



# Automatic Twitch On-Air Sign

Created by lady ada



<https://learn.adafruit.com/automatic-twitch-on-air-sign>

Last updated on 2024-06-03 01:53:16 PM EDT

# Table of Contents

Overview	3
• Previous projects	
Twitch API	4
Querying API	6
Final Sketch	11
• Sketch Explanation	
• Configuration	
• Setup	
• Loop	
Sign Assembly	13

---

# Overview

Twitch has changed their API so this tutorial will no longer work as written



We're doing a lot of streaming lately, and I wanted to make a sign that would let people know when we're on air.

All this guide will do is connect to the Twitch API and determine if the user is currently streaming - if so, the Feather will turn on some NeoPixels (you can also just use LEDs if you like) to light up the sign.

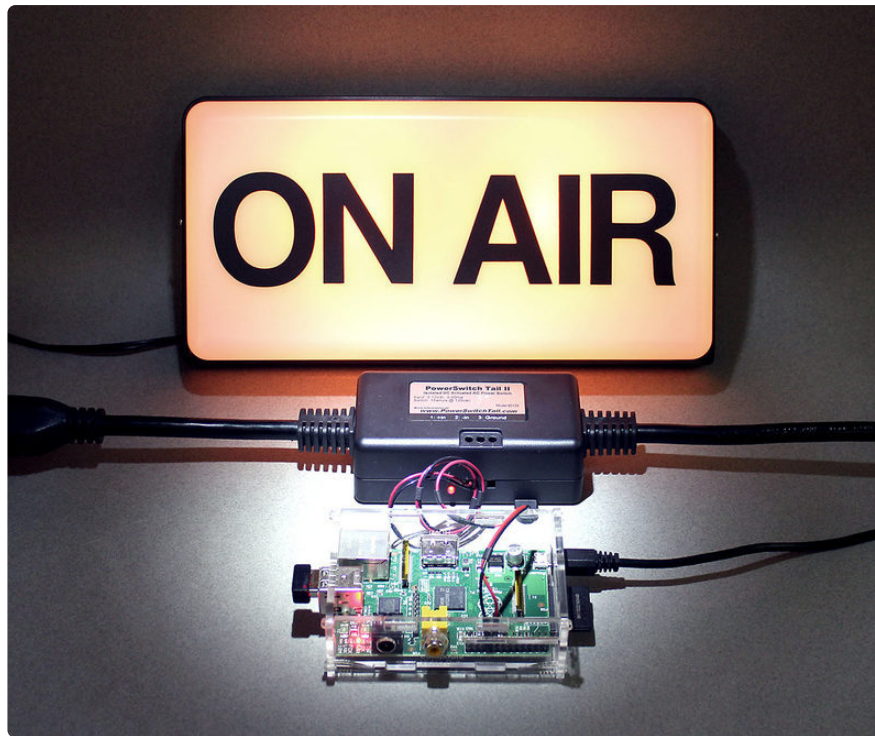
You can easily extend this to, instead of parsing for a live stream, display the stream name, number of viewers, game-being-played or [anything else available from the Twitch API \(https://adafru.it/ITd\)](https://adafru.it/ITd)

I did this project as an IoT-Tuesday project and livestreamed it on Youtube, you can watch it here:

(It turns out the SSL failure on ESP8226 is a timeout not an encryption issue, so you can also do this project with an ESP8266!)

## Previous projects

[We did this once already with a raspberry Pi to be able to connect to YouTube or UStream's API but it really does require quite a bit of hacking \(https://adafru.it/ITf\)](https://adafru.it/ITf)



I wanted to make something simpler, all-in-one, using a Feather ESP8266 or WINC1500. In the end, the Feather M0 ATWINC1500 & ESP8266 worked best, since you need to have an SSL connection to twitch.tv but at least you don't need any OAuth or API keys!

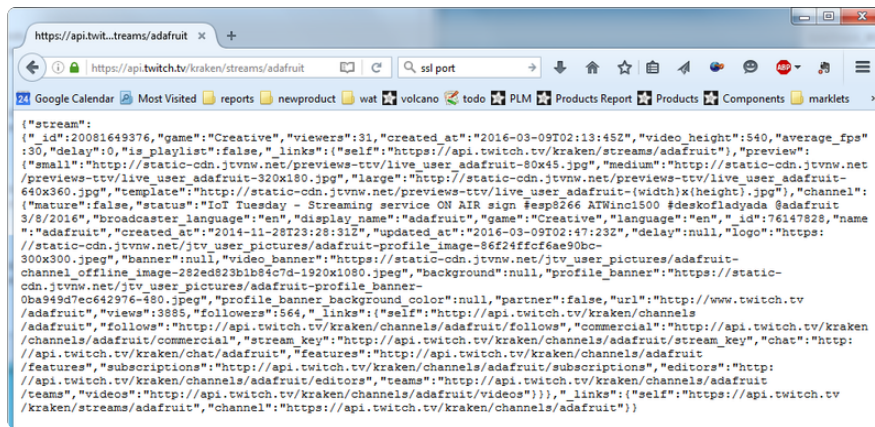
---

## Twitch API

Lucky for us, instead of forcing us to use OAuth or API keys or sacrificing chickens, twitch makes it really easy to see what's going on with any stream. Yay!

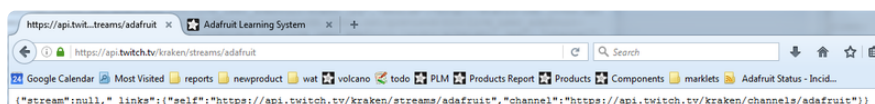
All you have to do is go to <https://api.twitch.tv/kraken/streams/adafruit> (<https://adafruit.it/ITc>) and you'll be able to see the current status of the **adafruit** stream. Just change the last part of the url to see another stream

The data will be a json package, in ascii text. For example, if the stream is active:



```
{
  "stream": {
    "_id": 20082214992,
    "game": null,
    "viewers": 0,
    "created_at": "2016-03-09T02:51:06Z",
    "video_height": 540,
    "average_fps": 30,
    "delay": 0,
    "is_playlist": false,
    "_links": {
      "self": "https://api.twitch.tv/kraken/streams/ladyada1337",
      "preview": {
        "small": "http://static-cdn.jtvnw.net/previews-ttv/live_user_ladyada1337-80x45.jpg",
        "medium": "http://static-cdn.jtvnw.net/previews-ttv/live_user_ladyada1337-320x180.jpg",
        "large": "http://static-cdn.jtvnw.net/previews-ttv/live_user_ladyada1337-640x360.jpg",
        "template": "http://static-cdn.jtvnw.net/previews-ttv/live_user_{width}x{height}.jpg"
      },
      "channel": {
        "mature": null,
        "status": null,
        "broadcaster_language": null,
        "display_name": "ladyada1337",
        "game": null,
        "id": 113575809,
        "name": "ladyada1337",
        "created_at": "2016-01-23T00:38:01Z",
        "updated_at": "2016-03-09T02:51:06Z",
        "views": 11,
        "followers": 0,
        "_links": {
          "self": "http://api.twitch.tv/kraken/channels/ladyada1337",
          "follows": "http://api.twitch.tv/kraken/channels/ladyada1337/follows",
          "commercial": "http://api.twitch.tv/kraken/channels/ladyada1337/commercial",
          "stream_key": "http://api.twitch.tv/kraken/channels/ladyada1337/stream_key",
          "chat": "http://api.twitch.tv/kraken/chat/ladyada1337",
          "features": "http://api.twitch.tv/kraken/channels/ladyada1337/features",
          "subscriptions": "http://api.twitch.tv/kraken/channels/ladyada1337/subscriptions",
          "editors": "http://api.twitch.tv/kraken/channels/ladyada1337/editors",
          "teams": "http://api.twitch.tv/kraken/channels/ladyada1337/teams",
          "videos": "http://api.twitch.tv/kraken/channels/ladyada1337/videos"
        }
      }
    }
  },
  "_links": {
    "self": "https://api.twitch.tv/kraken/streams/ladyada1337",
    "channel": "https://api.twitch.tv/kraken/channels/ladyada1337"
  }
}
```

And if not active...



```
{
  "stream": null,
  "_links": {
    "self": "https://api.twitch.tv/kraken/streams/adafruit",
    "channel": "https://api.twitch.tv/kraken/channels/adafruit"
  }
}
```

There is a ton of details over at <https://github.com/justintv/Twitch-API> (<https://adafru.it/ITd>) but we're going to be lazy and just parse the data manually.

If we see the string `{"stream":{"_id":...}}` then we know the stream is active. If we see the string `{"stream":null}` then we know it is not.

One thing to note is you **must** use **SSL** to connect, if you try to connect without, you'll get

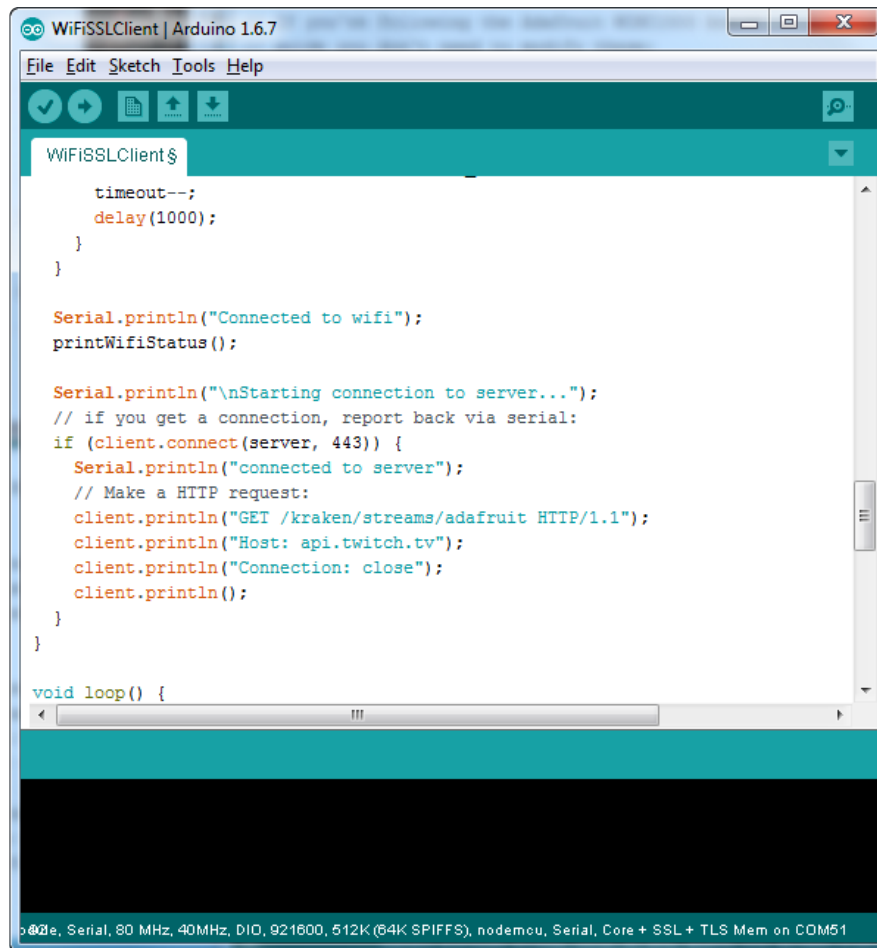
```
{"error": "Bad Request", "message": "Requests must be made over SSL", "status": 400}
```

## Querying API

OK now that we know we just need to connect with SSL, so open up the **Adafruit\_WINC1500->WiFiSSLClient** example, and change the server to **api.twitch.tv**



and the actual query to grab the twitch api url (you'll need to update the path and the host)



Here's the complete edited code, you'll also need to update the network SSID & password

```

#include <SPI.h>;
#include <Adafruit_WINC1500.h>;

// Define the WINC1500 board connections below.
// If you're following the Adafruit WINC1500 board
// guide you don't need to modify these:
#define WINC_CS 8
#define WINC_IRQ 7
#define WINC_RST 4
#define WINC_EN 2 // or, tie EN to VCC and comment this out
// The SPI pins of the WINC1500 (SCK, MOSI, MISO) should be
// connected to the hardware SPI port of the Arduino.
// On an Uno or compatible these are SCK = #13, MISO = #12, MOSI = #11.
// On an Arduino Zero use the 6-pin ICSP header, see:
// https://www.arduino.cc/en/Reference/SPI

// Setup the WINC1500 connection with the pins above and the default hardware SPI.
Adafruit_WINC1500 WiFi(WINC_CS, WINC_IRQ, WINC_RST);

// Or just use hardware SPI (SCK/MOSI/MISO) and defaults, SS -> #10, INT ->
// #7, RST -> #5, EN -> 3-5V
//Adafruit_WINC1500 WiFi;

char ssid[] = "networkssid"; // your network SSID (name)
char pass[] = "password"; // your network password (use for WPA, or use as key
for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;

```



```

char server[] = "api.twitch.tv";    // name address for Google (using DNS)

Adafruit_WINC1500SSLClient client;

void setup() {
#ifdef WINC_EN
  pinMode(WINC_EN, OUTPUT);
  digitalWrite(WINC_EN, HIGH);
#endif

  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while (true);
  }

  // attempt to connect to Wifi network:
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    uint8_t timeout = 10;
    while (timeout && (WiFi.status() != WL_CONNECTED)) {
      timeout--;
      delay(1000);
    }
  }

  Serial.println("Connected to wifi");
  printWifiStatus();

  Serial.println("\nStarting connection to server...");
  // if you get a connection, report back via serial:
  if (client.connect(server, 443)) {
    Serial.println("connected to server");
    // Make a HTTP request:
    client.println("GET /kraken/streams/adafruit HTTP/1.1");
    client.println("Host: api.twitch.tv");
    client.println("Connection: close");
    client.println();
  }
}

void loop() {
  // if there are incoming bytes available
  // from the server, read them and print them:
  while (client.available()) {
    char c = client.read();
    Serial.write(c);
  }

  // if the server's disconnected, stop the client:
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting from server.");
    client.stop();

    // do nothing forevermore:
    while (true);
  }
}

```



```

    }
}

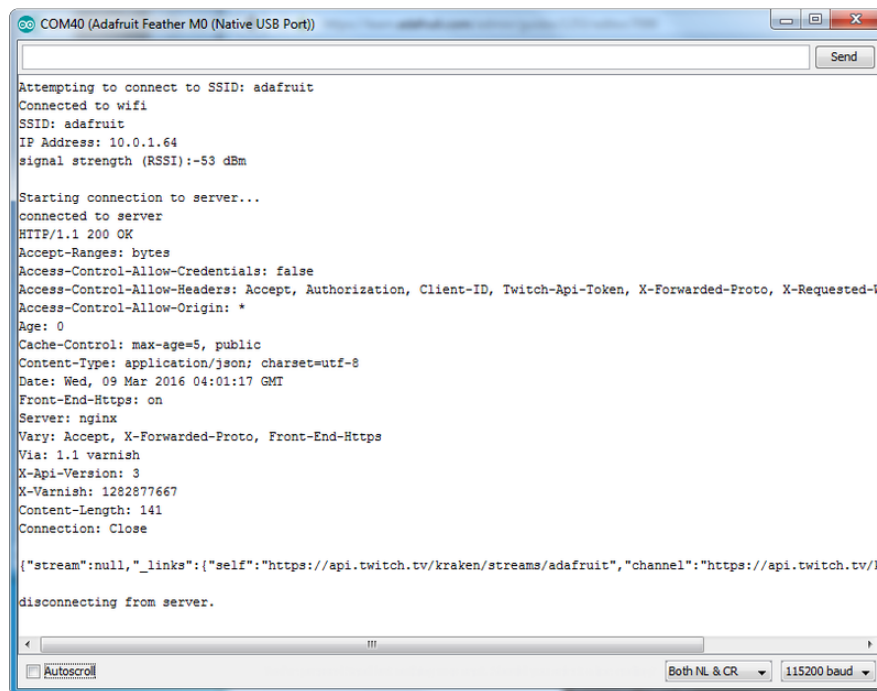
void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

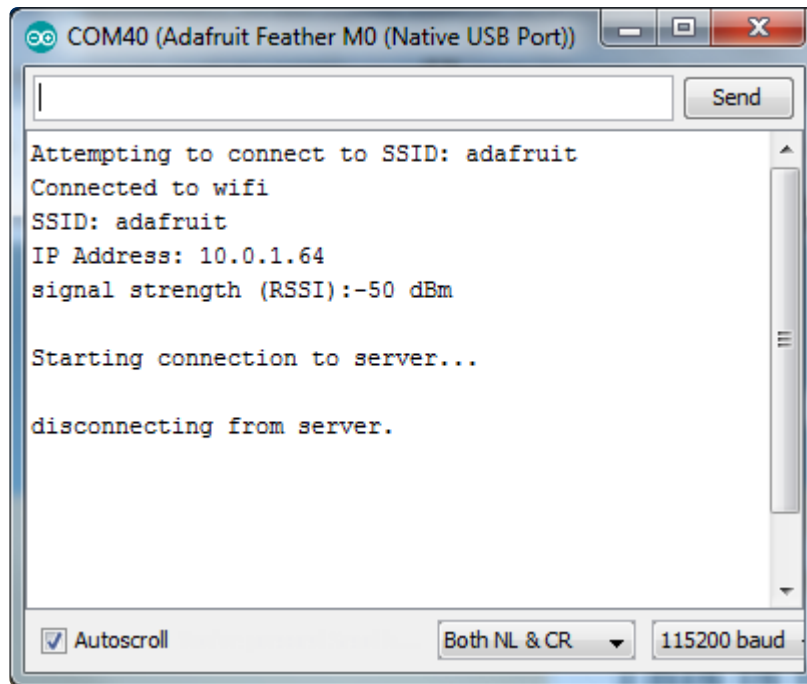
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}

```

Upload to your Feather WING and check that you can connect:

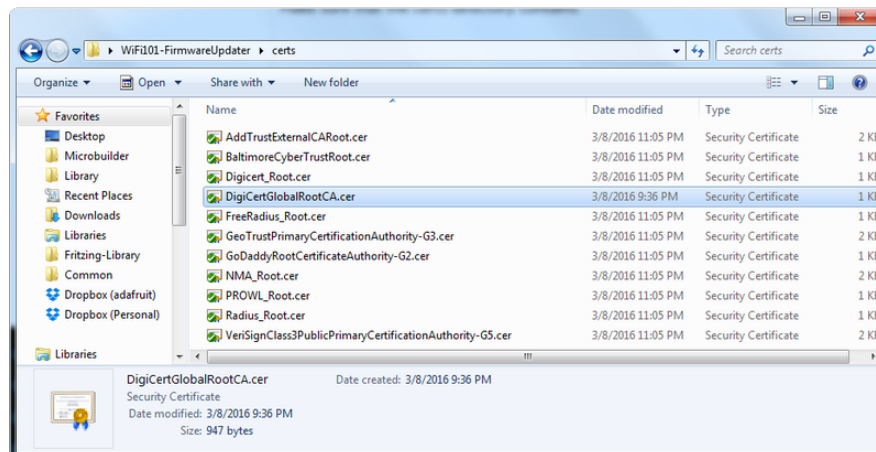


If you can connect to the SSID but then can't connect to twitch, you might be missing the SSL root cert for twitch (DigiCertGlobalRootCA)

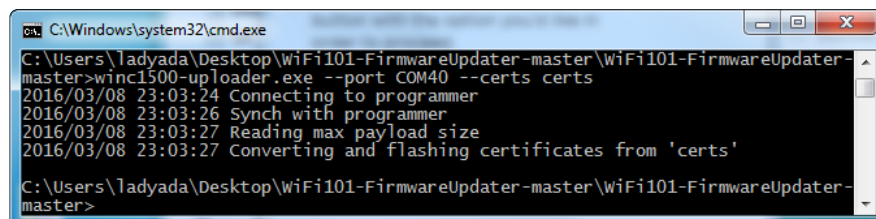


In which case, [run the instructions from this tutorial for updating the SSL certificates \(https://adafru.it/ITe\)](https://adafru.it/ITe)

Before doing so, make sure that the certs directory contains the DigiCertGlobal certificate file



Then upload the **FirmwareUploader** sketch & run the cert updater



Then re-upload the SSL connection test, make sure you can connect to twitch and get that json string

---

# Final Sketch

Once that works, upload either of these sketches, with your SSID & network password:

- [WINC1500 Demo \(https://adafru.it/m1e\)](https://adafru.it/m1e)
- [ESP8266 Demo \(https://adafru.it/m1f\)](https://adafru.it/m1f)

## Sketch Explanation

This sketch is not much different than the SSL test sketch.

## Configuration

In the pre-setup configuration, you can set up your SSID & password. Also, set what pin has the LED connected. Define the path you want to grab, change this to your favorite twitch streamer

```
// Define the WINC1500 board connections below.
// If you're following the Adafruit WINC1500 board
// guide you don't need to modify these:
#define WINC_CS 8
#define WINC_IRQ 7
#define WINC_RST 4
#define WINC_EN 2 // or, tie EN to VCC and comment this out

#define LED 13

// Setup the WINC1500 connection with the pins above and the default hardware SPI.
Adafruit_WINC1500 WiFi(WINC_CS, WINC_IRQ, WINC_RST);

// Or just use hardware SPI (SCK/MOSI/MISO) and defaults, SS -> #10, INT ->
#7, RST -> #5, EN -> 3-5V
//Adafruit_WINC1500 WiFi;

char ssid[] = "networkssid"; // your network SSID (name)
char pass[] = "networkpass"; // your network password (use for WPA, or use as
key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;
#define HOST "api.twitch.tv"
#define PATH "/kraken/streams/adafruit"
#define REFRESH 20 // seconds between refresh

Adafruit_WINC1500SSLClient client;
```

## Setup

In **setup** we just check that the WiFi module is attached, and set the pin #13 LED to an output

```
void setup() {
#ifdef WINC_EN
  pinMode(WINC_EN, OUTPUT);
  digitalWrite(WINC_EN, HIGH);
#endif

  pinMode(LED, OUTPUT);

  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while (true);
  }
}
```

## Loop

We start by checking if we are connected to the SSID, if not we connect

```
void loop() {
  // attempt to connect to Wifi network:
  if (WiFi.status() != WL_CONNECTED) {
    while (WiFi.status() != WL_CONNECTED) {
      Serial.print("Attempting to connect to SSID: ");
      Serial.println(ssid);
      // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
      status = WiFi.begin(ssid, pass);

      // wait 10 seconds for connection:
      uint8_t timeout = 10;
      while (timeout &&& (WiFi.status() != WL_CONNECTED)) {
        timeout--;
        delay(1000);
      }
    }

    Serial.println("Connected to wifi");
    printWifiStatus();
  }
}
```

Then we SSL connect to twitch.tv and ask for the path from the setup section

```
Serial.println("\nStarting connection to twitch...");
// if you get a connection, report back via serial:
if (client.connect(HOST, 443)) {
```

```

Serial.println("connected to twitch api server");
// Make a HTTP request:
client.println("GET " PATH " HTTP/1.1");
client.println("Host: " HOST);
client.println("Connection: close");
client.println();
}

```

We then use **find** to locate whether the magic string appears

```

boolean isStreaming = false;
while (client.connected()) {
  if (client.find("\"stream\":{\"_id\":}")) {
    isStreaming = true;
  }
}

```

if so, then **isStreaming** is set to true. Finally we print out the current status, wait a few seconds, and restart the loop

```

Serial.print("Streaming status: "); Serial.println(isStreaming);
digitalWrite(LED, isStreaming);
Serial.println("disconnecting from server.");
client.stop();

delay(REFRESH*1000);
}

```

---

## Sign Assembly



I started with an off-the-shelf ON AIR sign, that just plugs into a 12V power supply



If you open up the sign you'll see a little holder for two lamps, it's pretty basic!



I removed the lamps

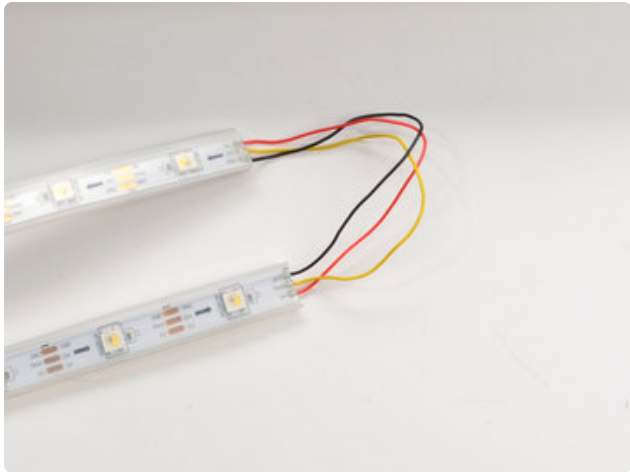


And cut some RGBW NeoPixel strip I had handy



I decided to go with two strips of 30 LED/ meter





Some thin wires connect the two strips



Some foam tape or packing tape can be used to keep the strips in place. Solder the strip wires to the Feather, the Power pin goes to USB, Ground to Ground, and Data to #12



I plugged in a [USB wall adapter with a MicroUSB plug \(http://adafru.it/1995\)](http://adafru.it/1995) to power it. You can also go with a long USB cable, that way you could plug it into a USB port and also reprogram it when necessary



Plug it in and rock out!

The final code will turn on the NeoPixel RGBW's dimly when powered up and bright red if the stream is detected. You may need to change the NeoPixel definition to RGB instead of RGBW if you're using plain NeoPixels