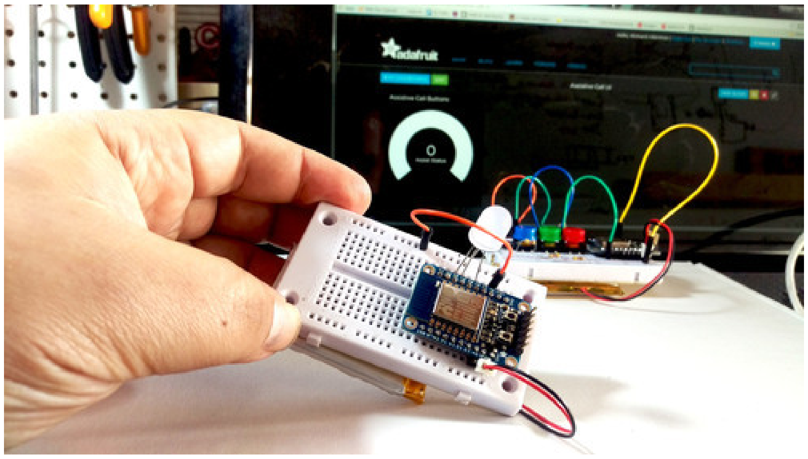
adafruit learning system

Remote Control with the Huzzah + Adafruit.io

Created by Richard Albritton



Last updated on 2018-08-22 03:50:18 PM UTC

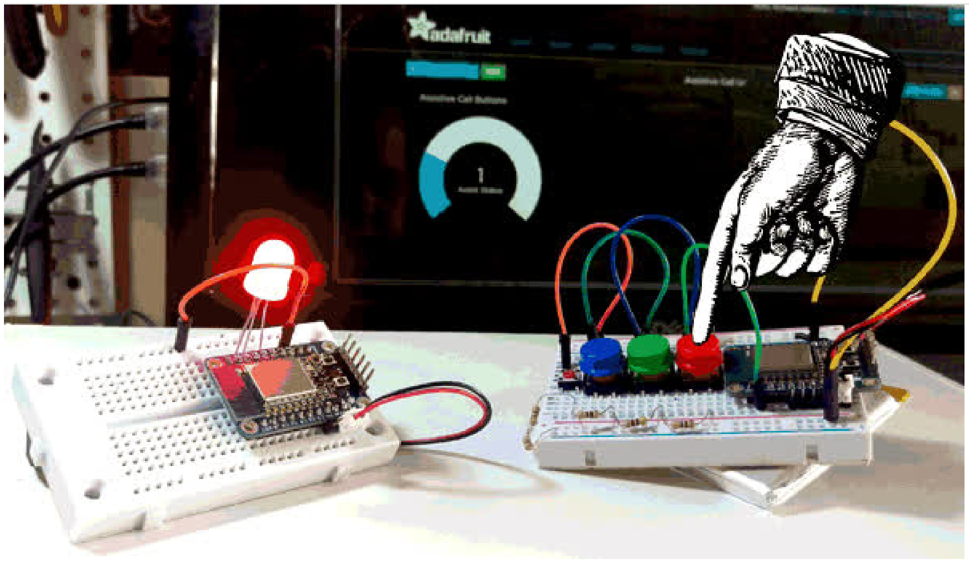
Guide Contents

Guide Contents 2
Overview 3
Adafruit IO Setup 4
- Creating the Buttons Feed 4
- Adding the Gauge Block 4

Arduino Setup 6
- Arduino Dependencies 6
  - Adafruit IO Setup 6
  - Wireless Setup 7

Button Board 8
Buttons Wiring 9
Buttons Code 10
LED 13
LED Wiring 14
LED Code 15
Test it out 18

Overview



This guide will show you how to send button press data to Adafruit IO from the HUZZAH ESP8266, then use that data to change the color of an LED connected to a second HUZZAH ESP8266.

If you haven't worked your way through the Adafruit IO feed and dashboard basics guides, you should do that before continuing with this guide so you have a basic understanding of Adafruit IO.

- Adafruit IO Basics: Feeds (https://adafru.it/ioA)
- Adafruit IO Basics: Dashboards (https://adafru.it/f5m)

You should also go through the setup guide associated with the HUZZAH ESP8266 for use with the Arduino IDE.

- Adafruit HUZZAH ESP8266 Setup Guide (https://adafru.it/inB)

If you have completed all of the prerequisites listed above, you are now ready to move on to the Adafruit IO setup steps.

Let's get started!

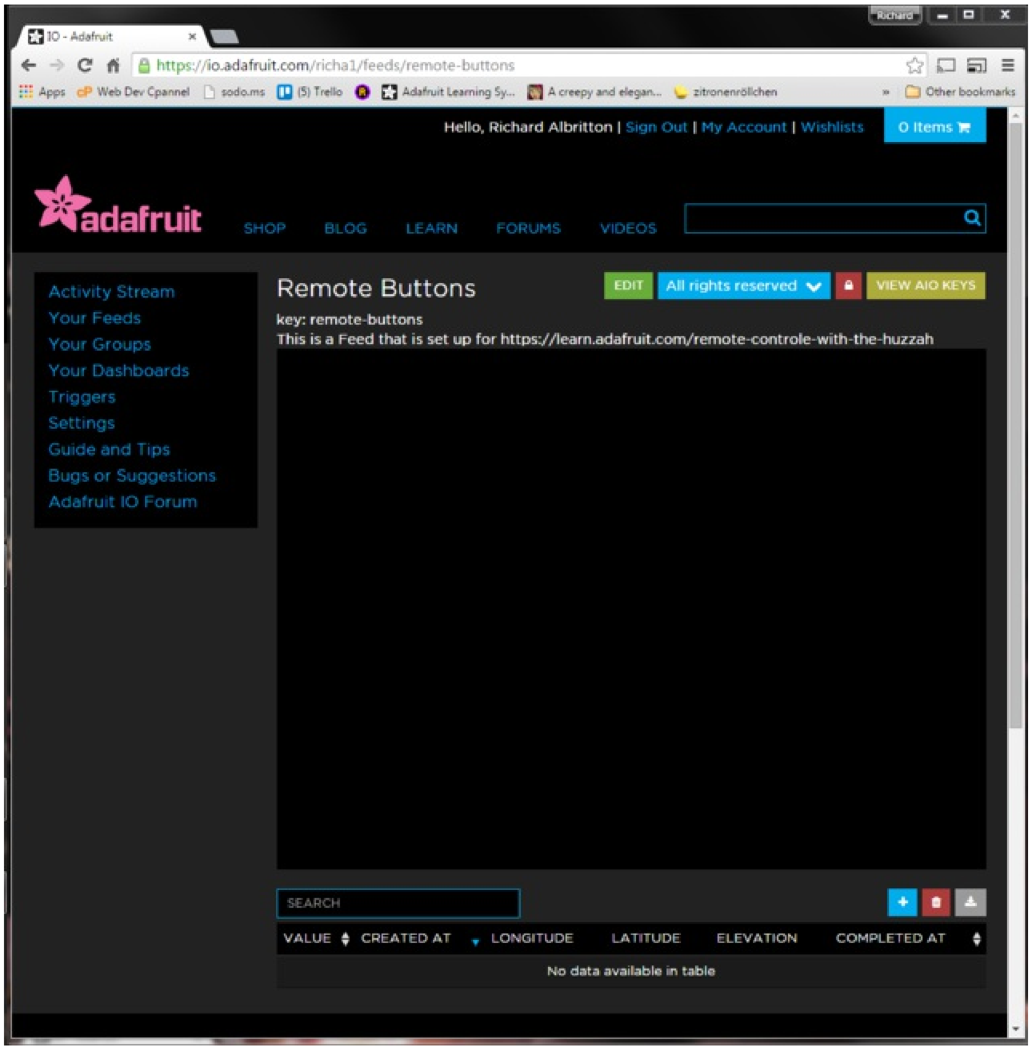
Adafruit IO Setup

The first thing you will need to do is to login to your Adafruit IO account (https://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 retrieve your key.

A window will pop up with your Adafruit IO. Keep a copy of this in a safe place. We'll need it later.

Creating the Buttons Feed

Next, you will need to create a feed called *"Remote Buttons"*. If you need help getting started with creating feeds on Adafruit IO, check out the Adafruit IO Feed Basics guide (https://adafru.it/ioA).



Remote Buttons

key: remote-buttons

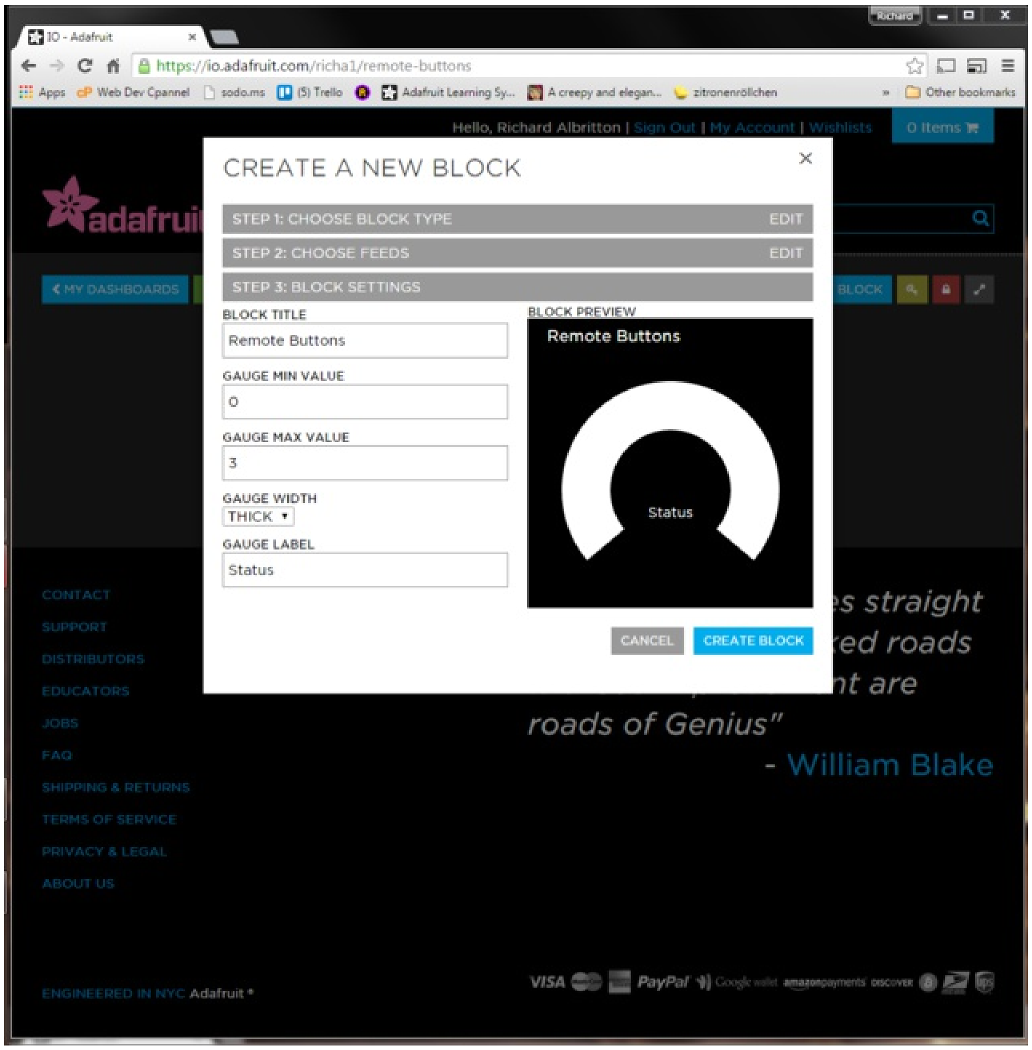
This is a Feed that is set up for https://learn.adafruit.com/remote-controle-with-the-huzzah

VALUE	CREATED AT	LONGITUDE	LATITUDE	ELEVATION	COMPLETED AT
No data available in table					

Adding the Gauge Block

Next, add a new **Gauge** block to a new or existing dashboard. Name the block whatever you would like, and give it a **max value** of **3**. Make sure you have selected the **Remote Buttons** feed as the data source for the gauge.

If you need help getting started with Dashboards on Adafruit IO, check out the Adafruit IO Dashboard Basics guide (https://adafru.it/f5m).



CREATE A NEW BLOCK

STEP 1: CHOOSE BLOCK TYPE

STEP 2: CHOOSE FEEDS

STEP 3: BLOCK SETTINGS

BLOCK TITLE: Remote Buttons

GAUGE MIN VALUE: 0

GAUGE MAX VALUE: 3

GAUGE WIDTH: THICK

GAUGE LABEL: Status

When you are finished editing the form, click **Create Block** to add the new block to the dashboard.

Next, we will look at how to connect an Arduino via WiFi to Adafruit IO.

Arduino Setup

Arduino Dependencies

The code in this guide are based off of the ESP8266 examples created by Tony DiCola and Todd Treece. You will need the following Arduino library installed to compile the sketch:

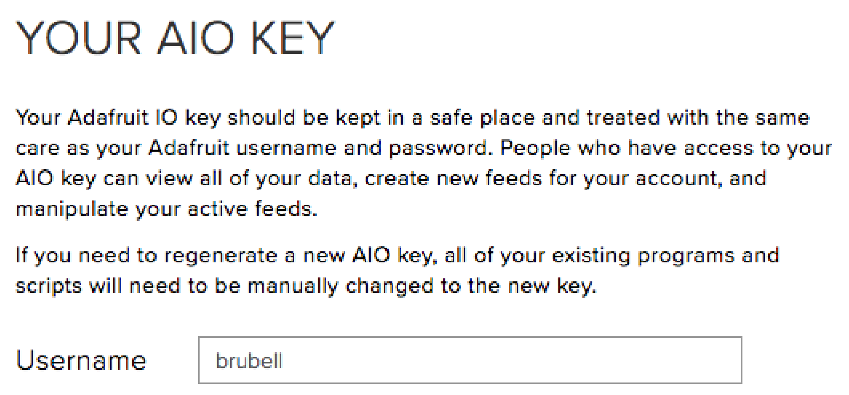
- Adafruit IO Arduino Library (https://adafru.it/fpd)

The easiest way to install it is by using the Arduino IDE v.1.6.4+ Library Manager (https://adafru.it/fCN). You will also need to have the ESP8266 board manager package installed. For more info about installing the ESP8266 package, visit the HUZZAH ESP8266 setup guide (https://adafru.it/irC).

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

Adafruit IO Setup

To find your IO_USERNAME , navigate to your profile on Adafruit IO (https://adafru.it/BmD) and click **View AIO Key**. Copy the **Username** field (ctrl+c or command+c)



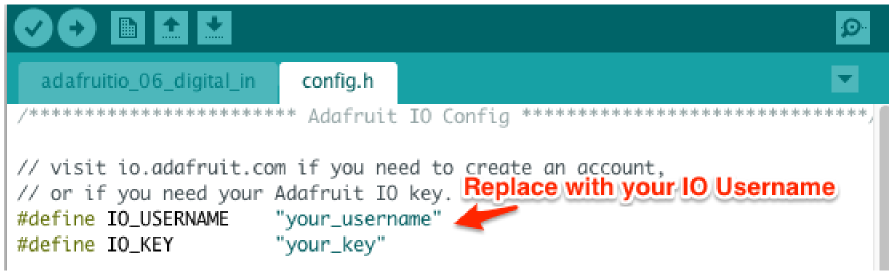
YOUR AIO KEY

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.

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.

Username: brubell

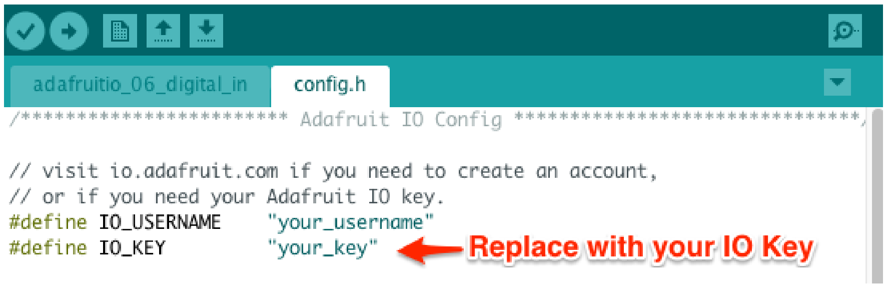
Then, in the config.h tab, replace the "your_username" text with your the username from your profile:



```
/*********************** Adafruit IO Config *******************************/

// visit io.adafruit.com if you need to create an account,
// or if you need your Adafruit IO key.
#define IO_USERNAME    "your_username"
#define IO_KEY         "your_key"
```

To find your IO_Key , navigate to your profile, click **View AIO Key**, and copy the **ACTIVE KEY** field to your clipboard (ctrl+c or command+c).

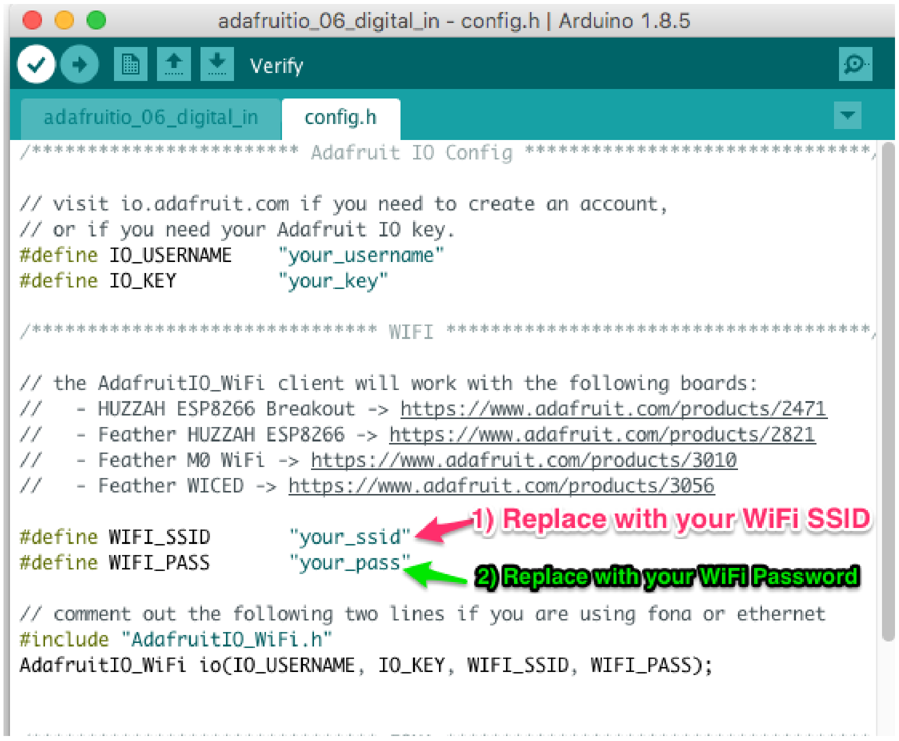
In config.h , replace the IO_KEY with the IO Key copied to your clipboard.



```
/************************ Adafruit IO Config *******************************/

// visit io.adafruit.com if you need to create an account,
// or if you need your Adafruit IO key.
#define IO_USERNAME    "your_username"
#define IO_KEY         "your_key"
```

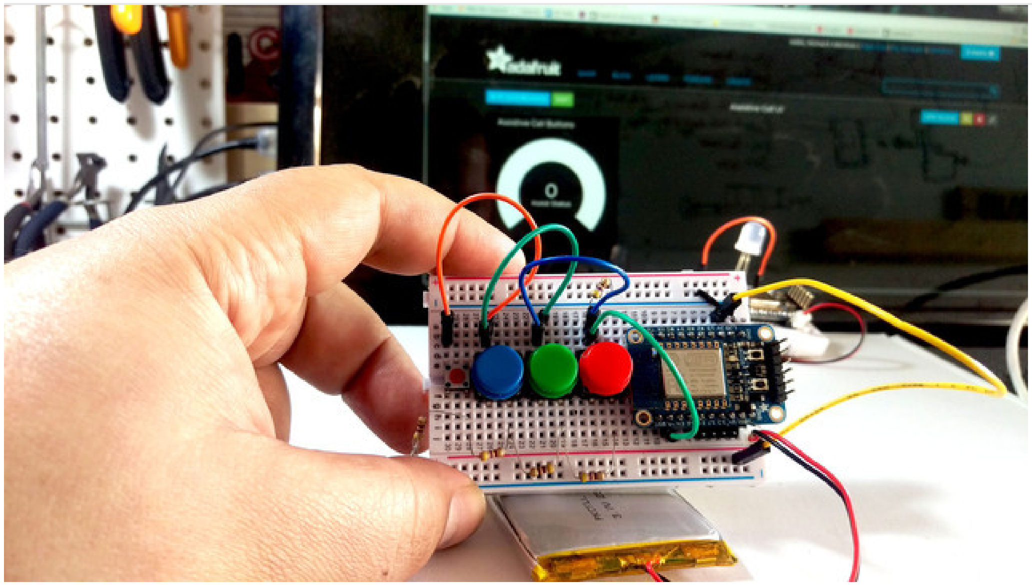
Wireless Setup

In the config.h tab, replace "your_ssid" with your WiFi's SSID and "your_pass" with your WiFi's password:


adafruitio_06_digital_in - config.h | Arduino 1.8.5
Verify
adafruitio_06_digital_in
config.h
1) Replace with your WiFi SSID
2) Replace with your WiFi Password

```
/************************ Adafruit IO Config *******************************/

// visit io.adafruit.com if you need to create an account,
// or if you need your Adafruit IO key.
#define IO_USERNAME    "your_username"
#define IO_KEY         "your_key"

/******************************* WIFI **************************************/

// the AdafruitIO_WiFi client will work with the following boards:
//   - HUZZAH ESP8266 Breakout -> https://www.adafruit.com/products/2471
//   - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821
//   - Feather M0 WiFi -> https://www.adafruit.com/products/3010
//   - Feather WICED -> https://www.adafruit.com/products/3056

#define WIFI_SSID       "your_ssid"
#define WIFI_PASS       "your_pass"

// comment out the following two lines if you are using fona or ethernet
#include "AdafruitIO_WiFi.h"
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```

Button Board



First we will create a simple input controller that will let Adafruit IO know what button was pressed. For this project we will be adding buttons to represent each of the possible states to be passed to our output device.

- 0 will represent "OFF"
- 1 for Red
- 2 for Green
- 3 for Blue

There are a few things we will need to get started:

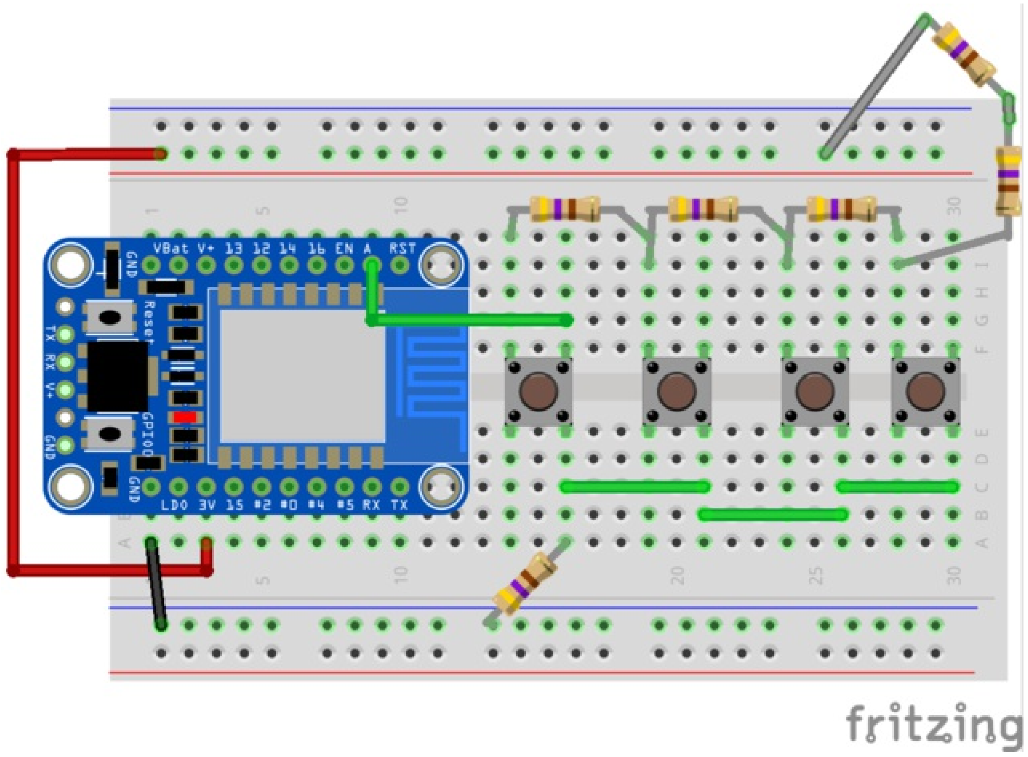
- 1 x HUZZAH ESP8266 (https://adafru.it/iDh)
- 3 x colored tactile push buttons (Red, Green, and Blue) (https://adafru.it/iDi)
- Six 470 Ohms resistors
- Jumper wires (https://adafru.it/iDj)
- Breadboard (https://adafru.it/erc)

Now let's get started with the wiring.

Buttons Wiring

There are basically two ways to add simple button input for this project. We could just add a button to each digital pin, but that would limit the number of buttons we could use. This time around we will use resistors along side of our buttons and read the analog value based on what button was pressed. This works by adding one resistor to each button, but also setting the resistors up in series. When nothing is pressed, all of the resistors are used and we get a low number value when the analog pin is read. When we press a button, it shorts out the resistor string and we get another value when the analog pin is read. The higher the button is on the chain, the lower the analog value is. Ultimately, this means that we can add as many buttons as we want and only use up one PIN to do it.

We will now connect all of the components to match what is shown below. There should be just enough room to fit 3 of the colored push buttons and one of the 6MM buttons side by side.



1. Connect one of the **GND** pins to the **-** side of the breadboard.
2. The **3V** pin should connect with the + side of the breadboard.
3. Connect one side of each button to it neighbouring button.
4. The last button will connect to the **A** pin on the HUZZAH.
5. Now add one resister connecting **GND** to the **A** pin.
6. Add the remaining resisters to the other side of each button in series with the last two connecting with the **3V** pin.

This can be powered through the FTDI cable, but you may want to add a battery or other power supply by connecting it to the **GND** and **VBat** pins.

Now it's time to code.

Buttons Code

This guide has been updated to use the Adafruit IO Arduino Library

The previous version of this guide used the Adafruit MQTT Arduino Library, but we suggest using the Adafruit IO Arduino Library. If you'd like to use this library, we suggest following the MQTT, Adafruit IO & You! Learn Guide (https://adafru.it/Bor)

The Arduino sketch for this project is fairly straight forward. Copy the following code into a new Arduino sketch. You'll need to modify your `config.h` file to reflect your Wireless Network and Adafruit IO configuration.

If you do not know how to do this, follow the steps on the the Arduino Setup (https://adafru.it/Bos) page.

```
// Remote Control with the Huzzah + Adafruit IO
//
// Button Board
//
// Adafruit invests time and resources providing this open source code.
// Please support Adafruit and open source hardware by purchasing
// products from Adafruit!
//
// Written by Richard Albritton, based on original code by Tony DiCola for Adafruit Industries
// Licensed under the MIT license.
//
// All text above must be included in any redistribution.

/************************** Configuration ***********************************/

// edit the config.h tab and enter your Adafruit IO credentials
// and any additional configuration needed for WiFi, cellular,
// or ethernet clients.
#include "config.h"

/************************ Example Starts Here *******************************/
#include <ESP8266WiFi.h>

// Analog Pin on ESP8266
#define Buttons A0

// remote-buttons state
int ButtonRead = 0;
int current = 0;
int last = -1;

// set up the 'remote-buttons' feed
AdafruitIO_Feed *RemoteButtons = io.feed("remote-buttons");

void setup () {

  // start the serial connection
  Serial.begin(115200);

  // wait for serial monitor to open
  while(! Serial);

  // connect to io.adafruit.com
  Serial.print("Connecting to Adafruit IO");
  io.connect();

  // wait for a connection
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  // we are connected
  Serial.println();
  Serial.println(io.statusText());

}

void loop() {

  // io.run(); is required for all sketches.
  // it should always be present at the top of your loop
  // function. it keeps the client connected to
  // io.adafruit.com, and processes any incoming data.
  io.run();

  ButtonRead = analogRead(Buttons);
  delay(1);
  // grab the current state of the remote-buttons
  if (ButtonRead > 500 && ButtonRead < 600) {
    current = 1;
  }
  if (ButtonRead > 600 && ButtonRead < 750) {
    current = 2;
  }
  if (ButtonRead > 750 && ButtonRead < 900) {
    current = 3;
  }
  if (ButtonRead > 900) {
    current = 0;
  }

  // ret if value hasnt changed
  if(current == last)
    return;

  int32_t value = current;

  // let's publish stuff
  Serial.print("Sending RemoteButtons Value: ");
  Serial.print(value);
  Serial.print("...");

  RemoteButtons->save(value);
  delay(3000);

}
```

When you are finished reviewing the sketch and have finished making the necessary config changes, upload the sketch to your HUZZAH using the Arduino IDE. You should also open up your Adafruit IO dashboard so you can monitor the button gauge.

If everything goes as expected, you will see the gauge update on Adafruit IO. You should make sure to open the Arduino IDE's serial monitor if you are having issues with the sketch. It will provide valuable debugging info.

To add more buttons, just copy one of the conditional (**if**) statements and palce it in with the others. Each of the conditional statments reprisents a button that registers a press if the analog value is within a certain threshhold. You will need to set a minimum and maximum threshhold for all but the last button. The last button just needs to be grater than the maximum value of the button before it.

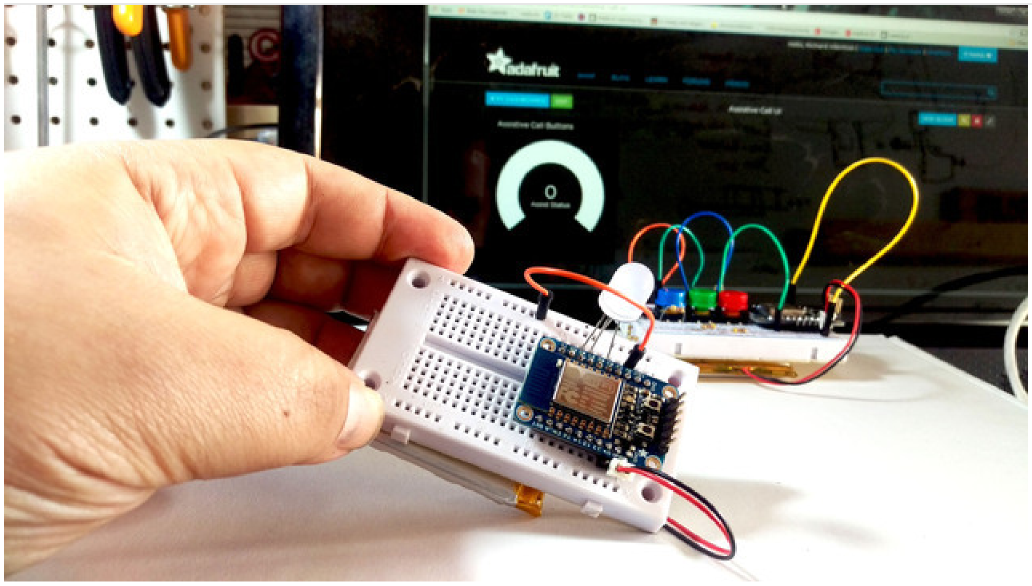
```
// grab the current state of the remote-buttons
   if (ButtonRead > 500 && ButtonRead < 600) {
     current = 1;
   }
   if (ButtonRead > 600 && ButtonRead < 750) {
     current = 2;
   }
   if (ButtonRead > 750 && ButtonRead < 900) {
     current = 3;
   }
   if (ButtonRead > 900) {
     current = 0;
   }
```

This sketch is set up to output the analog value of whatever button is pressed. You could figure out was the value of each button is with some math, but it is easier to just look at what the reading is using the serial monitor. Give each button a wide birth to accommodate for any fluctuations.

Don't forget to add a new number to be set as the **current** value. Also be sure to set the Gauge so that it can accommodate the extra numbers.

Next, we will set up another HUZZAH to output the feed data to an RGB LED.

LED



Now it is time to connect an RGB LED to Adafruit IO so that it can display the color of the button that was pressed.

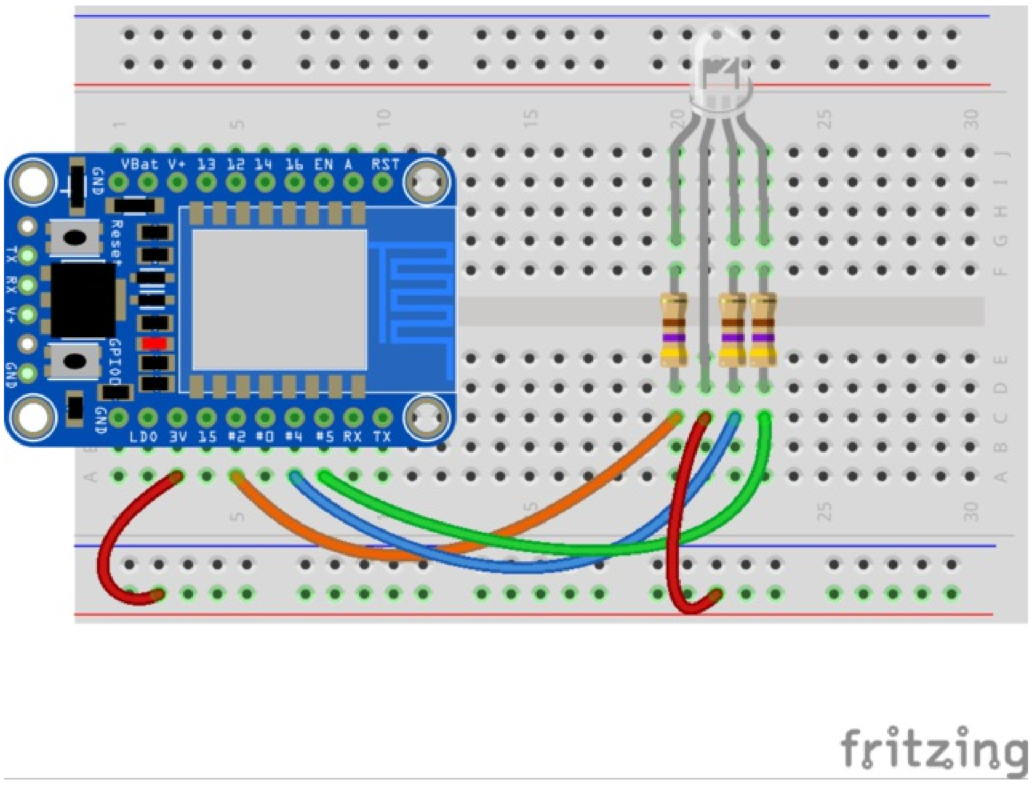
There are a few things we will need to get started:

- 1 x HUZZAH ESP8266 (https://adafru.it/iDh)
- 1 x RGB LED (https://adafru.it/iDk)
- Jumper wire (https://adafru.it/iDj)
- Breadboard (https://adafru.it/erc)
- 3 x 470 ohm resistors (not shown)

Now let's get started with the wiring.

LED Wiring

We will now connect all of the components to match what is shown below.



This design uses a Common Anode LRD. You can use a Common Cathode LED, but you will need to change the Arduino code accordingly.

1. Connect the **3V** pin to the + side of the breadboard.
2. The longest pin of the LED needs to connect with the **3V** pin.
3. Pin **#2** will connect to the **Red** pin on the LED.
4. The **Blue** pin sould connect with pin **#4**.
5. Connect pin **#5** to the **Green** pin.

This can be powered through the FTDI cable, but you may want to add a battery or other power supply by connecting it to the **GND** and **VBat** pins.

Now it's time to code.

LED Code

This guide has been updated to use the Adafruit IO Arduino Library

The previous version of this guide used the Adafruit MQTT Arduino Library, but we suggest using the Adafruit IO Arduino Library. If you'd like to use this library, we suggest following the MQTT, Adafruit IO & You! (https://adafru.it/Bor) Learn Guide

The Arduino sketch for the LEDs is also fairly straight forward. Copy the following code into a new Arduino sketch. You'll need to modify your `config.h` file to reflect your Wireless Network and Adafruit IO configuration.

If you do not know how to do this, follow the steps on the the Arduino Setup (https://adafru.it/Bos) page.

```
// Remote Control with the Huzzah + Adafruit IO
//
// LED Board
//
// Adafruit invests time and resources providing this open source code.
// Please support Adafruit and open source hardware by purchasing
// products from Adafruit!
//
// Written by Richard Albritton, based on original code by Tony DiCola for Adafruit Industries
// Licensed under the MIT license.
//
// All text above must be included in any redistribution.

/************************** Configuration ***********************************/

// edit the config.h tab and enter your Adafruit IO credentials
// and any additional configuration needed for WiFi, cellular,
// or ethernet clients.
#include "config.h"

/************************ Example Starts Here *******************************/

#include <ESP8266WiFi.h>

// RGB LED Pins
#define Blue  5
#define Green 4
#define Red   2

// set up the 'digital' feed
AdafruitIO_Feed *AssistiveCallButtons = io.feed("assistive-call-buttons");

void setup() {

  // set power switch tail pin as an output
  pinMode(Blue, OUTPUT);
  pinMode(Green, OUTPUT);
  pinMode(Red, OUTPUT);

  digitalWrite(Red, HIGH);
  digitalWrite(Blue, HIGH);
  digitalWrite(Green, HIGH);
  // set up serial monitor
  Serial.begin(115200);

  // wait for serial monitor to open
  while(! Serial);

  // connect to io.adafruit.com
  Serial.print("Connecting to Adafruit IO");
  io.connect();

  // set up a message handler for the 'digital' feed.
  // the handleMessage function (defined below)
  // will be called whenever a message is
  // received from adafruit io.
  AssistiveCallButtons -> onMessage(handleMessage);

  // wait for a connection
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  // we are connected
  Serial.println();
  Serial.println(io.statusText());
  // recv. the assistive-call-buttons feed
  AssistiveCallButtons->get();

}

void loop() {

  // io.run(); is required for all sketches.
  // it should always be present at the top of your loop
  // function. it keeps the client connected to
  // io.adafruit.com, and processes any incoming data.
  io.run();

}

// this function is called whenever an 'digital' feed message
// is received from Adafruit IO. it was attached to
// the 'digital' feed in the setup() function above.
void handleMessage(AdafruitIO_Data *data) {

  Serial.print("received <- AssistiveCallButtons");
  switch(data->toInt()) {
    case 0: // your hand is on the sensor
      digitalWrite(Red, HIGH);
      digitalWrite(Blue, HIGH);
      digitalWrite(Green, HIGH);
      break;
    case 1: // your hand is close to the sensor
      digitalWrite(Red, LOW);
      digitalWrite(Red, HIGH);
      digitalWrite(Red, HIGH);
      break;
    case 2: // your hand is a few inches from the sensor
      digitalWrite(Red, HIGH);
      digitalWrite(Red, HIGH);
      digitalWrite(Red, LOW);
      break;
    case 3: // your hand is nowhere near the sensor
      digitalWrite(Red, HIGH);
      digitalWrite(Red, LOW);
      digitalWrite(Red, HIGH);
      break;
  }
  // delay in-between reads for stability
  delay(1);
}
```

You will then need to check that the name of your feed matches the feed defined in the sketch:

```
AdafruitIO_Feed *AssistiveCallButtons = io.feed("assistive-call-buttons");
```

When you are finished reviewing the sketch and have finished making the necessary config changes, upload the sketch to your HUZZAH using the Arduino IDE. You should also open up your Adafruit IO dashboard so you can monitor the Remote Buttons gauge.

Let's test this thing out now.

Test it out

Now that we have things constructed, power on the Buttons controller as well as the LED bourds.

When ever you press one of the buttons on the first Huzzah board, that should change the color of the LED on the second board. So long as they are both connected to the Internet, they can be used together. The status of the buttons that were pressed can also be seen on your Adafruit.IO dashboard.