

CANCEL

CREATE FEED

706

Photocell

photocell

35

24 days ago

Hello, Todd Treece | [Sign Out](#) | [My Feeds](#)

CREATE A NEW FEED

NAME

battery

DESCRIPTION

CANCEL

CREATE FEED

706

Photocell

photocell

35

24 days ago

Now you can start pushing data into the feeds, by wiring up and uploading the sketch to your Arduino!

Arduino Code

You will need the [Adafruit IO Arduino](http://adafru.it/fpd) (<http://adafru.it/fpd>) library installed to compile the example sketch. The easiest way to install the library is by using the [Arduino IDE v.1.6.4+ Library Manager](http://adafru.it/fCN) (<http://adafru.it/fCN>), and searching for **Adafruit IO Arduino**.

You will also need to have the ESP8266 Arduino board package installed. For more info about installing the ESP8266 package, visit the [HUZZAH ESP8266 setup guide](http://adafru.it/irC) (<http://adafru.it/irC>).

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Use the package index URL above to get the latest version of the ESP8266 package.

You must have your adafruit.io account set up first before you try to connect!

Arduino Sketch

The Arduino sketch for this project is fairly straight forward. If you haven't downloaded or cloned the [Adafruit IO Basics GitHub repo](http://adafru.it/f27) (<http://adafru.it/f27>), you should do that first. We will be using the **trigger** sketch located in the **ESP8266** folder.

[Adafruit IO Basics Sketches](http://adafru.it/f28)
<http://adafru.it/f28>

The first thing you will need to do is to edit the WiFi connection info at the top of the ESP8266 *trigger* sketch. You can use the same connection info you used when you tested your WiFi connection in the_ (<http://adafru.it/itc>)[HUZZAH ESP8266 setup guide](http://adafru.it/itc) (<http://adafru.it/irC>).

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

You will need to replace the Adafruit IO key placeholder in the sketch with the Adafruit IO key that you retrieved in the Adafruit IO Setup section of this guide.

```
#define AIO_KEY      "...your AIO key..."
```

By default, the sketch sends the battery level to Adafruit IO once every 5 minutes, and checks the state of the door once every 3 seconds. You can modify these intervals by changing the **BATTERY_INTERVAL** and **SLEEP_LENGTH** constants. Setting these

constants to lower intervals will result in reduced battery life.

```
// how often to report battery level to adafruit IO (in minutes)
#define BATTERY_INTERVAL 5

// how long to sleep between checking the door state (in seconds)
#define SLEEP_LENGTH 3
```

The bulk of the work for this example happens in the **setup()** function. The sketch restarts after it wakes from sleep, so the main **loop()** never runs.

```
void setup() {
  Serial.begin(115200);
  Serial.println("HUZZAH Trigger Basic");

  EEPROM.begin(512);
  pinMode(DOOR, INPUT_PULLUP);

  // get the current count position from eeprom
  byte battery_count = EEPROM.read(0);

  // we only need this to happen once every X minutes,
  // so we use eeprom to track the count between resets.
  if(battery_count >= ((BATTERY_INTERVAL * 60) / SLEEP_LENGTH)) {
    // reset counter
    battery_count = 0;
    // report battery level to Adafruit IO
    battery_level();
  } else {
    // increment counter
    battery_count++;
  }

  // save the current count
  EEPROM.write(0, battery_count);
  EEPROM.commit();

  // if door isn't open, we don't need to send anything
  if(digitalRead(DOOR) == LOW) {
    Serial.println("Door closed");
    // we don't do anything
  } else {
    // the door is open if we have reached here,
    // so we should send a value to Adafruit IO.
    Serial.println("Door is open!");
    door_open();
  }

  // we are done here. go back to sleep.
  Serial.println("zzzz");
  ESP.deepSleep(SLEEP_LENGTH * 1000000, WAKE_RF_DISABLED);
}
```

```
}
```

The code that sends the state of the door to Adafruit IO can be found in the **door_open()** function.

```
void door_open() {

  // turn on wifi if we aren't connected
  if(WiFi.status() != WL_CONNECTED) {
    wifi_init();
  }

  // grab the door feed
  Adafruit_IO_Feed door = aio.getFeed("door");

  Serial.println("Sending to Adafruit.io");
  // send door open value to AIO
  door.send("1");

}
```

The battery level is also sent to Adafruit IO once every 5 minutes. You can increase this interval using the **BATTERY_INTERVAL** constant, but it will reduce your battery life.

```
void battery_level() {

  // read the battery level from the ESP8266 analog in pin.
  // analog read level is 10 bit 0-1023 (0V-1V).
  // our 1M & 220K voltage divider takes the max
  // lipo value of 4.2V and drops it to 0.758V max.
  // this means our min analog read value should be 580 (3.14V)
  // and the max analog read value should be 774 (4.2V).
  int level = analogRead(A0);

  // convert battery level to percent
  level = map(level, 580, 774, 0, 100);
  Serial.print("Battery level: "); Serial.print(level); Serial.println("%");
  // turn on wifi if we aren't connected
  if(WiFi.status() != WL_CONNECTED)
    wifi_init();

  // grab the battery feed
  Adafruit_IO_Feed battery = aio.getFeed("battery");

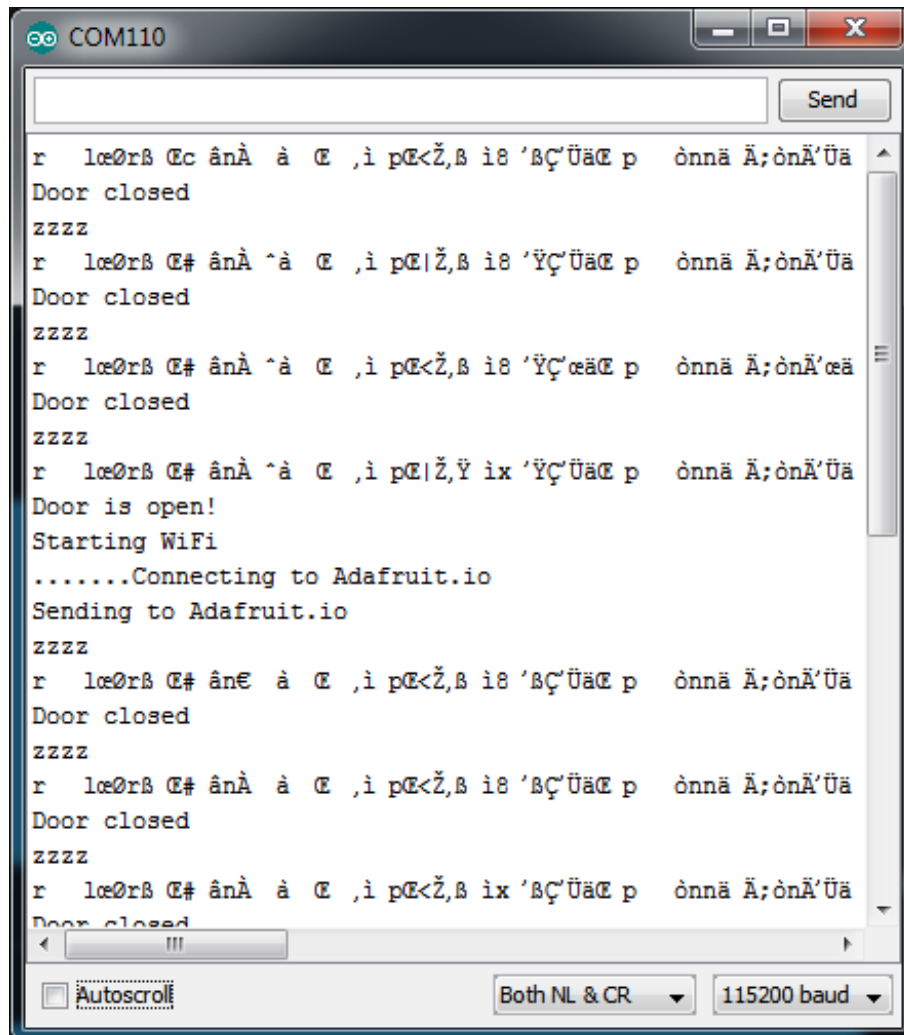
  // send battery level to AIO
  battery.send(level);

}
```

Upload and Test!

OK now that you have everything wired up, and your adafruit.io account set up, upload the above sketch to your HUZZAH ESP8266 board via the FTDI cable or other serial connection.

After uploading, open up the serial port, you should see a long stream of text that looks like this:



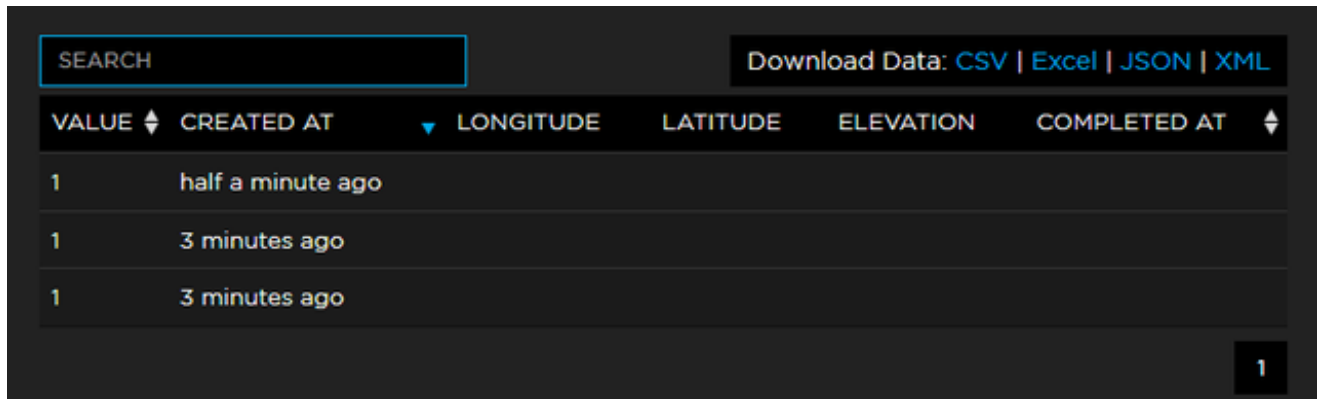
```
COM110
r 1æ0r8 0c ânÀ à 0 ,i p0<Ž,8 i8 'BÇ'Üä0 p ònnä Ä;ònÄ'Üä
Door closed
zzzz
r 1æ0r8 0# ânÀ ^à 0 ,i p0|Ž,8 i8 'YÇ'Üä0 p ònnä Ä;ònÄ'Üä
Door closed
zzzz
r 1æ0r8 0# ânÀ ^à 0 ,i p0<Ž,8 i8 'YÇ'ä0 p ònnä Ä;ònÄ'ä
Door closed
zzzz
r 1æ0r8 0# ânÀ ^à 0 ,i p0|Ž,Y ix 'YÇ'Üä0 p ònnä Ä;ònÄ'Üä
Door is open!
Starting WiFi
.....Connecting to Adafruit.io
Sending to Adafruit.io
zzzz
r 1æ0r8 0# ân0 à 0 ,i p0<Ž,8 i8 'BÇ'Üä0 p ònnä Ä;ònÄ'Üä
Door closed
zzzz
r 1æ0r8 0# ânÀ à 0 ,i p0<Ž,8 i8 'BÇ'Üä0 p ònnä Ä;ònÄ'Üä
Door closed
zzzz
r 1æ0r8 0# ânÀ à 0 ,i p0<Ž,8 ix 'BÇ'Üä0 p ònnä Ä;ònÄ'Üä
Door closed
```

What you're seeing is the HUZZAH reading the sensor (Door closed and Door is open!) and then going to sleep (the zzz text indicates its about to go into sleep mode. When the ESP8266 goes to sleep, it resets the board, and so the weird text after the 'zzz' is the reset debug string. Its normal but some terminal programs print it out a little differently.

This program only pushes data to the feed when the door is open, since that's a rare event. So when the door is closed, nothing 'happens'. When you remove the magnet half from the

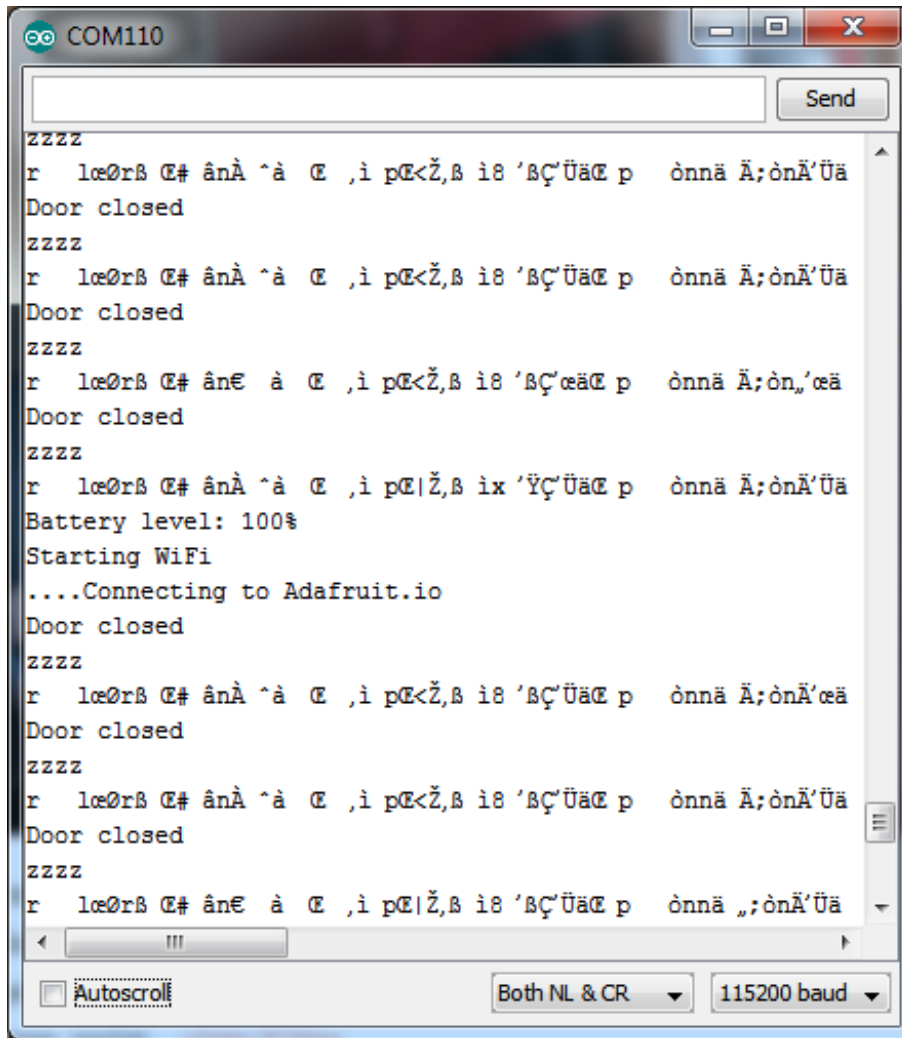
sensor half, you'll get that text that says the door is open, and it will connect to WiFi and update your feed

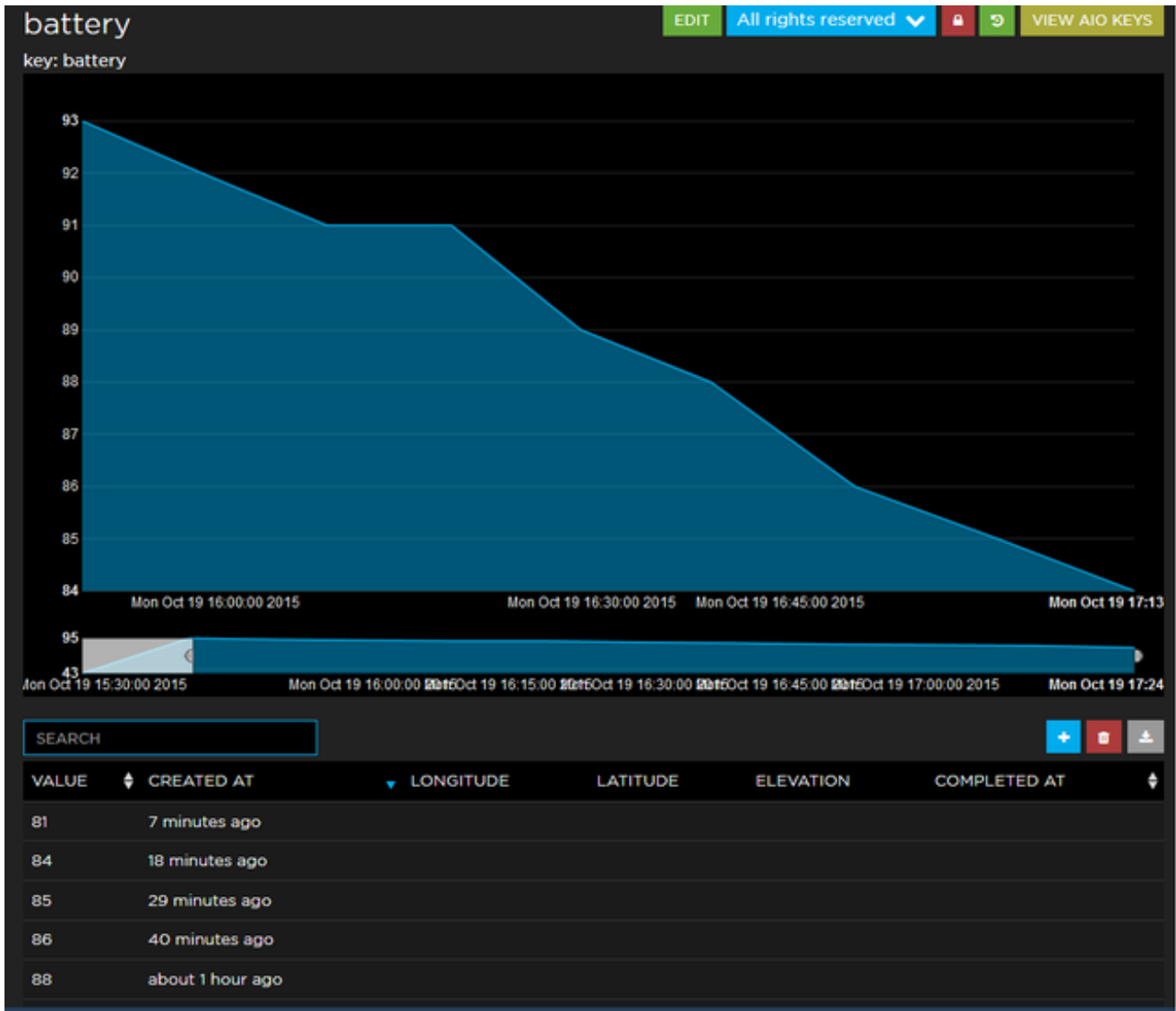
Log into your **adafruit.io** account and look at the **door** feed to see the messages as they are logged!



VALUE	CREATED AT	LONGITUDE	LATITUDE	ELEVATION	COMPLETED AT
1	half a minute ago				
1	3 minutes ago				
1	3 minutes ago				

After 5 minutes, you'll also see the board measure the battery voltage and upload that to the separate **battery** feed.

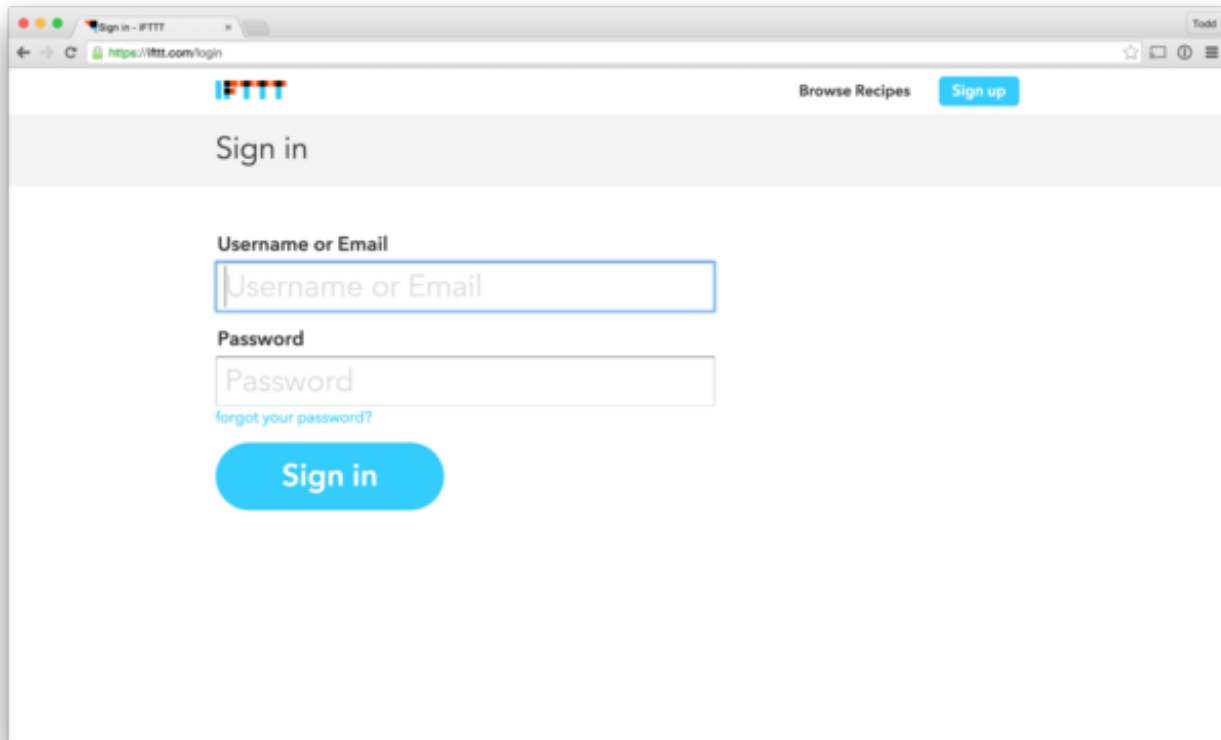




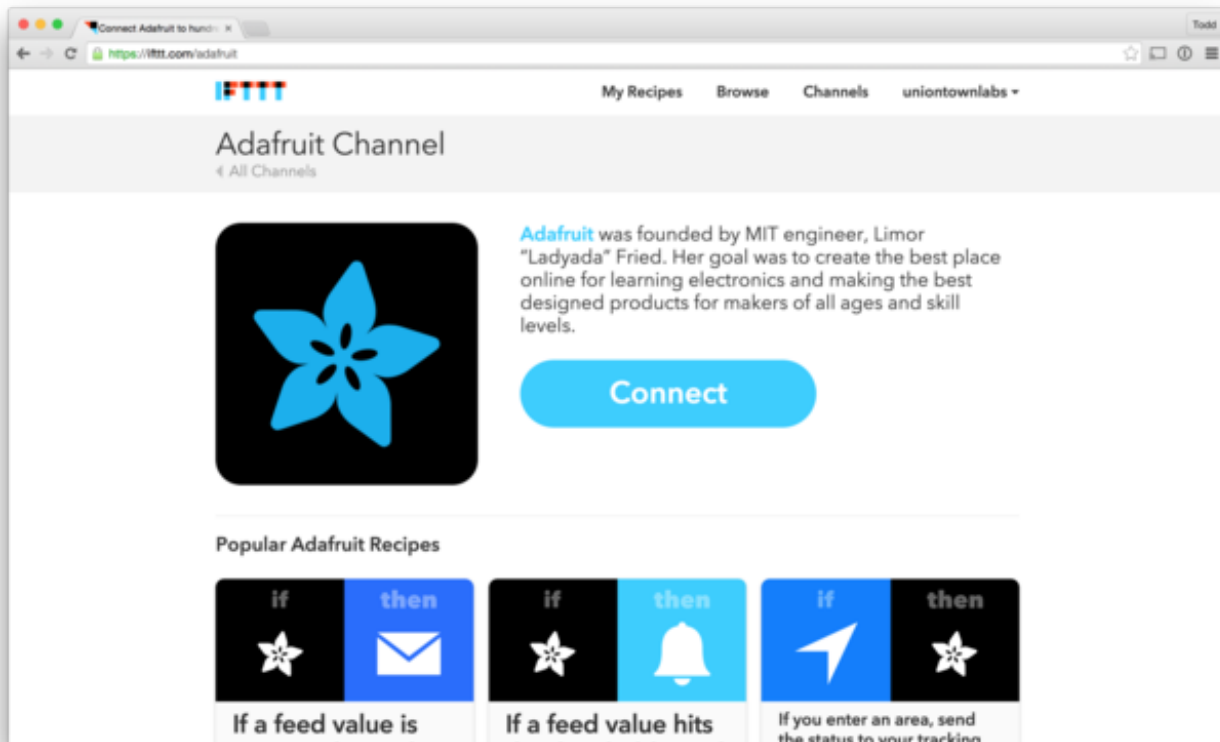
IFTTT <-> Adafruit.io

IFTTT Connection

OK don't forget you will have to visit ifttt.com (<http://adafru.it/fYY>) and sign up for an account.



Once you have signed in, visit ifttt.com/adafruit (<http://adafru.it/fYZ>) and click the **Connect** button to connect your IFTTT account to Adafruit IO.



You will be redirected to your Adafruit account page, and will be asked to authorize IFTTT. Click the **AUTHORIZE** button to grant access.

Your Account

[Profile](#)[Addresses](#)[Order History](#)[My Wishlists](#)[Manage Gift Certificates](#)[Product Notifications](#)[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

[CONTACT](#)[SUPPORT](#)

"Th

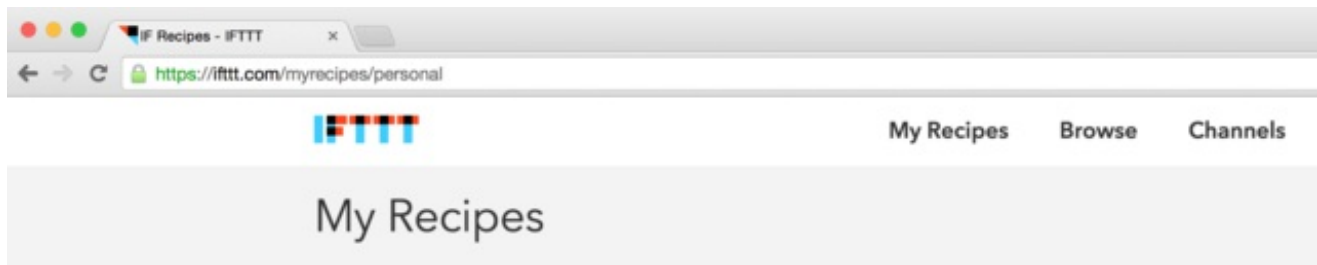
You will then be redirected back to IFTTT, and should see a **Channel connected!** message at the top of the page.

[My Recipes](#)[Browse](#)[Channels](#)[unio](#)**Channel connected!**

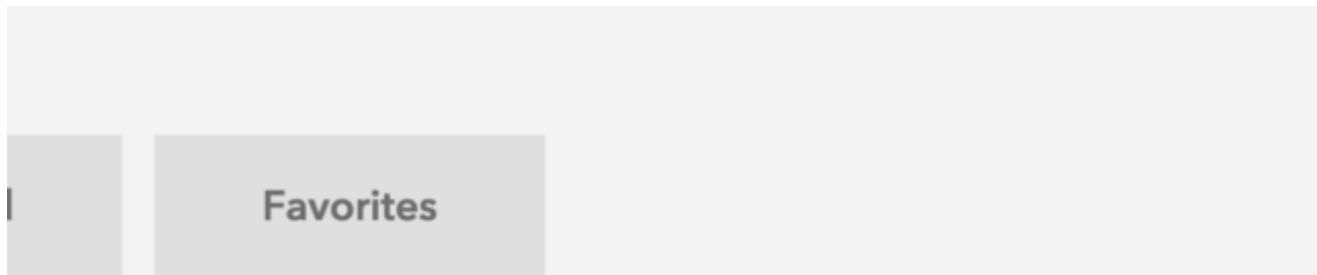
Adafruit was founded by MIT engineer, Limor "Ladyada" Fried. Her goal was to create the best online for learning electronics and making the designed products for makers of all ages and skill levels.

Creating the IFTTT Recipe

Next, navigate to your personal [IFTTT Recipes](http://adafru.it/fZ0) (<http://adafru.it/fZ0>) page to start the recipe creation process.



Click the **Create a Recipe** button.



ground.

Create a Recipe

Recipes yet! Check out one of
find one you'll love.

Click the blue **this** block of text to get started.

Create a Recipe

ifthisthen that

Click this to get started.

Click on the **Adafruit** channel.

Choose Trigger Channel

Showing Channels that provide at least one Trigger. [View all Channels](#)



500px



Adafruit



Amazon Alexa

Click on **Monitor a feed in Adafruit IO**.



Choose a Trigger

step 2 of 7

Any new data

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

Select the **battery** feed from the dropdown, and make the relationship less than the value of **10**. Click the **Create Trigger** button when finished.



Complete Trigger Fields

step 3 of 7

Monitor a feed on Adafruit IO

Feed

battery 

Relationship

less than 

Value

10

The value to compare against.

Create Trigger

Then, click the blue **that** block of text to select what happens when the battery level is low.




Next, choose an action to happen when the trigger activates. In this example, we will be sending an email.


Choose Action Channel step 4 of 7 back ^

Showing Channels that provide at least one Action. [View all Channels](#)


email



Email



Email Digest



Gmail

Select the **send me an email** option.



Choose an Action

step 5 of 7

Send me an email

This Action will send you an HTML based email. Images and links are supported.

Enter the **subject** and **body** messages that you would like sent to you when the trigger happens, and then click **Create Action**.



Complete Action Fields

step 6 of 7

Send me an email



Subject

FeedName low



Body

The FeedValue is Operator TriggerValue at
CreatedAt !

Create Action

Edit the recipe title, and click **Create Recipe** to complete the process.

if  then

Data on battery feed is less than
10

Recipe Title

If data on battery feed is less than 10, then send me
an email at me@example.com

59

use '#' to add tags

Create Recipe

Next. Repeat the same process for the **door** feed, but instead of monitoring for specific values, choose to monitor for **Any new data**.



Complete Trigger Fields

step 3 of 7

Any new data



Feed name

door



Create Trigger

Your recipes are now finished and waiting for door and battery changes.

IF Recipes run automatically in the background.

Create a Recipe



If any new data on door feed, then send me an email at me@example.com



created less than a minute ago
never run



If data on battery feed is less than 10, then send me an email at me@example.com



created 2 minutes ago
never run

You will then be setup to receive IFTTT emails whenever someone opens the door!

Back Door Open



Inbox x



IFTTT Action <action@ifttt.com>

to me ▾

Someone is breaking into your house.