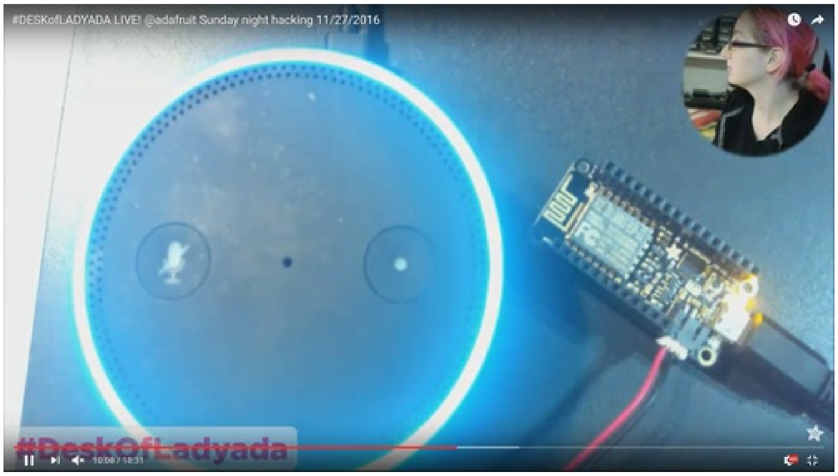
Easy Alexa (Echo) Control of your ESP8266 Huzzah

Created by lady ada


#DESKofLADYADA LIVE! @adafruit Sunday night hacking 11/27/2016
#DeskOfLadyada
10:08 / 18:31

Last updated on 2017-01-24 09:59:32 AM UTC

Guide Contents

Overview

If you've already got an Amazon Echo or Echo Dot in your home or office, you can easily add your very own devices. In this quick project we'll show how to use an Adafruit ESP8266 Feather HUZZAH to control NeoPixels or a relay.

You can easily adapt the code to add any number of devices to a single Feather for all sorts of interactive home automation projects with ease!

No external server, gateway, service, computer, Raspberry Pi, etc. required!

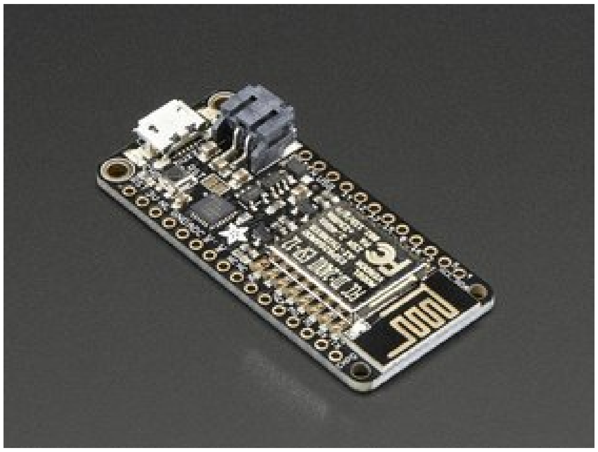
Credits!

Code and tutorial based on the great open source lib & example code at http://tinkerman.cat/emulate-wemo-device-esp8266/ (http://adafru.it/sES) which is based off of the Python example code by https://github.com/makermusings/fauxmo (http://adafru.it/sET)

thanks!

Required Parts!

You'll need an ESP8266 of some sort. We suggest our Feather (all-in one!)

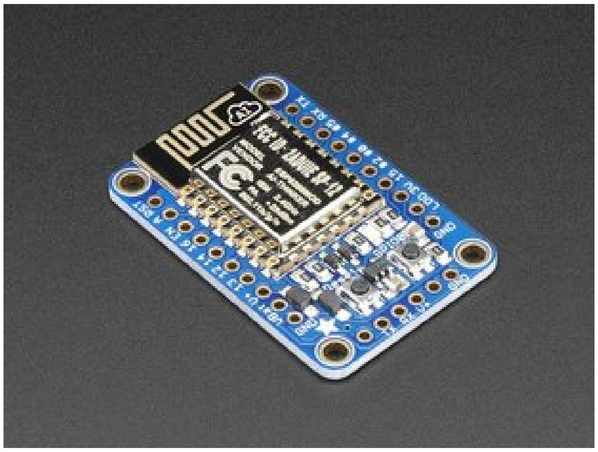


Adafruit Feather HUZZAH with ESP8266 WiFi

PRODUCT ID: 2821
Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller cores....
http://adafru.it/n6A

$15.95
IN STOCK

Or HUZZAH breakout (harder to use, requires an FTDI cable or USB-Serial adapter)



Adafruit HUZZAH ESP8266 Breakout

PRODUCT ID: 2471
Add Internet to your next project with an adorable, bite-sized WiFi microcontroller, at a price you like! The ESP8266 processor from Espressif is an 80 MHz microcontroller with a full...
http://adafru.it/f9X
$9.95
IN STOCK

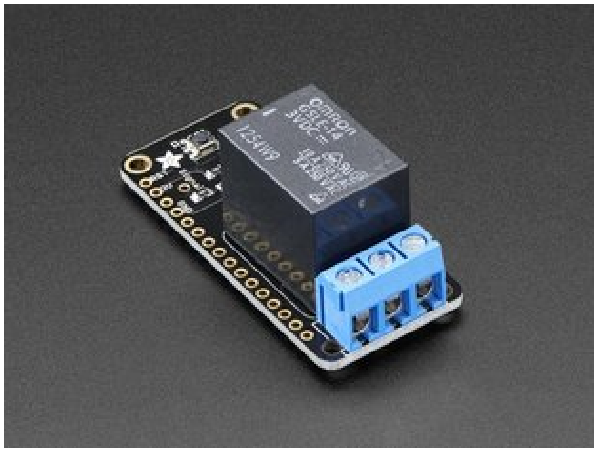
You'll also need a USB cable for the above and/or a battery for portable use.

Our demo code also uses a Relay FeatherWing and a NeoPixel Featherwing

Your browser does not support the video tag.

NeoPixel FeatherWing - 4x8 RGB LED Add-on For All Feather Boards

PRODUCT ID: 2945
A Feather board without ambition is a Feather board without FeatherWings! This is the NeoPixel FeatherWing, a 4x8 RGB LED Add-on For All Feather Boards ! Using...
http://adafru.it/sEl
$14.95
IN STOCK



Adafruit Power Relay FeatherWing

PRODUCT ID: 3191
A Feather board without ambition is a Feather board without FeatherWings! This is the Power Relay FeatherWing . It gives you power to control, and control over power. Put simply, you can...
http://adafru.it/sEJ
$9.95
IN STOCK

You can also use a Powerswitch tail if you want to control an outlet without splicing any cables

Powerswitch tail 2

PRODUCT ID: 268
The Power Switch Tail II is a smart alternative to slicing apart power cords to wire up your own relays. Its a compact 120V 3-pronged extension cord, with a relay board embedded in the...
http://adafru.it/eml
$25.95
IN STOCK

Software setup

Pre-Requisite Setup

Before you begin you'll want to make sure your Feather ESP8266 is running properly, you have drivers installed, Arduino IDE etc.

Visit the product tutorial for the Feather ESP8266 to get setup, once you've tested it out and got it all working, come back! (http://adafru.it/nEN)

Required Libraries

You'll need a few libraries installed in Arduino to continue. Here's a list of them, install by downloading the linked Zip file, uncompressing and installing in your sketchbook's **libraries** folder. See our Arduino Libraries Guide for details if you've never installed libraries(http://adafru.it/sEK)

1. ESPAsyncTCP library (http://adafru.it/sEL)(to download zip click here (http://adafru.it/sEM))
2. ESPAsyncUDP library (http://adafru.it/sEN) (to download zip click here (http://adafru.it/sEO))
3. FauxMoESP library (http://adafru.it/sEP) (to download zip click here (http://adafru.it/sEQ) and then select *download repository*)

Compile an Example

Instead of the example that comes with FauxMoESP, try this all-in-one sketch.

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include "fauxmoESP.h"

#define WIFI_SSID "..."
#define WIFI_PASS "..."
#define SERIAL_BAUDRATE                 115200

fauxmoESP fauxmo;

// -----------------------------------------------------------------------------
// Wifi
// -----------------------------------------------------------------------------

void wifiSetup() {

  // Set WIFI module to STA mode
  WiFi.mode(WIFI_STA);
```

```
    // Connect
    Serial.printf("[WIFI] Connecting to %s ", WIFI_SSID);
    WiFi.begin(WIFI_SSID, WIFI_PASS);

    // Wait
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(100);
    }
    Serial.println();

    // Connected!
    Serial.printf("[WIFI] STATION Mode, SSID: %s, IP address: %s\n", WiFi.SSID().c_str(), WiFi.localIP().toString().c_str());
}

void callback(uint8_t device_id, const char * device_name, bool state) {
  Serial.print("Device "); Serial.print(device_name);
  Serial.print(" state: ");
  if (state) {
    Serial.println("ON");
  } else {
    Serial.println("OFF");
  }
}

void setup() {
    // Init serial port and clean garbage
    Serial.begin(SERIAL_BAUDRATE);
    Serial.println("FauxMo demo sketch");
    Serial.println("After connection, ask Alexa/Echo to 'turn <devicename> on' or 'off'");

    // Wifi
    wifiSetup();

    // Fauxmo
    fauxmo.addDevice("relay");
    fauxmo.addDevice("pixels");
    fauxmo.onMessage(callback);
}

void loop() {
  fauxmo.handle();
}
```

Just change

```
#define WIFI_SSID "..."
#define WIFI_PASS "..."
```

To your SSID & Password!

Try to compile it (no need to upload yet!)

Compilation Problems?

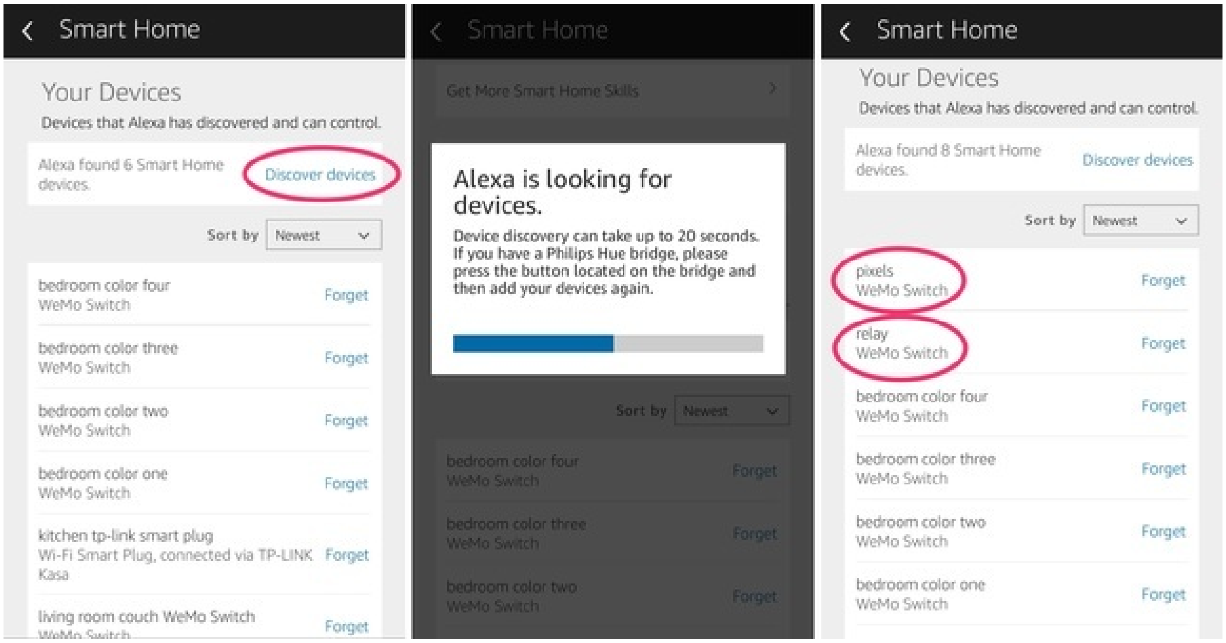
Please note: as of November 27, 2016 the ESP8266 release core v2.3.0 did not have the right lwip code so there were some compilation issues. If you get a complaint about udp_set_multicast_ttl not being defined, you'll need to *uninstall* the ESP8266 board support, then *manually* install the most recent core fromhttps://github.com/esp8266/arduino (http://adafru.it/sER) by following the instructions at https://github.com/esp8266/arduino#using-git-version (http://adafru.it/sER)

Essentialy you'll need to `git clone https://github.com/esp8266/Arduino.git` into your Arduino sketchbook folder under **hardware/esp8266com/esp8266** and then in a terminal shell in **hardware/esp8266com/esp8266/tools** run `python get.py`

Once that's done, go back and test out your ESP8266 to make sure you can compile/upload code to it.

Upload & Test!

Once you've compiled and uploaded the code, open up the Alexa app on your phone, select the **Smart Home** item from the menu, scroll down to**Your Devices**, and tap **Discover Devices.**



Smart Home

Your Devices

Devices that Alexa has discovered and can control.

Alexa found 6 Smart Home devices. Discover devices

Sort by Newest

bedroom color four
WeMo Switch Forget

bedroom color three
WeMo Switch Forget

bedroom color two
WeMo Switch Forget

bedroom color one
WeMo Switch Forget

kitchen tp-link smart plug
Wi-Fi Smart Plug, connected via TP-LINK Kasa Forget

living room couch WeMo Switch
WeMo Switch Forget

Smart Home

Get More Smart Home Skills

Alexa is looking for devices.

Device discovery can take up to 20 seconds. If you have a Philips Hue bridge, please press the button located on the bridge and then add your devices again.

Sort by Newest

bedroom color four
WeMo Switch Forget

bedroom color three
WeMo Switch Forget

bedroom color two
WeMo Switch Forget

Smart Home

Your Devices

Devices that Alexa has discovered and can control.

Alexa found 8 Smart Home devices. Discover devices

Sort by Newest

pixels
WeMo Switch Forget

relay
WeMo Switch Forget

bedroom color four
WeMo Switch Forget

bedroom color three
WeMo Switch Forget

bedroom color two
WeMo Switch Forget

bedroom color one
WeMo Switch Forget

The Alexa app should recognize the new**pixels** & **relay** devices as seen above. If it doesn't, try resetting the Feather and running the discovery process again.

Now open up the Arduino IDE serial console at 115200 baud.

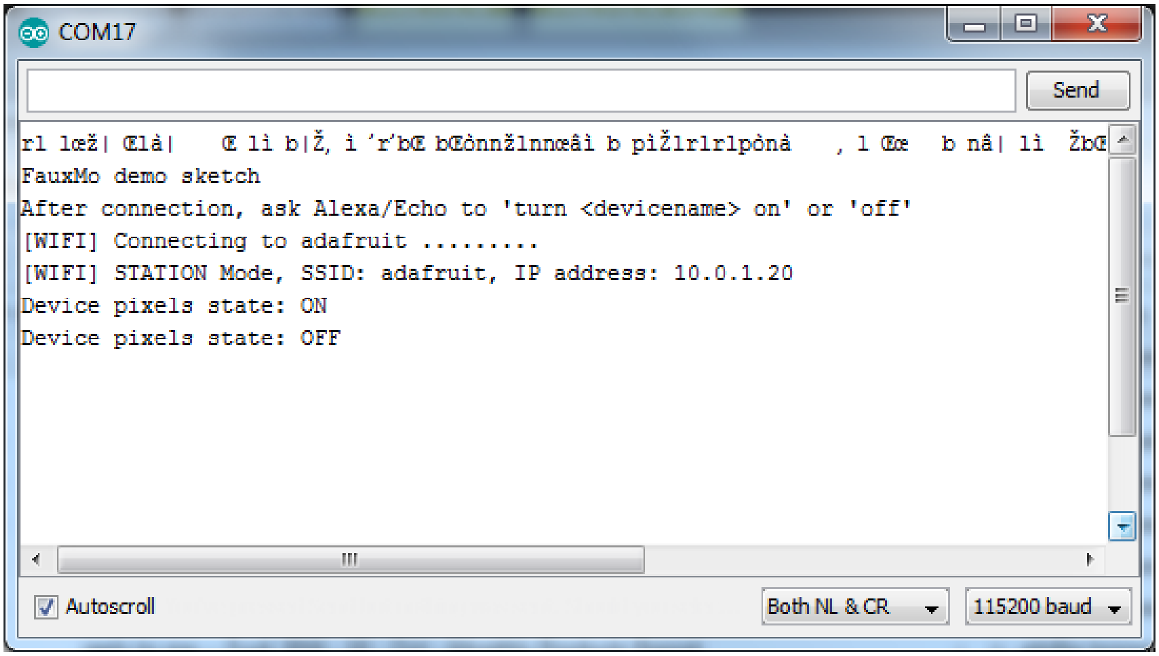
Ask your Alexa (or Echo):

"Alexa, turn pixels on"

and

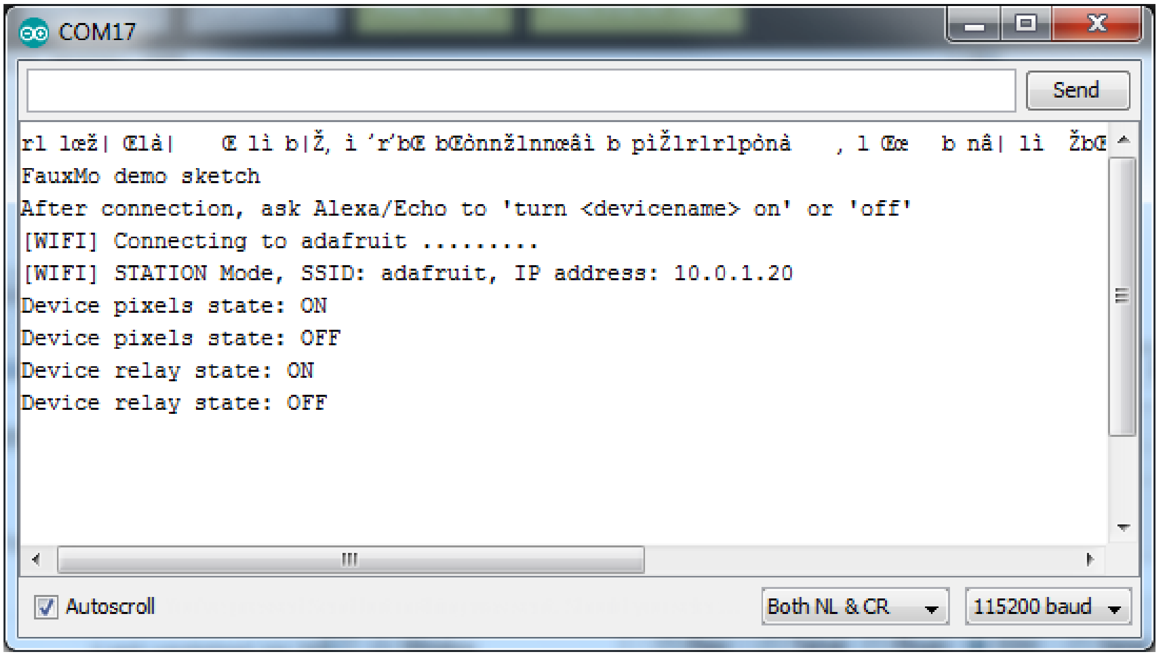
"Alexa, turn pixels off"

You'll see the ESP8266 print out the commands it received!



```
rl lœž| Œlà|    Œ lì b|Ž, ì ʼrʼbŒ bŒònnžlnnœâì b piŽlrlrlpònà    , l Œœ   b nâ| lì   ŽbŒ
FauxMo demo sketch
After connection, ask Alexa/Echo to 'turn <devicename> on' or 'off'
[WIFI] Connecting to adafruit .........
[WIFI] STATION Mode, SSID: adafruit, IP address: 10.0.1.20
Device pixels state: ON
Device pixels state: OFF
```

You can also try it with**relay**, the second device it is emulating



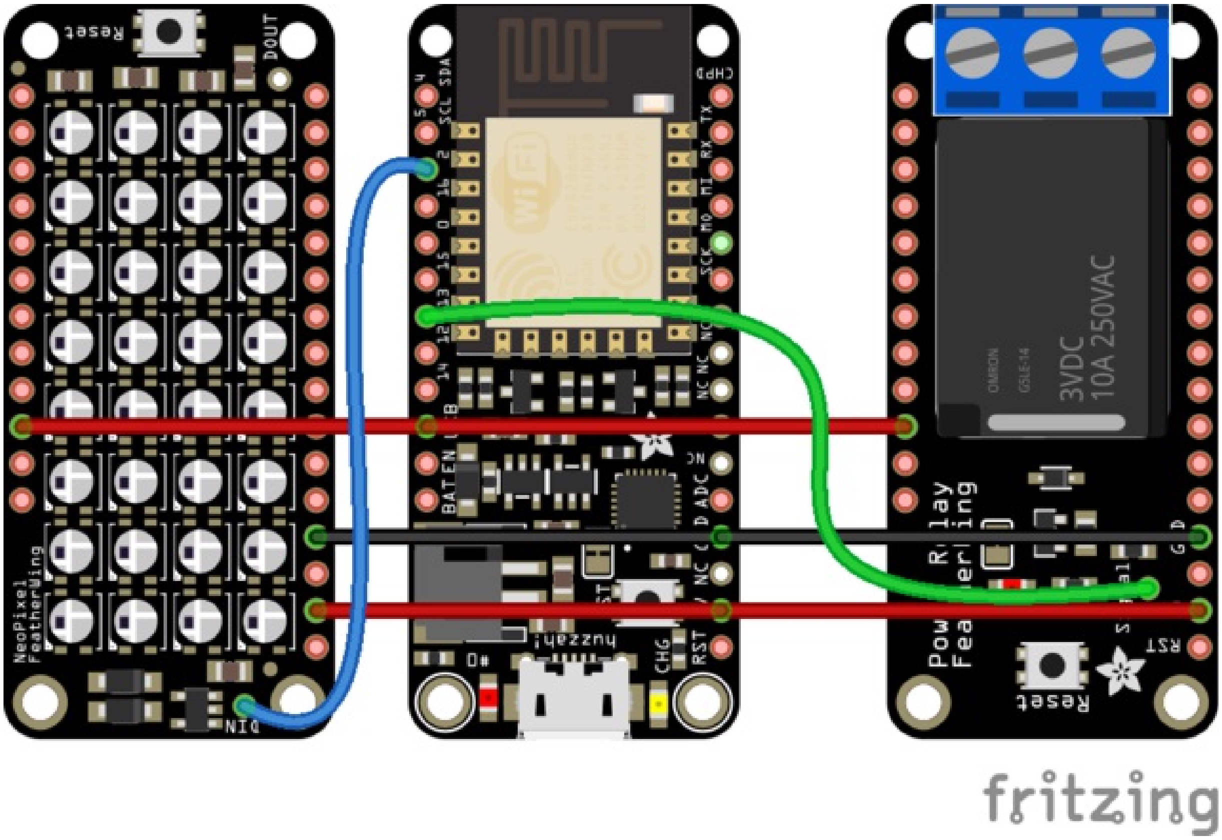
```
rl lœž| Œlà|    Œ lì b|Ž, ì ʼrʼbŒ bŒònnžlnnœâì b piŽlrlrlpònà    , l Œœ   b nâ| lì   ŽbŒ
FauxMo demo sketch
After connection, ask Alexa/Echo to 'turn <devicename> on' or 'off'
[WIFI] Connecting to adafruit .........
[WIFI] STATION Mode, SSID: adafruit, IP address: 10.0.1.20
Device pixels state: ON
Device pixels state: OFF
Device relay state: ON
Device relay state: OFF
```

Now that you have it working you can change the names of the devices. Try to make them things that are easy to pronounce so the Alexa/Echo can understand them when yelled! :)

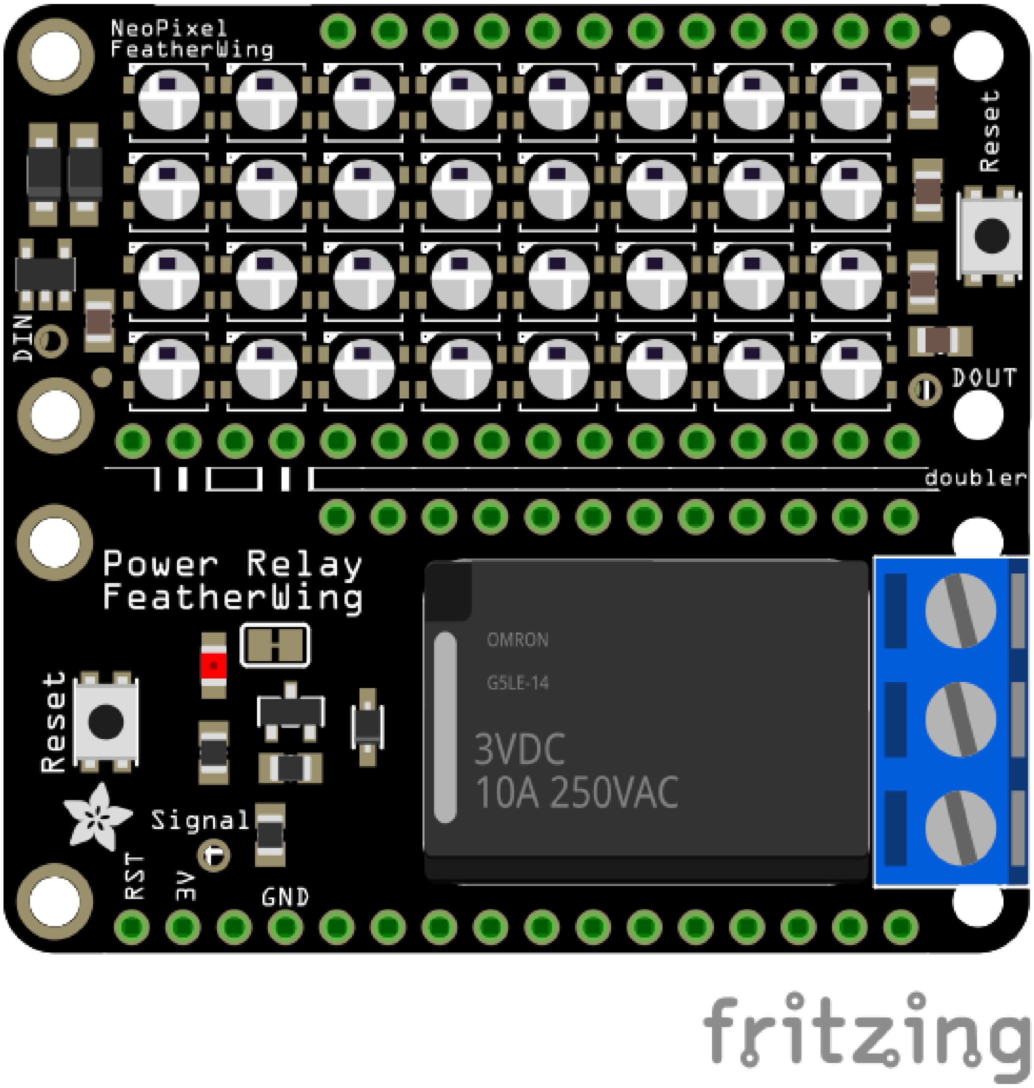
NeoPixel + Relay Demo

Thanks to the easy-to-use Feather system, we can quickly craft a working demo with lights and a relay. Grab a NeoPixel FeatherWing and a Relay FeatherWing (there's three types of Relay 'wings, use any which one you like)

We'll connect the NeoPixel data line to ESP pin #2 and the relay signal to #13 (you can change these pins around, except NeoPixel can't be on pin #16). You can either 'freewire' these boards together and just make sure you get the power and ground pins...



Or use a Featherwing Doubler + stacking headers on the ESP8266, and then stack the NeoPixel wing on top of it. Then put the relay board on the side



Then just make sure you cut/solder the trace on the bottom of each wing so that it is connected to the digital control pin specified above

Test Wings

You may want to go through the tutorials for both relay and NeoPixel wing to make sure they're working on the pins you want! That way you wont have to debug both the Alexa/WeMo code as well as your hardware

Full Demo Code

Now upload this sketch:

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <ESP8266WiFi.h>
```

```
#include "fauxmoESP.h"

#define WIFI_SSID "adafruit"
#define WIFI_PASS "ffffffff"

#define SERIAL_BAUDRATE                 115200

fauxmoESP fauxmo;

#define RELAY_PIN 13
#define NEOPIX_PIN 2
Adafruit_NeoPixel strip = Adafruit_NeoPixel(40, NEOPIX_PIN, NEO_GRB + NEO_KHZ800);
volatile boolean neopixel_state = false; // off by default!

uint32_t Wheel(byte WheelPos); // function prototype

// -----------------------------------------------------------------------------
// Wifi
// -----------------------------------------------------------------------------


void wifiSetup() {
  // Set WIFI module to STA mode
  WiFi.mode(WIFI_STA);

  // Connect
  Serial.printf("[WIFI] Connecting to %s ", WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASS);

  // Wait
  while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
      delay(100);
  }
  Serial.println();

  // Connected!
  Serial.printf("[WIFI] STATION Mode, SSID: %s, IP address: %s\n", WiFi.SSID().c_str(), WiFi.localIP().toString().c_str());
}


void callback(uint8_t device_id, const char * device_name, bool state) {
  Serial.printf("[MAIN] %s state: %s\n", device_name, state ? "ON" : "OFF");

  if ( (strcmp(device_name, "pixels") == 0) ) {
    // this just sets a variable that the main loop() does something about
    if (state) {
      neopixel_state = true;
    } else {
      neopixel_state = false;
    }
  }

  if ( (strcmp(device_name, "relay") == 0) ) {
    // adjust the relay immediately!
    if (state) {
      digitalWrite(RELAY_PIN, HIGH);
    } else {
      digitalWrite(RELAY_PIN, LOW);
    }
  }
}

void setup() {
  strip.begin();
  strip.setBrightness(20);
  strip.show(); // Initialize all pixels to 'off'
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);

    // Init serial port and clean garbage
    Serial.begin(SERIAL_BAUDRATE);
    Serial.println();
    Serial.println();
    Serial.println("FauxMo demo sketch");
    Serial.println("After connection, ask Alexa/Echo to 'turn pixels on' or 'off' or 'turn relay on' or 'off'");

  // Wifi
  wifiSetup();

  // Fauxmo
  fauxmo.addDevice("relay");
  fauxmo.addDevice("pixels");
  fauxmo.onMessage(callback);
}

uint8_t j = 0; // color swirl incrementer
void loop() {
  fauxmo.handle();
  if (neopixel_state) {
    for(int16_t i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    j++;
    delay(20);
  } else {
    for(int16_t i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, 0);
    }
    strip.show();
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Since there's two devices we'll show two ways of using the call-back function to do what you want.

The callback

The way the code works is that when a new command is received, the void callback(uint8_t device_id, const char * device_name, bool state) function is called with the *name* as a character array and then a boolean with true/false depending on whether the device should be on or off.

The state boolean is easy to deal with. For the character array, you'll want to use **strcmp** which will do a **str**ing **comp**arison for you. If the strings are identical, it will return 0.

Thus to test if you got a **relay** command, the line if ((strcmp(device_name, "relay") == 0)) will test to see if that's the device requested!

In-line control (Relay)

Once you know what string is sent, the code can do something simple based on the **state** variable. For example, just turn a pin on or off!

```
  if ( (strcmp(device_name, "relay") == 0) ) {
    // adjust the relay immediately!
    if (state) {
      digitalWrite(RELAY_PIN, HIGH);
    } else {
      digitalWrite(RELAY_PIN, LOW);
    }
  }
```

External Flag control (NeoPixels)

For NeoPixels, we want to have the pixels do a rainbow swirl in **loop** so that we aren't in the callback for more than a millisecond. Rather than halt the callback to do pixel management, we set an external **volatile** variable. It's key that we indicate it is **volatile** because that tells the ESP8266 "hey, this variable will change in a callback or interrupt so don't assume it's the same value it was a second ago"

```
volatile boolean neopixel_state = false; // off by default!

...

  if ( (strcmp(device_name, "pixels") == 0) ) {
    // this just sets a variable that the main loop() does something about
    if (state) {
      neopixel_state = true;
    } else {
      neopixel_state = false;
    }
  }
```

Then in the main loop() code, we can test to see if **neopixel_state** is true or false. If it's **true**, it should continue with the color swirl output. The swirl code is from the **rainbow()** NeoPixel example, by the way, it just sets each pixel to an equidistant color on the color wheel.

```
  if (neopixel_state) {
    for(int16_t i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    j++;
    delay(20);
  }
```

If it's **false**, turn off all the pixels

```
else {
  for(int16_t i=0; i< strip.numPixels(); i++) {
    strip.setPixelColor(i, 0);
  }
  strip.show();
}
```

Now that you've got it working - try other devices like servos, motors, IR or RF remote controls, etc! Maybe call the device "party" and then you can tell Alexa to "turn the party on"?