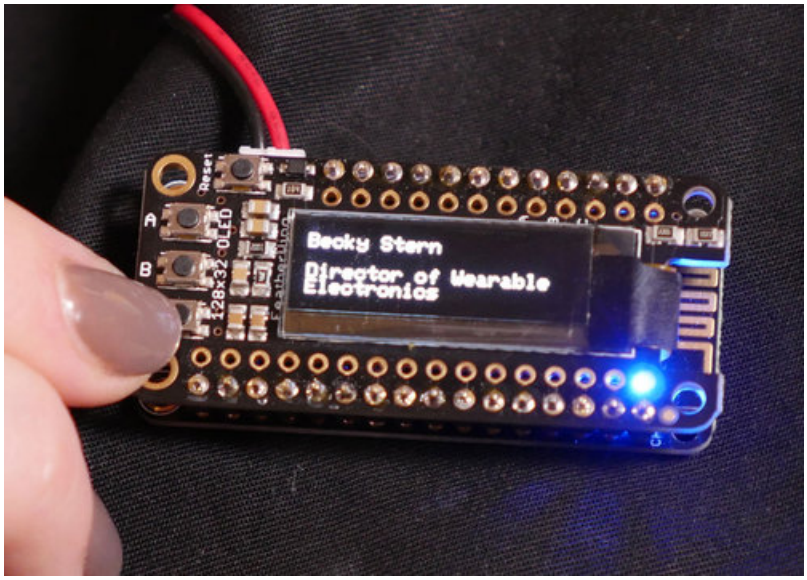
adafruit learning system

# WiFi OLED Display Badge

Created by Becky Stern




Last updated on 2020-05-19 12:32:34 AM EDT

# Overview

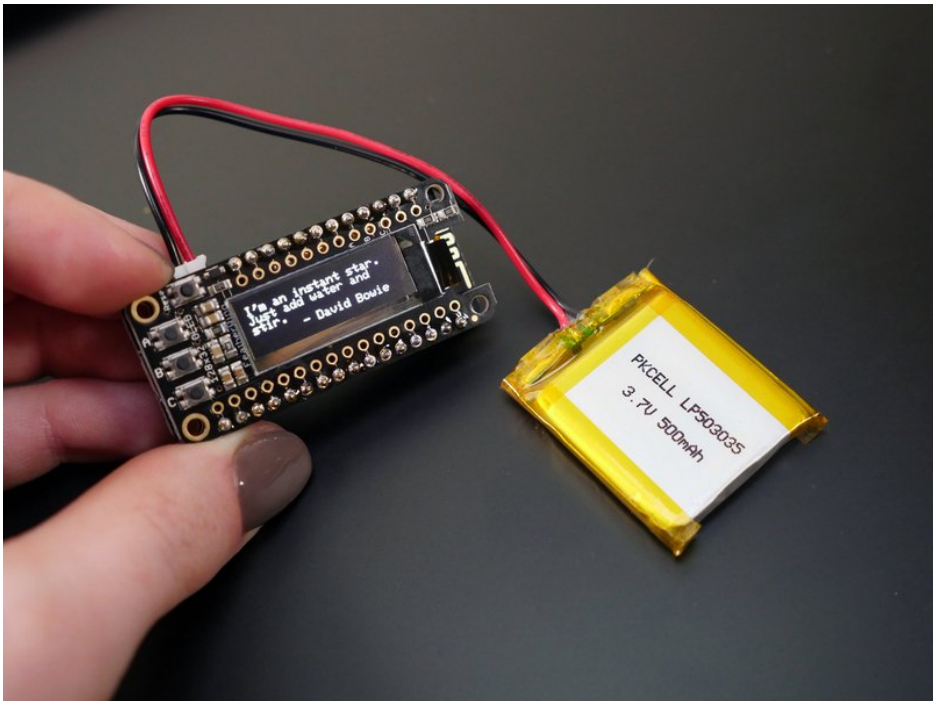
Create an internet connected display brooch and wear your data on your sleeve (or jacket, or backpack...)! This simple Adafruit IO project pulls forecast data from IFTTT and displays it on a FeatherWing OLED display, made wearable with a magnetic pin back. It also toggles to display a favorite quote, or serve as a nametag!

For this project you will need:

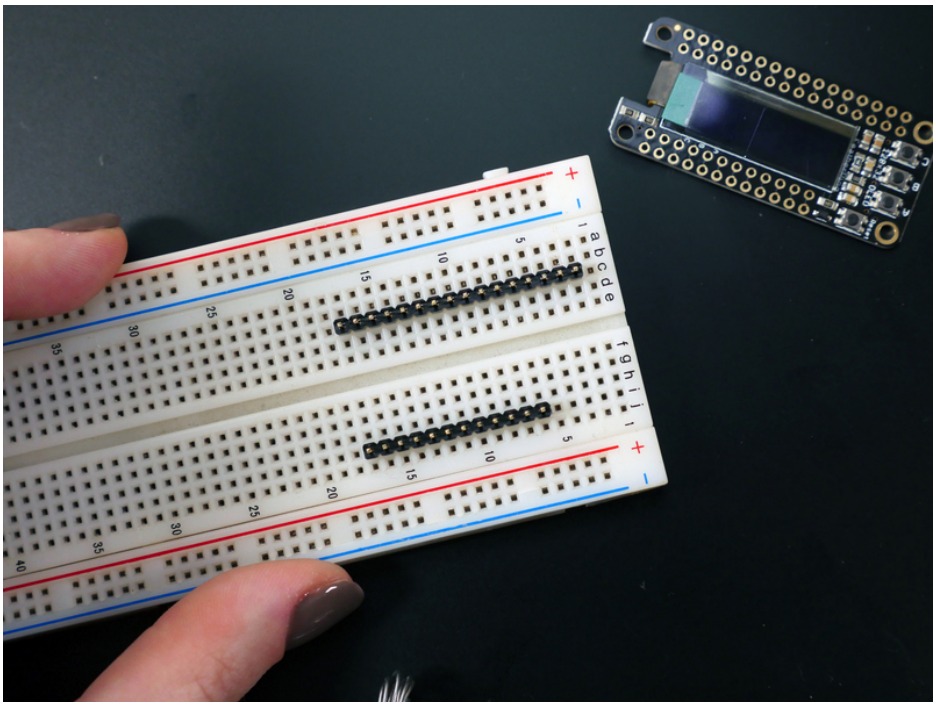
- Adafruit Feather HUZZAH ESP8266 WiFi board (http://adafru.it/2821)
- FeatherWing OLED display (http://adafru.it/2900)
- Magnetic pin back (http://adafru.it/1170)
- Lipoly battery (350mAh (http://adafru.it/2750) or 500mAh (http://adafru.it/1578) recommended)
- Soldering tools and supplies (https://adafru.it/drl)
- Accounts on Adafruit IO (https://adafru.it/fsU) and IFTTT (https://adafru.it/iOe)

Before you begin, please review and understand the following prerequisite guides:

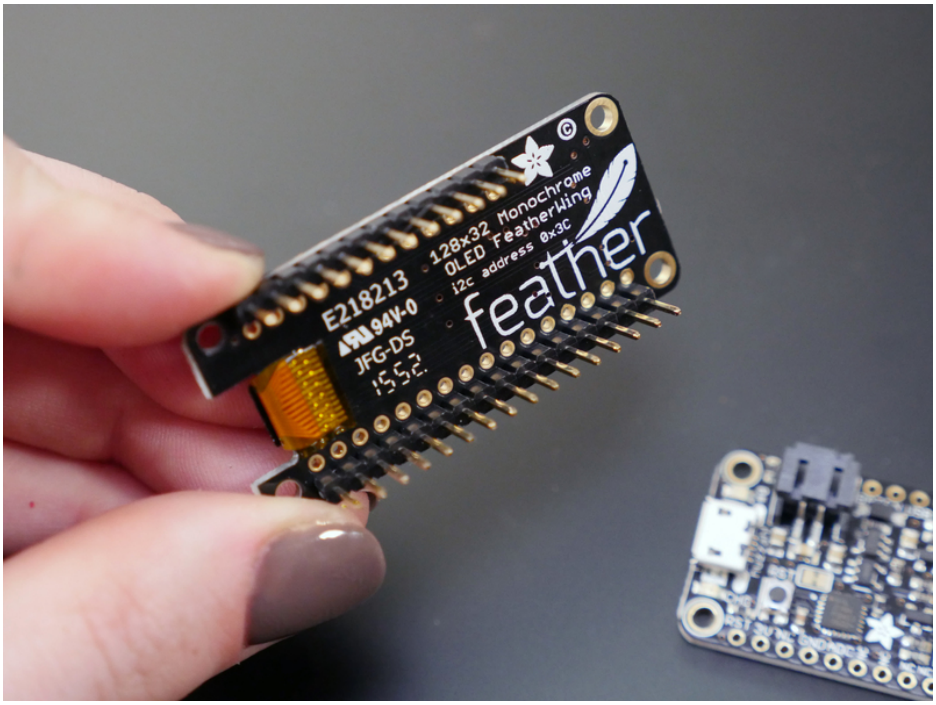
- Adafruit Feather HUZZAH ESP8266 (https://adafru.it/kD3)
- Monochrome OLED breakouts (https://adafru.it/dAZ)
- Adafruit IO (https://adafru.it/kSb)
- MQTT, Adafruit.IO & you! (https://adafru.it/Cgj)
- Adafruit guide to excellent soldering (https://adafru.it/drl)



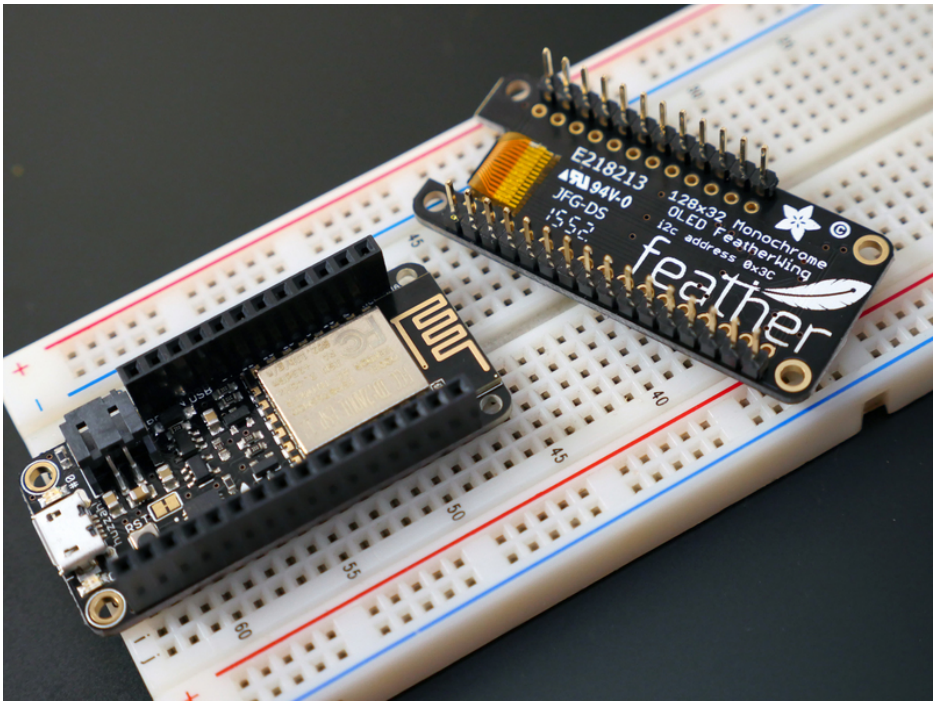
## Assemble Circuit

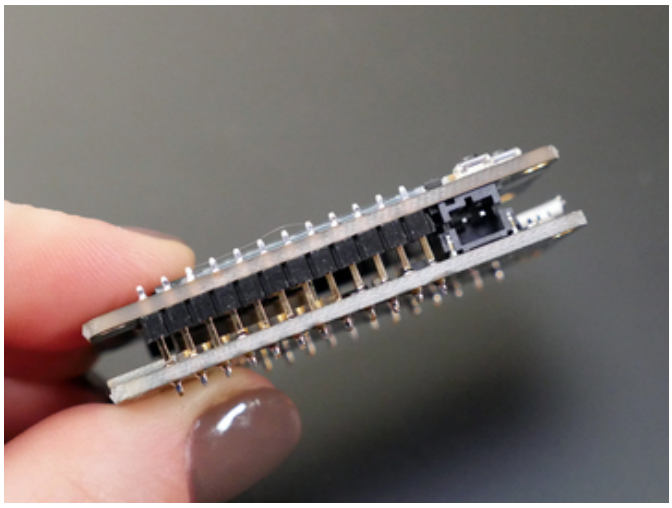


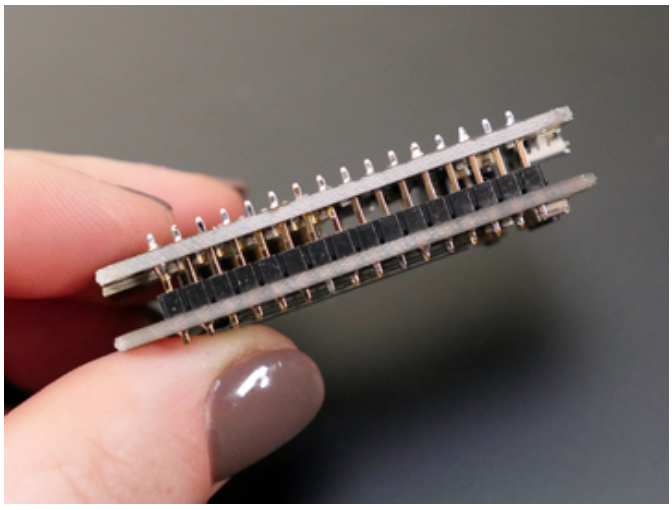
First up, solder headers onto the FeatherWing OLED. The easiest way to get the headers installed with proper alignment is to do this in a solderless breadboard. Trim one of the sections to fit the shorter row of holes on the FeatherWing and press into the breadboard. Slide the FeatherWing OLED onto the headers and solder in place.




If you have an extra feather to spare, setting one up on a breadboard with stacking headers (http://adafru.it/2830) is a great way to prototype with FeatherWings. Though not strictly necessary, this lets you try out different FeatherWings and assess any faulty solder connections without committing to permanently connecting the two.

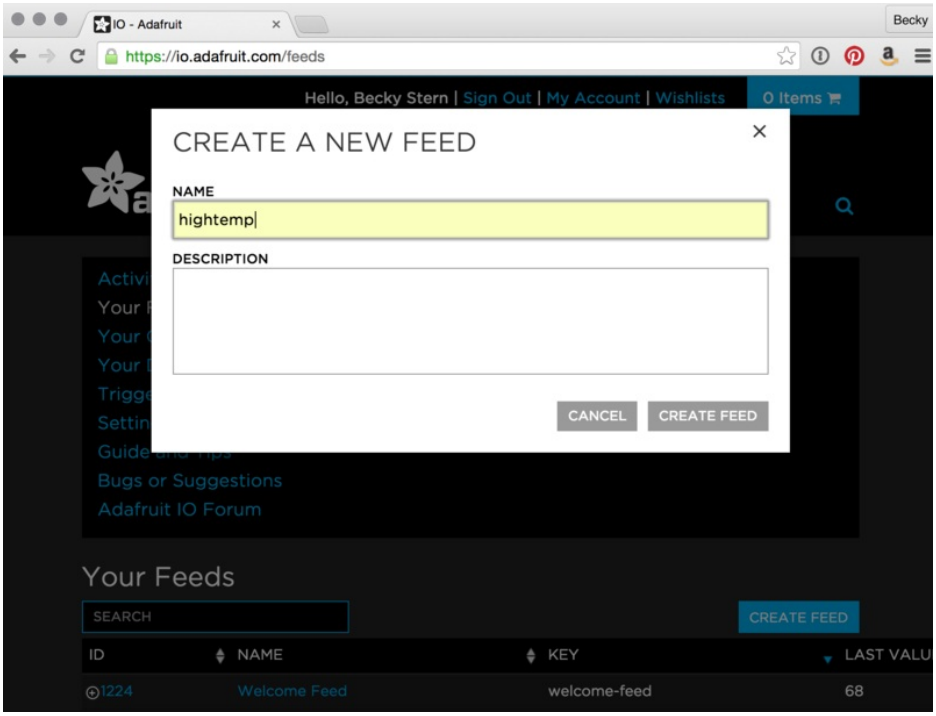


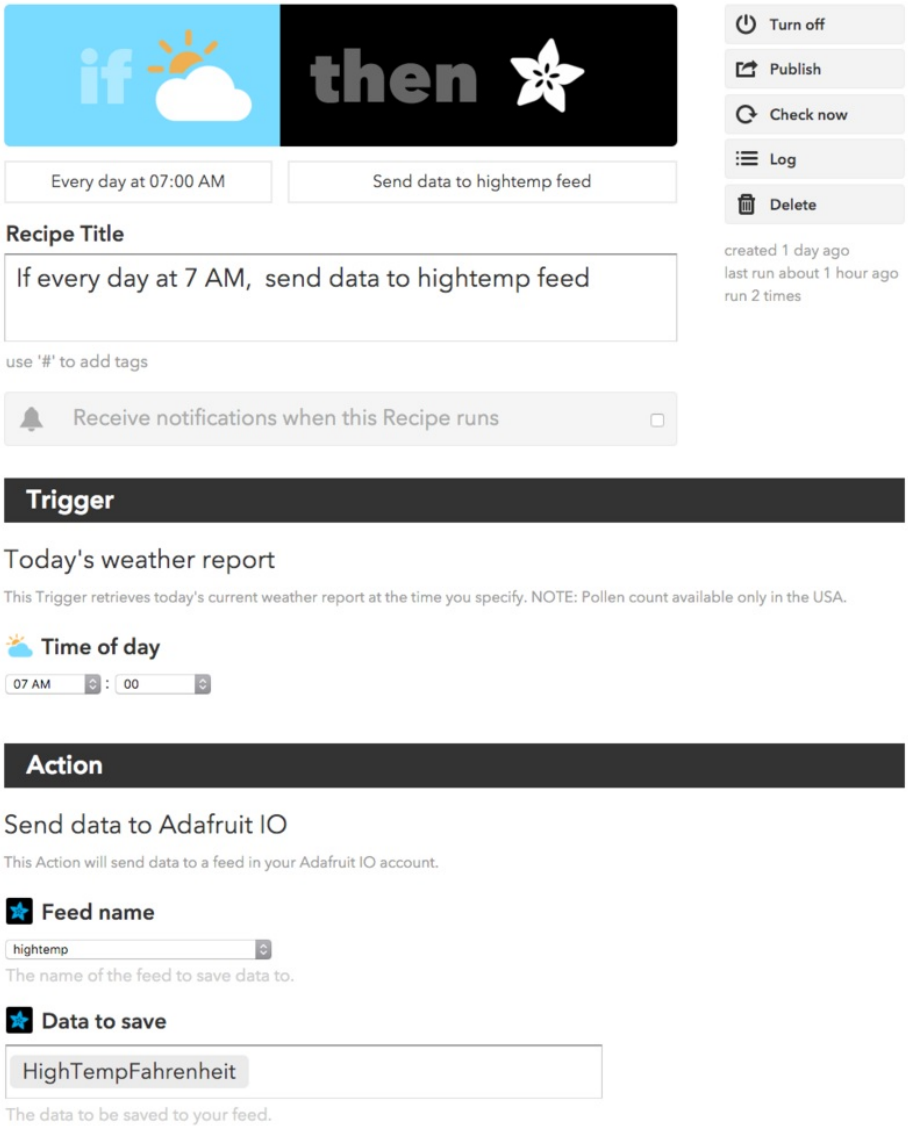


Next, solder your FeatherWing OLED to your Feather main board-- but don't let it be crooked! It's easiest to solder the pin closest to the JST battery connector first, then level the board as that solder cools. Then the rest of the pins will be oriented nicely for soldering up a perfectly parallel board sandwich.

# Adafruit IO and IFTTT Setup




First, create a feed to store your data on Adafruit IO called "hightemp"-- this will catch the value from IFTTT (https://adafru.it/iOe) and your Feather will check this feed for new values.




Next, create a recipe on IFTTT using the weather channel, and set it to trigger at the time of your choice. Select the Adafruit channel as the output and configure it to send today's forecasted high temperature value to the "hightemp" feed you already created.

You can also adopt the pre-made recipe we published on IFTTT! (https://adafru.it/kSd)

Now each day the recipe will update the feed with today's projected high temperature! Substitute this recipe with any value, weather related or not, to customize the project.

You can manually add a value to the feed to test the project, since the IFTTT recipe will only update the feed once per day.

## Code

This guide assumes you've completed the setup required to get your ESP8266 up and running with Arduino IDE and Adafruit IO.

- If you haven't yet set up your ESP8266 for use with Adafruit IO and the Arduino IDE, follow along with this guide (https://adafru.it/DCl). The setup only needs to be performed once.

The following Arduino libraries are required (install manually or through the library manager under Sketch-> Include Library-> Manage Libraries...):

- Adafruit SSD1306 (https://adafru.it/aHq)
- Adafruit GFX (https://adafru.it/aJa)
- Adafruit BusIO (https://adafru.it/GxD)

Modify the variables near the top of the sketch to match your WiFi network's SSID, password, Adafruit IO username, and Adafruit IO key before programming your Feather with the sketch.

```
/*
 *  WiFi OLED Display Badge
 *  guide: https://learn.adafruit.com/digital-display-badge/
 *  based on Home Automation sketch by M. Schwartz
 *  with contributions from Becky Stern and Tony Dicola
 */

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

const char* ssid     = "SSID";
const char* password = "password";

// Adafruit IO
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "your IO username"
#define AIO_KEY         "your IO key"

// Functions
void connect();

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

// Store the MQTT server, client ID, username, and password in flash memory.
// This is required for using the Adafruit MQTT library.
const char MQTT_SERVER[] PROGMEM    = AIO_SERVER;
// Set a unique MQTT client ID using the AIO key + the date and time the sketch
// was compiled (so this should be unique across multiple devices for a user,
// alternatively you can manually set this to a GUID or other random value).
const char MQTT_CLIENTID[] PROGMEM  = AIO_KEY __DATE__ __TIME__;
const char MQTT_USERNAME[] PROGMEM  = AIO_USERNAME;
const char MQTT_PASSWORD[] PROGMEM  = AIO_KEY;

// Setup the MQTT client class by passing in the WiFi client and
// MQTT server and login details.
```

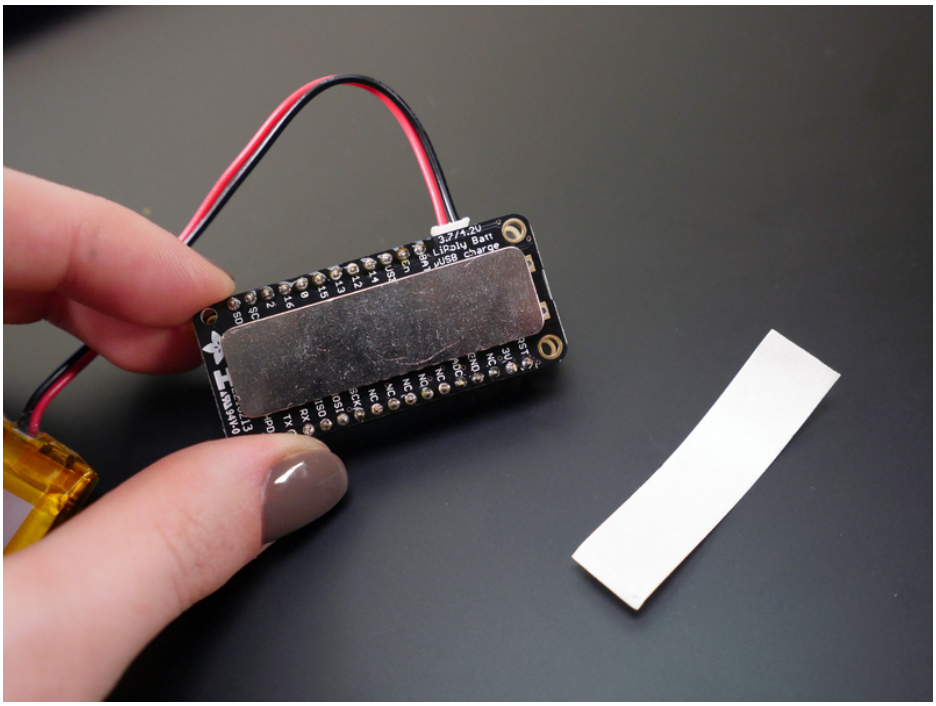
```
// MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, AIO_SERVERPORT,
  MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD);

/****************************** Feeds ***************************************/

// Setup a feed called 'hightemp' for subscribing to changes.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
const char hightemp_FEED[] PROGMEM = AIO_USERNAME "/feeds/hightemp";
Adafruit_MQTT_Subscribe hightemp = Adafruit_MQTT_Subscribe(&mqtt, hightemp_FEED);

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 3
Adafruit_SSD1306 display(128, 32, &Wire, OLED_RESET);

// this constant won't change:
const int buttonPin = 2;     // the pin that the pushbutton is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

void setup() {
  Serial.begin(115200);
  delay(100);

  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT_PULLUP);

  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Address 0x3C for 128x32

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds

  // Clear the buffer.
  display.clearDisplay();

  // We start by connecting to a WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.print("Connected to ");
  display.println(ssid);
  display.setCursor(0,16);
  display.print("IP address: ");
  display.println(WiFi.localIP());
  display.display();
  delay(2000);

  // listen for events on the weather feed
  mqtt.subscribe(&hightemp);

  // connect to adafruit io
  connect();
}

int temp = 151;
int whichbutton = 0;

void loop() {
  Adafruit_MQTT_Subscribe *subscription;

  // ping adafruit io a few times to make sure we remain connected
  if(! mqtt.ping(3)) {
    // reconnect to adafruit io
    if(! mqtt.connected())
      connect();
  }

  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);

  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == LOW) {
      // if the current state is LOW then the button was pressed
      buttonPushCounter++;
      Serial.print("number of button pushes:  ");
      Serial.println(buttonPushCounter);
    }
  }

  // this is our 'wait for incoming subscription packets' busy subloop
  if (subscription = mqtt.readSubscription(1000)) {
    // we only care about the weather events
    if (subscription == &hightemp) {
      temp = atoi((char*)hightemp.lastread);
      Serial.print(F("Received: "));
      Serial.println(temp);
    }
  }

  //A button position - display today's high temperature
  if (buttonPushCounter % 3 == 0) {
```

```
  if (buttonPushCounter % 3 == 0) {
    if(temp == 151) {
      display.clearDisplay();
      display.setCursor(0,16);
      display.println("Waiting for temp data");
      display.display();
      Serial.println(F("Printed: Waiting for temp data"));
    } else {
      display.clearDisplay();
      display.setCursor(0,16);
      display.print("Today's high: ");
      display.print(temp);
      display.println(" F");
      display.display();
      Serial.print(F("Printed: "));
      Serial.println(temp);
    }
  }

  //B button position - David Bowie quote
  if (buttonPushCounter % 3 == 1) {
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("I'm an instant star. Just add water and ");
    display.println("stir.");
    display.println("       - David Bowie");
    display.display();
  }

  //C button position - nametag
  if (buttonPushCounter % 3 == 2) {
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("Your Name");
    display.println();
    display.println("Your Title");
    display.display();
  }

  // save the current state as the last state,
  //for next time through the loop
  lastButtonState = buttonState;
}

// connect to adafruit io via MQTT
void connect() {

  Serial.print(F("Connecting to Adafruit IO... "));

  int8_t ret;

  while ((ret = mqtt.connect()) != 0) {

    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
      case 5: Serial.println(F("Not authed")); break;
      case 6: Serial.println(F("Failed to subscribe")); break;
```

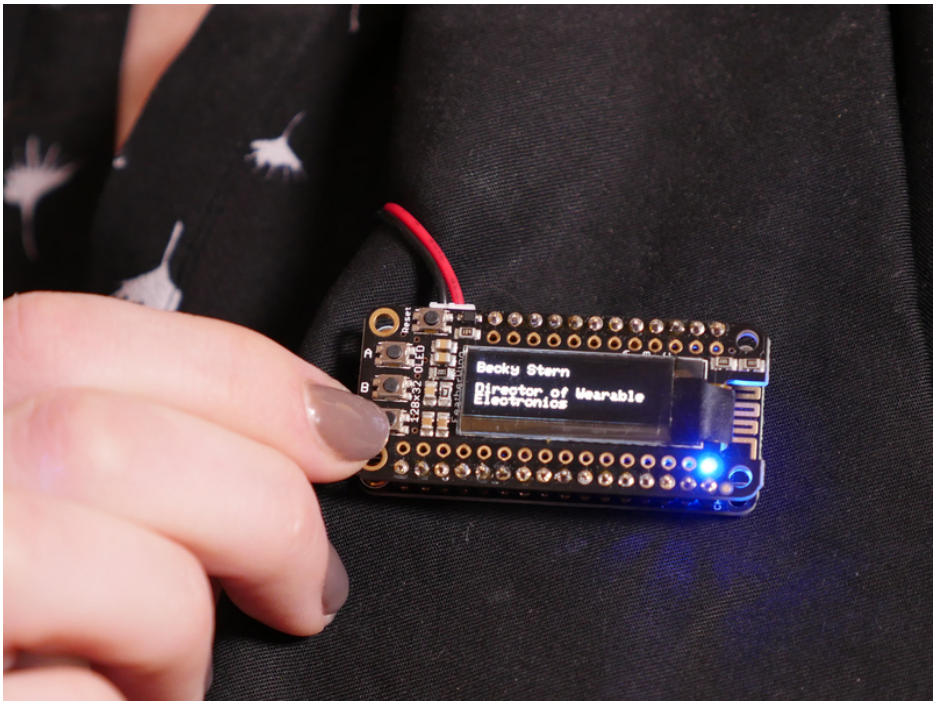
```
      default: Serial.println(F("Connection failed")); break;
    }

    if(ret >= 0)
      mqtt.disconnect();

    Serial.println(F("Retrying connection..."));
    delay(5000);
  }

  Serial.println(F("Adafruit IO Connected!"));
}
```

## Wear it!



When your circuit is working properly, attach the magnetic pin back by peeling the paper from the adhesive backing on the metal plate. Stick it to the back of the Feather, and use the magnetic plate to attach it to your jacket, backpack, and more!

The C button on the FeatherWing OLED serves to toggle between display modes. If you wish, you may tuck your battery inside a 3D printed pocket (https://adafru.it/kSa)!



# OCEAN CHIPS
## Океан Электроники
## Поставка электронных компонентов

Компания «Океан Электроники» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Поставка оригинальных импортных электронных компонентов напрямую с производств Америки, Европы и Азии, а так же с крупнейших складов мира;
- Широкая линейка поставок активных и пассивных импортных электронных компонентов (более 30 млн. наименований);
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Помощь Конструкторского Отдела и консультации квалифицированных инженеров;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Поставка электронных компонентов под контролем ВП;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- При необходимости вся продукция военного и аэрокосмического назначения проходит испытания и сертификацию в лаборатории (по согласованию с заказчиком);
- Поставка специализированных компонентов военного и аэрокосмического уровня качества (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Actel, Aeroflex, Peregrine, VPT, Syfer, Eurofarad, Texas Instruments, MS Kennedy, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics и др.);

Компания «Океан Электроники» является официальным дистрибьютором и эксклюзивным представителем в России одного из крупнейших производителей разъемов военного и аэрокосмического назначения «**JONHON**», а так же официальным дистрибьютором и эксклюзивным представителем в России производителя высокотехнологичных и надежных решений для передачи СВЧ сигналов «**FORSTAR**».

AVIC JONHON

**«JONHON»** (основан в 1970 г.)

Разъемы специального, военного и аэрокосмического назначения:

(Применяются в военной, авиационной, аэрокосмической, морской, железнодорожной, горно- и нефтедобывающей отраслях промышленности)

**«FORSTAR»** (основан в 1998 г.)

ВЧ соединители, коаксиальные кабели, кабельные сборки и микроволновые компоненты:

(Применяются в телекоммуникациях гражданского и специального назначения, в средствах связи, РЛС, а так же военной, авиационной и аэрокосмической отраслях промышленности).

FORSTAR®

Телефон: 8 (812) 309-75-97 (многоканальный)
Факс: 8 (812) 320-03-32
Электронная почта: ocean@oceanchips.ru
Web: http://oceanchips.ru/
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, д. 2, корп. 4, лит. А