Adafruit IO Time Tracking Cube

Created by Brent Rubell

https://learn.adafruit.com/time-tracking-cube

Last updated on 2024-06-03 02:30:16 PM EDT

Table of Contents

Overview

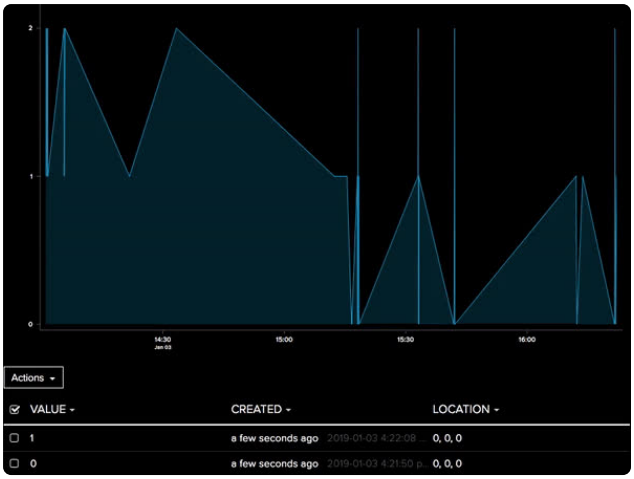
Time Tracking Tasks

It's difficult to remember exactly what time you started work on a project, and when you stopped. Tasks like taking out the dog and brewing a fresh pot of coffee to keep you focused do not count towards your project's "total hours".

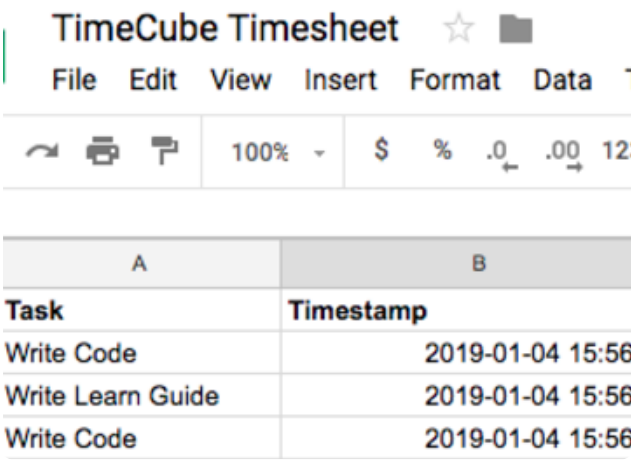
Automate Time-Sheets

You can use Microsoft Excel to track your data using a time-sheet, but manually inputting times and tasks also becomes a task. We're going to build a different way to track your - using a flippable cube.

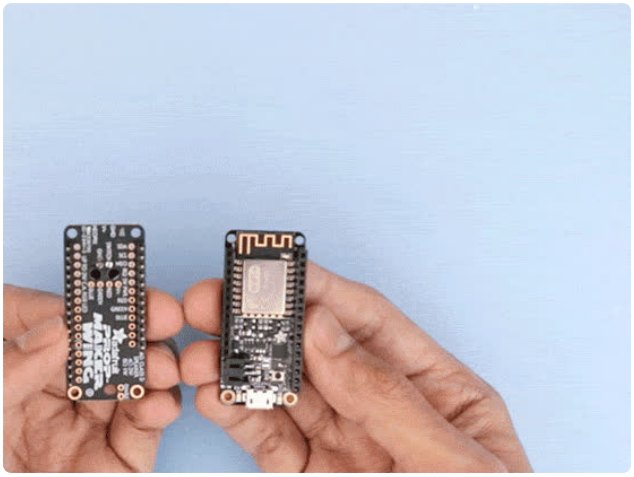
New task? Flip it over.Taking a break? Flip it over. Finished with work for the day? Flip it it over!



We'll be connecting the Time Tracking Cube to Adafruit IO (https://adafru.it/fH9) to collect **metadata** about when the cube was flipped and its orientation.



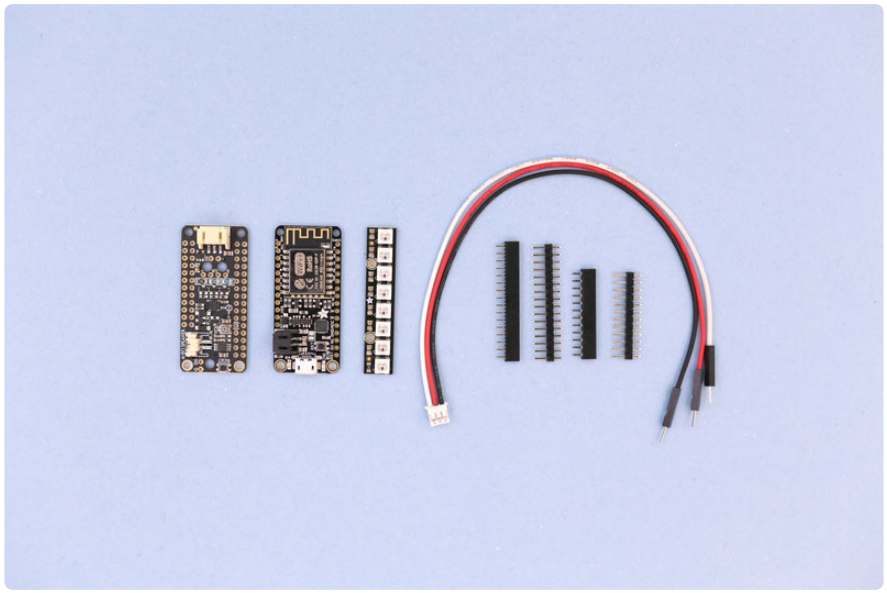
We're also going to create a Zapier (https://adafru.it/fVP) Zap to periodically collect data from this feed and send it to a timesheet we create on Google Sheets.



Adafruit Feather Platform

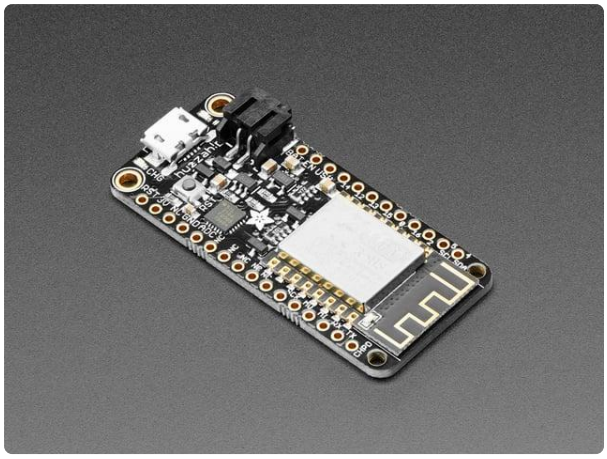
The Adafruit Feather HUZZAH with ESP8266 is an all in one WiFi dev board with built in USB and battery charging. The Adafruit Feather format is easy to use gives you tons of add-ons. The Prop-Maker Wing adds NeoPixel support and sound effects.

Parts



You'll need the following parts to complete this guide.

The ESP8266 is simple to set up and supported by the Adafruit IO Arduino library.

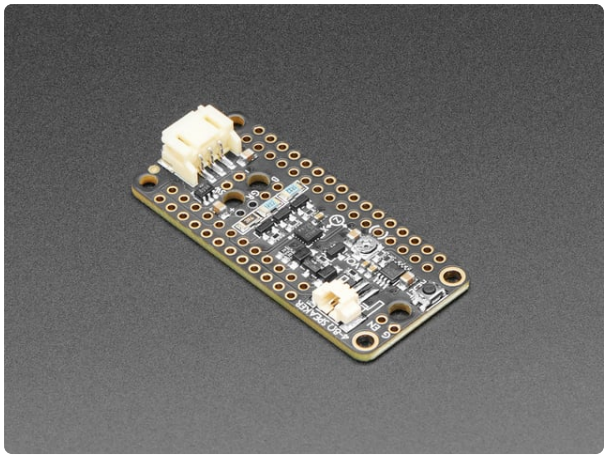


Adafruit Feather HUZZAH with ESP8266 - Loose Headers

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...

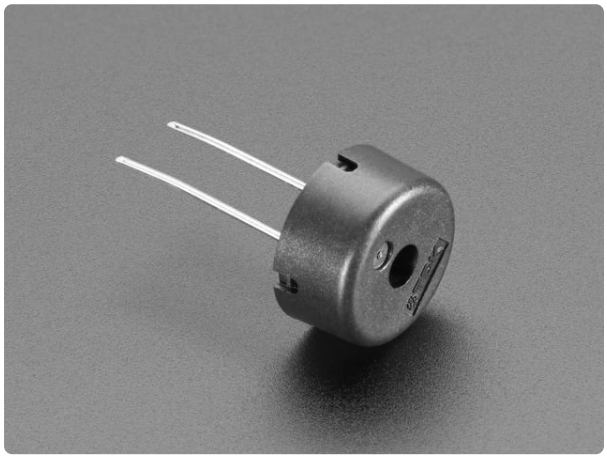
https://www.adafruit.com/product/2821

The Prop-Maker FeatherWing has **lots** of options for building either props, or desk-toys. We'll use one feature of the FeatherWing - the LIS3DH Accelerometer. This sensor is perfect for detecting when the cube is tilted in different orientations on the desk.



Adafruit Prop-Maker FeatherWing
The Adafruit Feather series gives you lots of options for a small, portable, rechargeable microcontroller board. Perfect for fitting into your next prop build! This FeatherWing will...
https://www.adafruit.com/product/3988

We'll also use a small Piezo Buzzer to let us know if the cube was tilted, without checking Adafruit IO or the Serial Monitor.



Piezo Buzzer
Piezo buzzers are used for making beeps, tones and alerts. This one is petite but loud! Drive it with 3-30V peak-to-peak square wave. To use, connect one pin to ground (either one) and...
https://www.adafruit.com/product/160

1 x 8x NeoPixel Stick — https://www.adafruit.com/product/1426
5050 RGB LED NeoPixels

1 x Feather Female Headers Kit — https://www.adafruit.com/product/2940
12-pin and 16-pin short female headers

1 x Feather Male Headers Kit — https://www.adafruit.com/product/3002
12-pin and 16-pin male headers

1 x 3-Pin JST PH Cable — https://www.adafruit.com/product/3894
3-pin JST-PH cable

Materials

Tools

To build your Time Tracking Cube, you'll need access to the following tools. Don't have something listed? Pick it up from the Adafruit shop.

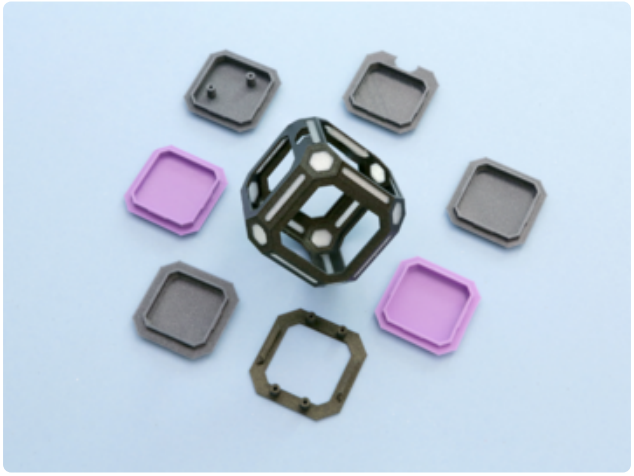
1 x 3D Printer
Ultimaker 2+ 3D Printer
https://www.adafruit.com/product/2673

---

6 x M2.5 x 8mm metric machine screws
M2.5 x 8mm flat head metric screws
https://www.albanycountyfasteners.com/2-5-MM-x-45-Phillips-Flat-Head-Machine-Screw-p/1011-1002.htm

---

3D Printing



3D Printed Parts

Parts are designed to be 3D printed with FDM based machines. STL files are oriented to print "as is". Machines with dual extrusion or single extrusion setups are listed below with parts name and description. Parts require tight tolerances that might need adjusting slice setting. Reference the suggested settings below.

Parts List

Use the parts list to reference filenames and extruder versions.

- **ttc-box.stl** – cube frame (single extrusion)
- **ttc-box-A.stl** – cube frame (dual extrusion)
- **ttc-box-B.stl** – clear panels (dual extrusion)
- **ttc-side.stl** – 3x quantity symmetrical side panels
- **ttc-top-side.stl** – Install NeoPixel to this panel
- **ttc-bot-side.stl** – Install to cover to this panel
- **ttc-back-side.stl** – Panel with USB port

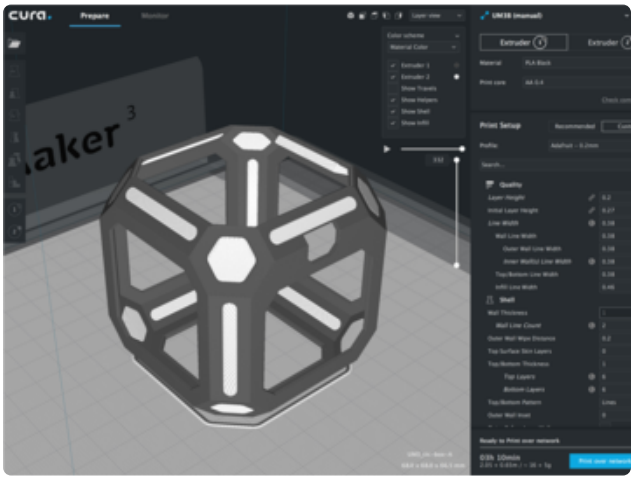
- **ttc-cover.stl –** Install Feather HUZZAH to this panel

Tap Mounting Holes

I used a set of screw taps to create threads in the various mounting holes. This greatly improves fastening screws. of the A mix of metric sizes are used here. M2.5 taps for the various mounts.

Parts Assembly

The Feather is secured to the bottom cover using M2.5 x 8mm flat head machine screws. The bottom cover snap fits onto the bottom of the cube. Several panels are snap fitted into the openings.



CURA Slicing Software

Rotate to orient the cube so the microUSB opening facing the top – This orientation is best for 3D printing support free.

Tool heads equipped for dual extrusion can be setup using the **ttc-box-A.stl** and **ttc-box-B.stl** named parts. Use a transparent filament for part B and a darker color for part A.

Download CAD Parts

https://adafru.it/DCz

Design Source Files

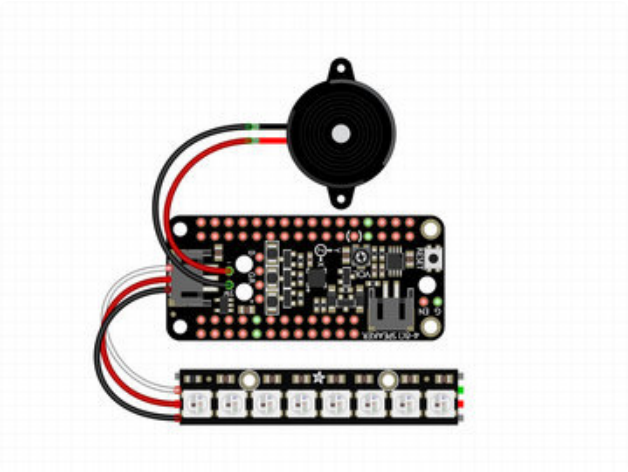
The enclosure assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, SAT and more. Electronic components like the board,

displays, connectors and more can be downloaded from our Fusion 360 CAD parts github repo (https://adafru.it/AW8).

Download Adafruit CAD Parts

https://adafru.it/AW8

Wiring



Circuit Diagram

This provides a visual reference for wiring of the components. They aren't true to scale, just meant to be used as reference. This diagrams was created using Fritzing software (https://adafru.it/oEP).

The FeatherWing performs all of the heavy lifting. But we are going to **solder a low-cost piezo buzzer** to the Prop-Maker FeatherWing.

Connect one end of the a piezo buzzer (it doesn't matter which one) to the **SW** pin and the other end to the **GND** pin.

Three wires are needed for connecting the 8x NeoPixel stick to the Prop-Maker FeatherWing.

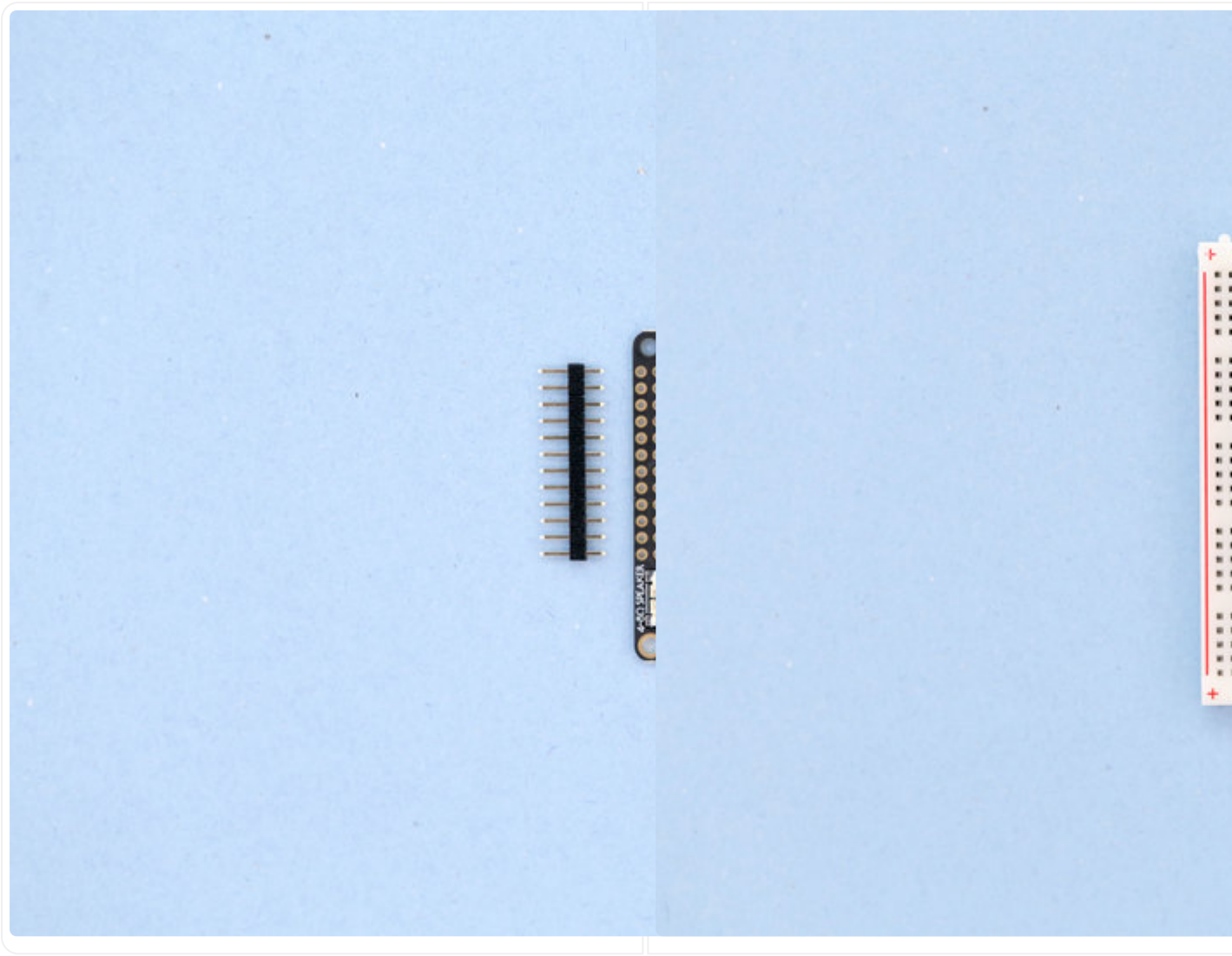
- **DIN** from NeoPixel to Prop-Maker NeoPixel port
- **GND** from NeoPixel to Prop-Maker NeoPixel port
- **5VDC** from NeoPixel to Propmaker NeoPixel port

Powering

The Adafruit Feather can be powered via USB. Use a micro USB cable and connect to a 5V USB battery pack or wall adapter.

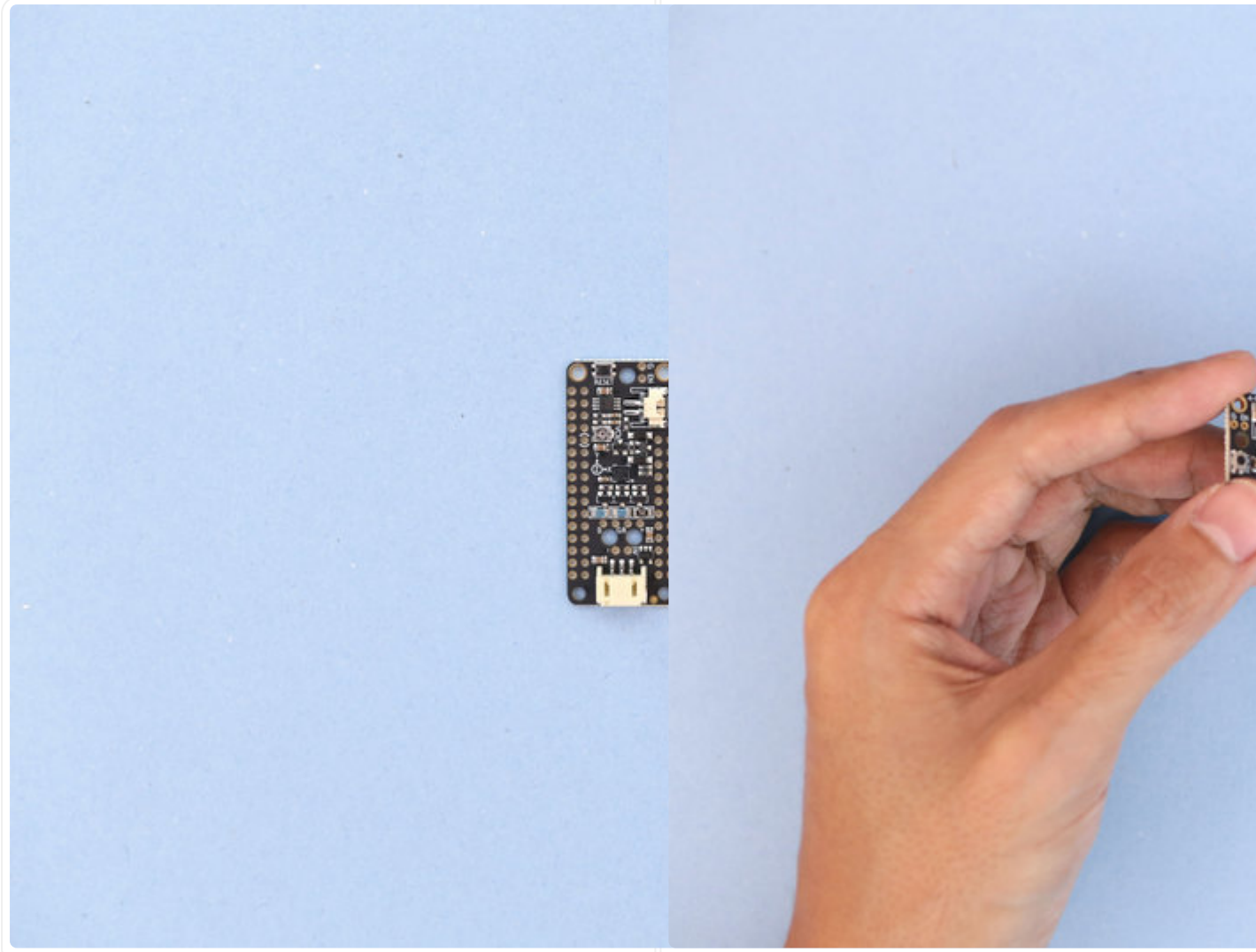
Install Prop-Maker Wing Headers

Start by fitting the 12 and 16 pin headers onto the bottom of the Prop-Maker FeatherWing PCB with the short ends going into the pins. Solder all of the pins. I suggest using a breadboard to help keep the pins in place while soldering.



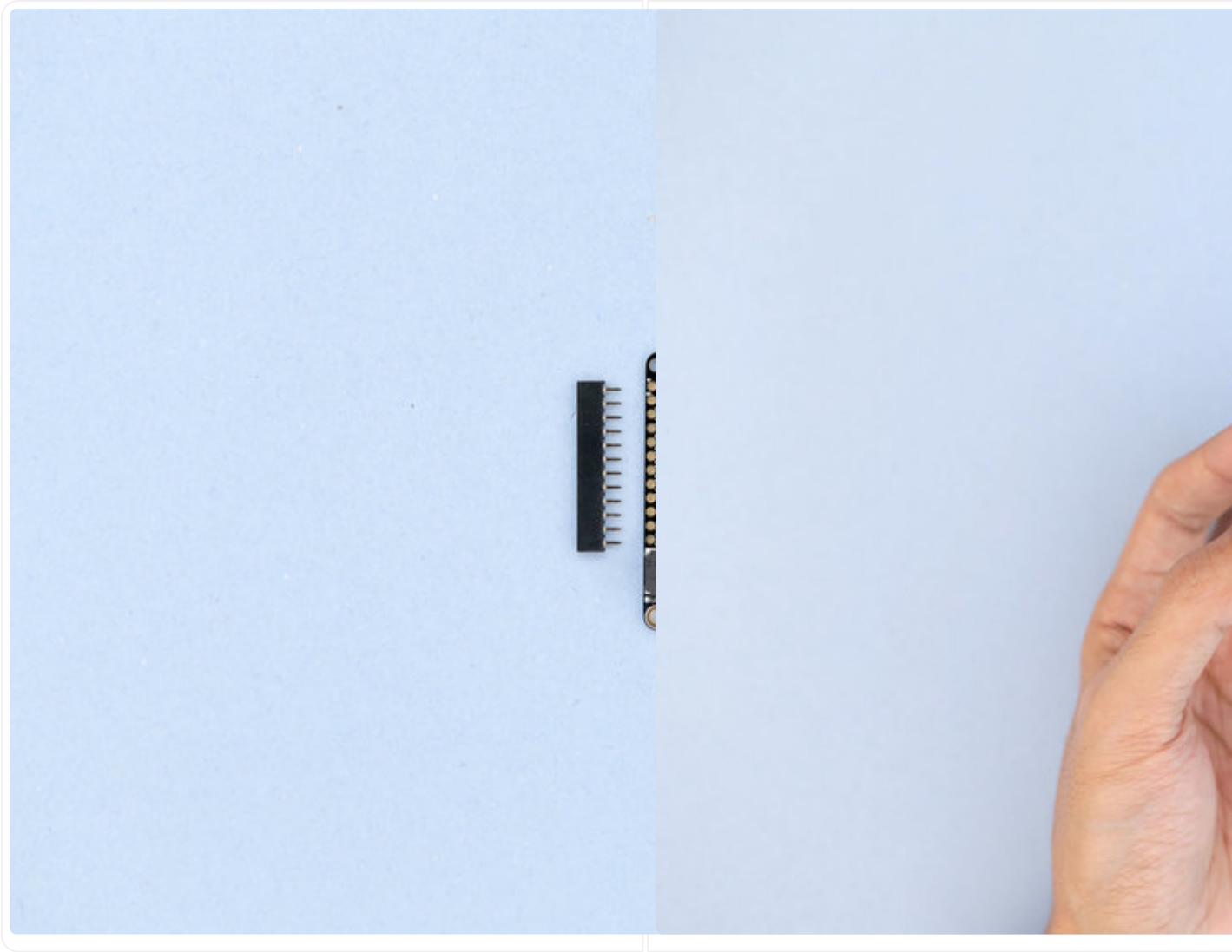
Install Piezo to Prop-Maker

Trim the legs of the piezo short. Fit the pins from the piezo onto the SWITCH and ground (GND) pins on the Prop-Maker FeatherWing. Flip the PCB over and solder the pins in place.



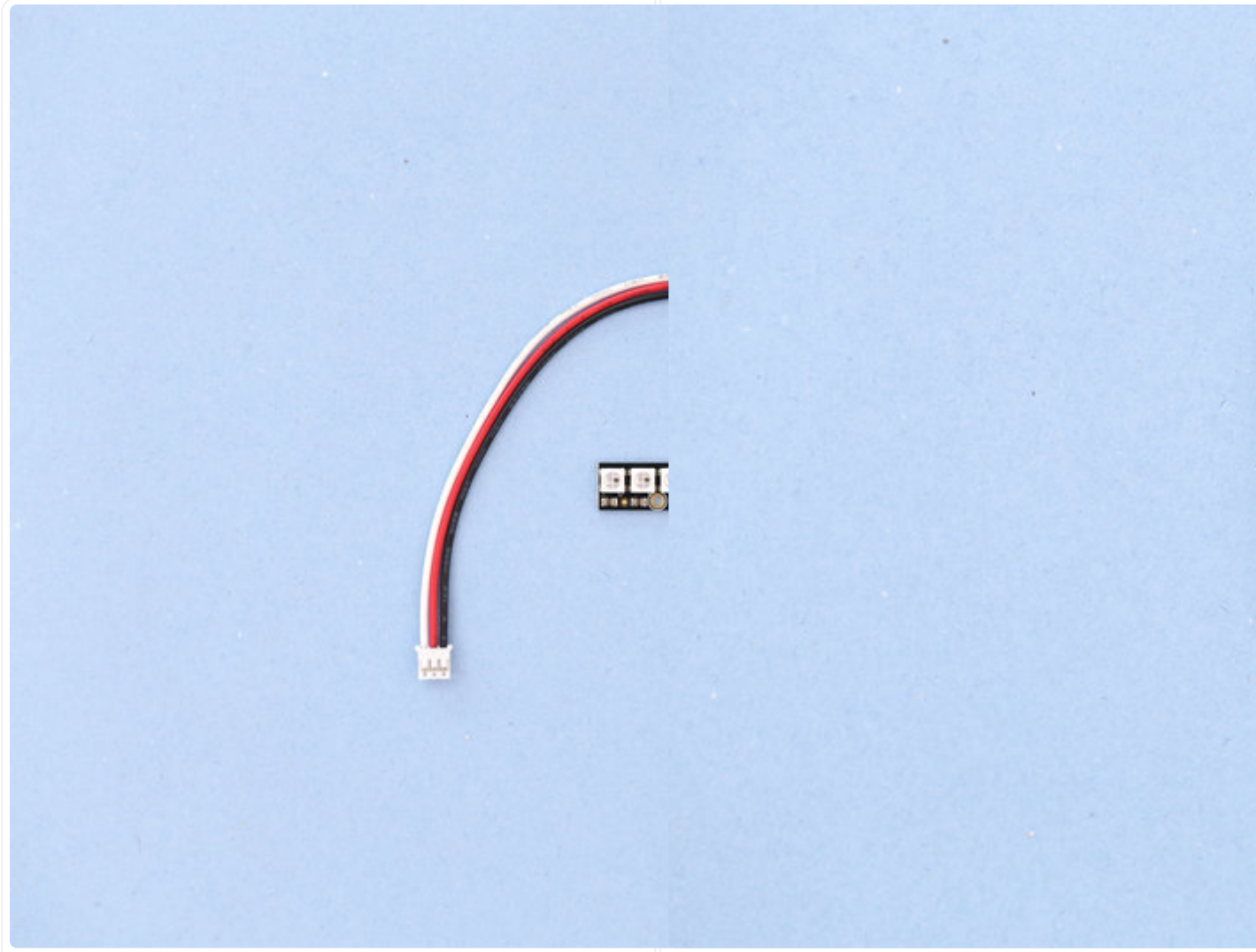
Install HUZZAH Headers

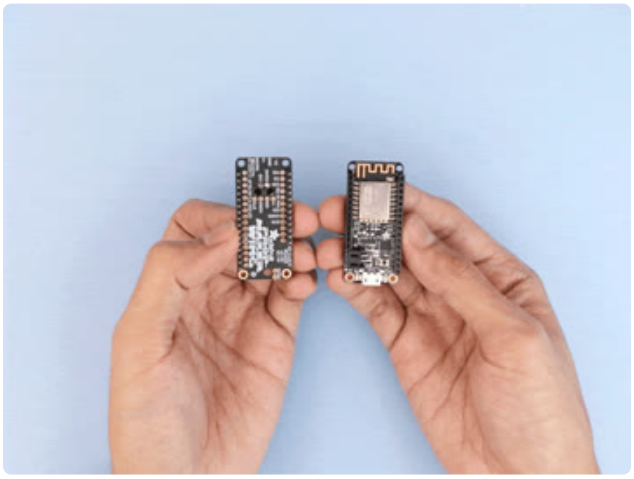
Use 12-pin and 16-pin female headers on the Adafruit HUZZAH ESP8266. These are installed on top of the PCB. To make soldering easier, I suggest installing the headers onto the Prop-Maker – This will keep the female headers in place while soldering. Solder the pins from the female headers.



NeoPixel Wiring

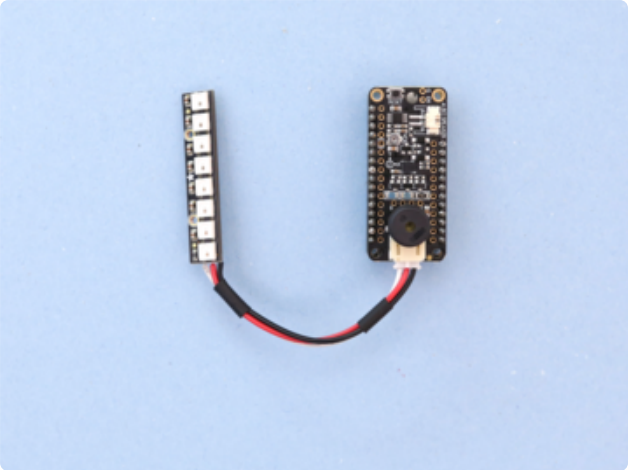
Cut the JST cable so the wire length is about 85mm (3.3in). Using wire stripper, remove a bit of insulation from the tips of each wire. Apply a bit of solder to the exposed strands of wire – This helps prevent the wires from fraying. Tin the **DIN**, **5+** and **GND** pads on the NeoPixel PCB with a bit of solder. Solder the wires from the JST cable to the pads on the NeoPixel PCB.





Snap-On FeatherWing

Line up the pins from the Feather HUZZAH ESP8266 with the Prop-Maker FeatherWing. Press fit the headers together so they're fully seated.



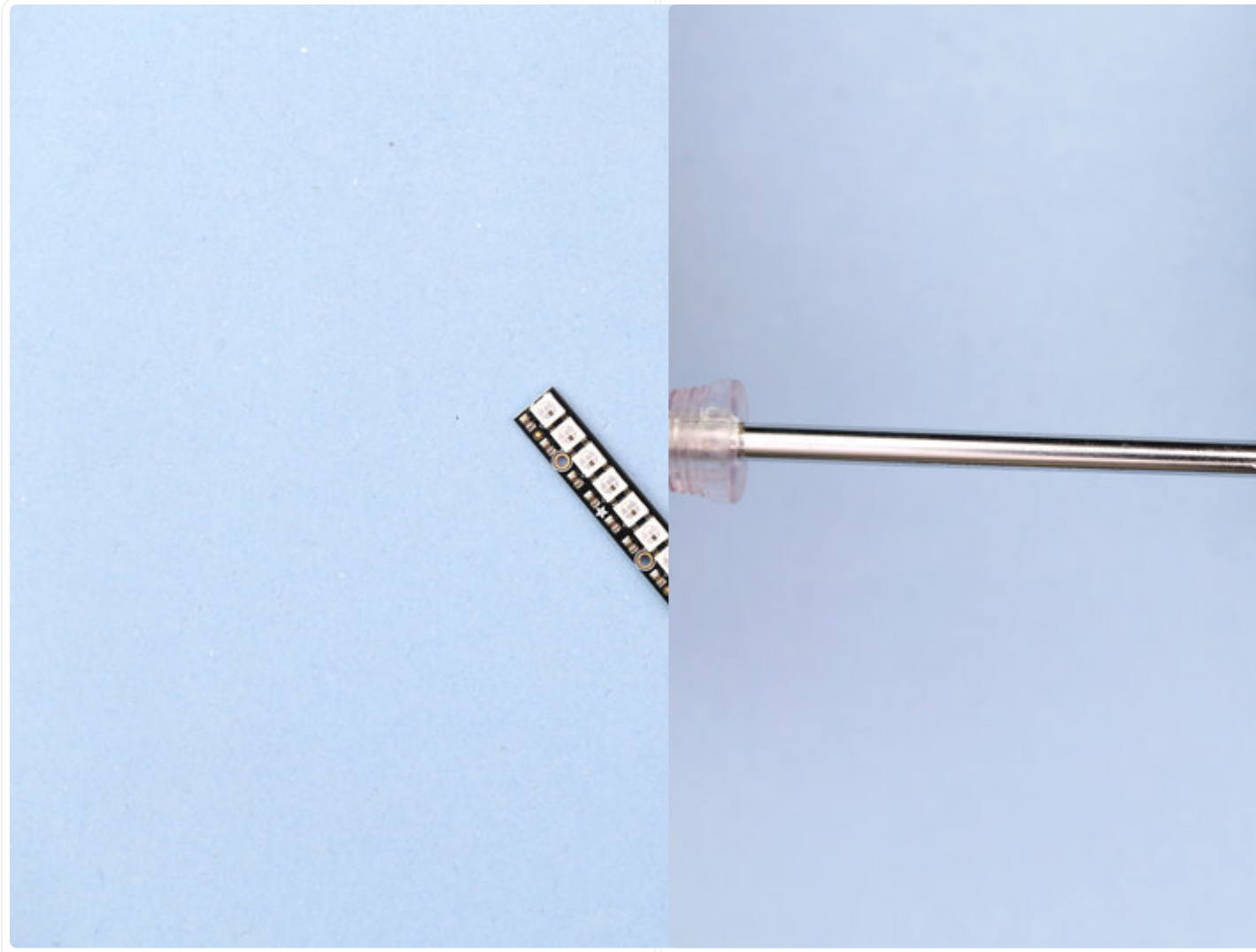
Test Circuit

Plug in the JST cable from the NeoPixel into the port on the Prop-Maker FeatherWing. Connect micro USB cable to your computer or powered hub to power the circuit.

Assembly

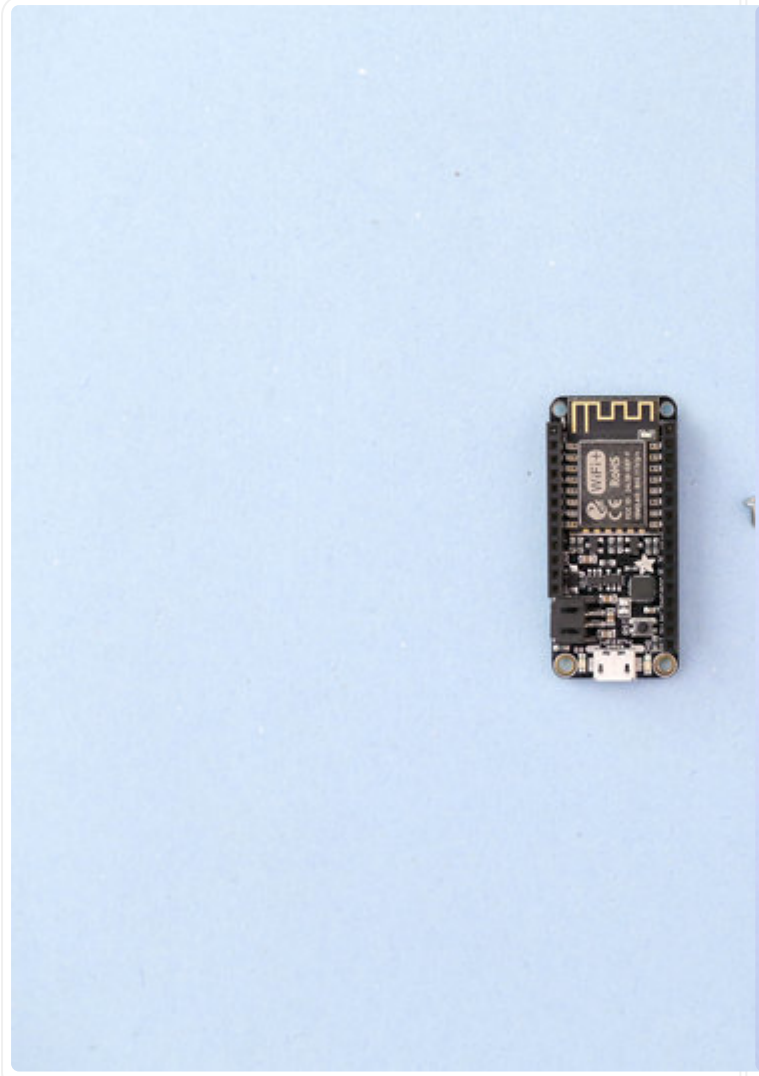
Install NeoPixel to Panel

Use two M2.25 x 8mm flat head machine screws to secure the NeoPixel stick PCB to the standoffs on the top panel. Tap the mounting holes on the PCB for a better fitting. Insert and fasten the screws into the mounting holes. Place the PCB over the standoffs with the holes lined up with the standoffs. Hold PCBin place while fastening the screws.



Install HUZZAH to Panel

Use four M2.5 x 8mm flat head machine screws to secure the Feather PCB to the standoffs on the bottom cover. Place the Feather PCB over the standoffs and line up the mounting holes. Insert and fasten the screws to secure the PCB to the bottom cover.



Install Panels to Cube

Get the cube and the panels ready! The side panels are printed in a different color to distinguish which side will trigger the timer.

Bottom Panel

Starting with the bottom cover panel, orient the piece so the micro USB port on the Feather is lined up with the square cut out on the cube. Insert the cover through the opening and click the edges into the nubs to secure it closed.

Top Panel

The panel with the NeoPixel stick is installed in the opening directly above the Feather HUZZAH PCB. Insert the lip of the panel into the opening at an angle and press the rest of the edges through until they click into place.

Connect JST

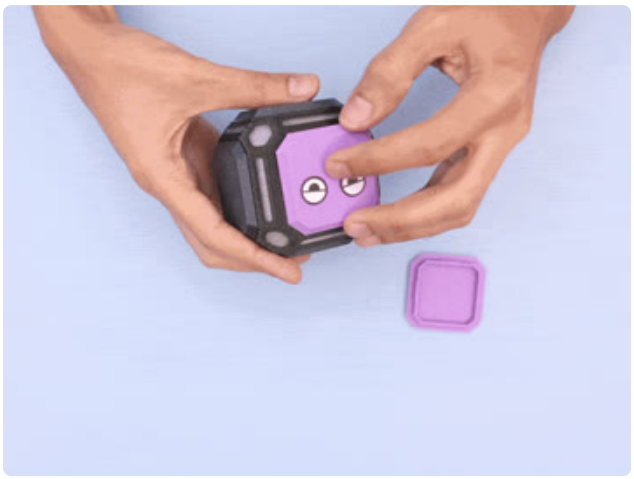
Grab the JST cable from the NeoPixel and plug it into the Prop-Maker FeatherWing.

Back Panel

Orient the back panel with the micro USB port and press fit the lip into the opening.

Front Panel

Repeat the same snap fitting procedure to the front facing panel. Use both hands to slightly flex the opening and firmly press fit edges through.



Task Panels

Now it's time to install the special task panels for time tracking. Use stickers, vinyl decals, or different colored parts to tell the tasks apart. Sticky note or label for something with more legibility.

Assembled Cube

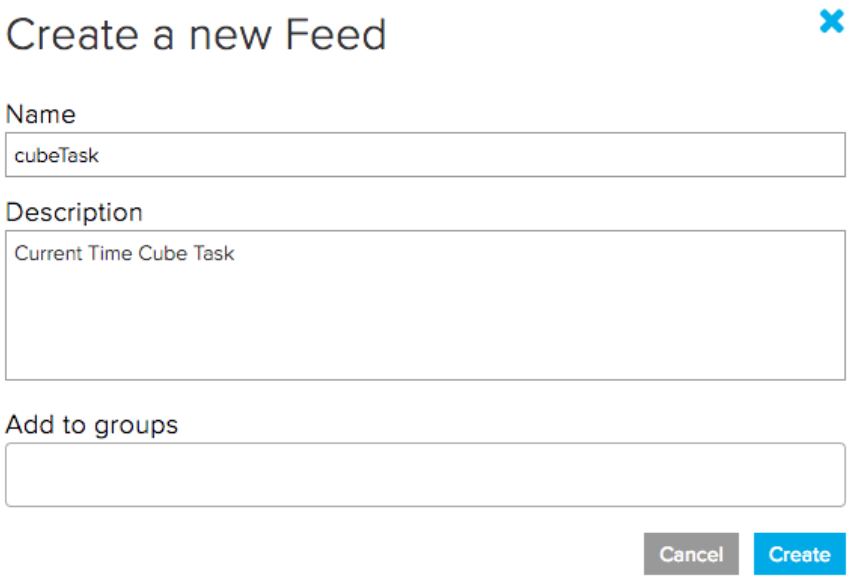
Take a moment to inspect the panels and make any necessary adjustments. If everything is as expected, congrats! The time tracking poly cube is fully assembled and ready to log your data!

Adafruit IO Setup

If you do not already have an Adafruit IO account set up, head over to io.adafruit.com (https://adafru.it/fH9) to link your Adafruit.com account to Adafruit IO.

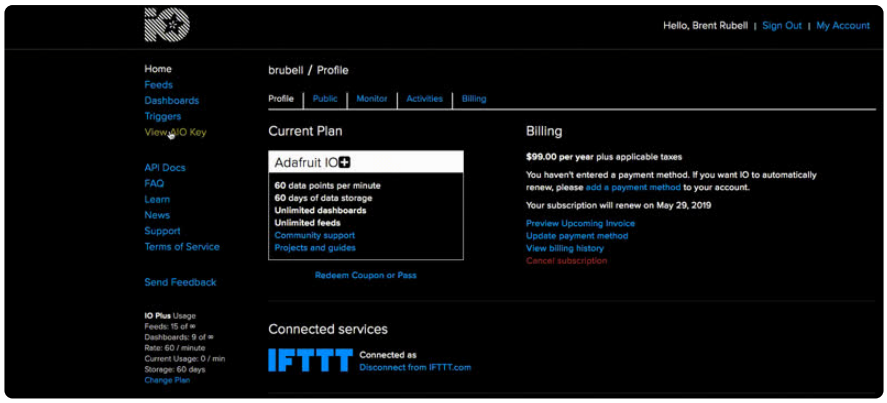
We need to create a new feed to hold data for the cube's current face (its orientation). Navigate to the feeds page (https://adafru.it/mxC) on Adafruit IO. Then click **Actions -> Create New Feed**, and name this feed **cubeorientation**.

- If you do not already know how to create a feed, head over to Adafruit IO Basics: Feeds (https://adafru.it/ioA).



We're also going to need our Adafruit IO username and our secret API key.

Navigate to your profile and click the **View AIO Key** button to retrieve them. Write them down in a safe place, we'll need them for later.



Now that Adafruit IO is set up, let's move on to setting up our Feather HUZZAH ESP8266 with the Prop-Maker Wing.

Zapier Setup

Using Zapier (https://adafru.it/fVP) with Adafruit IO allows you to automate tasks on the Internet with your data, like writing to Google Sheets or sending a Tweet.

Linking Adafruit IO and Zapier

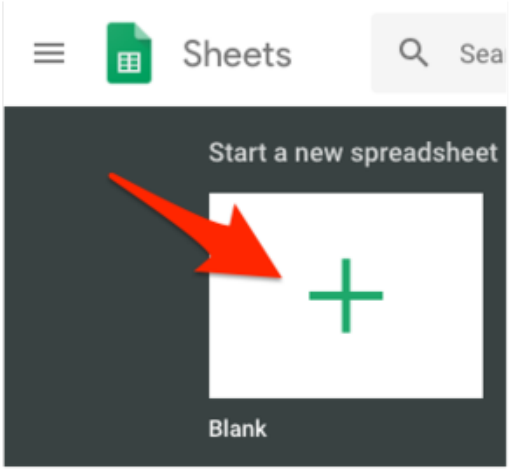
You'll want to first link Zapier with your Adafruit IO Account.

Zapier for Adafruit IO is currently not listed on the Zapier Integrations page (we need 10 active users to make it public), you can sign up for it using this invite link **(https://adafru.it/Dw6)**.

After signing up for Zapier and linking your Adafruit IO account, you'll need to create a Google Sheets (https://adafru.it/DCi) account if you haven't already.

Google Sheets Setup

We'll want to create a spreadsheet to hold all the data from the Time Tracking Cube.

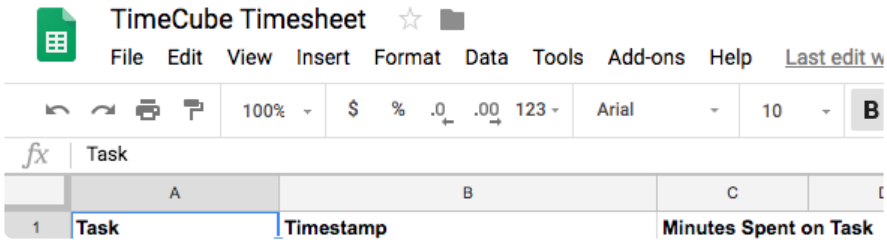


Navigate to the Google Sheets Homepage (https://adafru.it/DCj), **Click Start a New Spreadsheet**

On the Spreadsheet, **make three row headers**:

- **Task**
- **Timestamp**
- **Minutes Spent on Task**

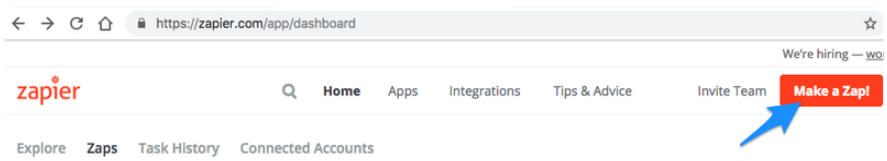
These three columns will hold the metadata from the Adafruit IO feed created in the previous step.

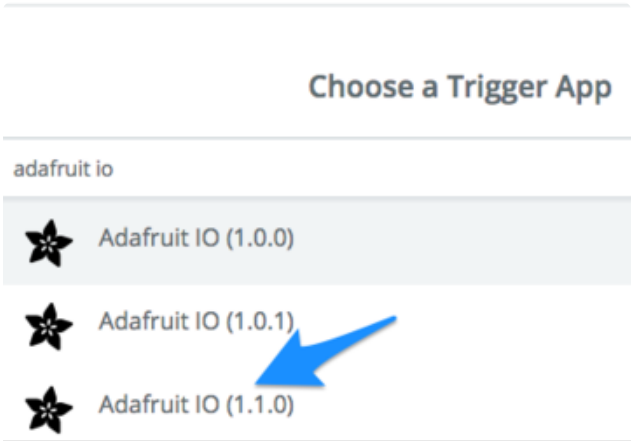


Setting up a Zap for Google Sheets

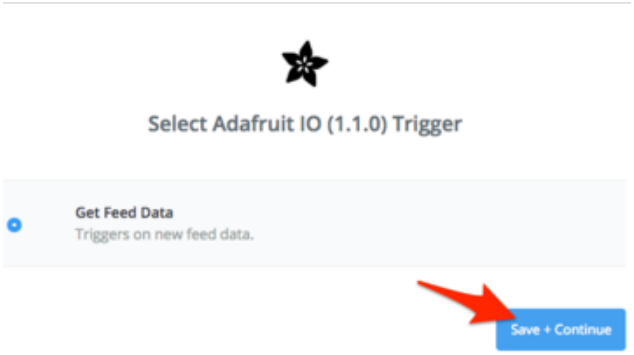
Next, we're create a **Zap**. A Zap is a combination of a trigger (like an Adafruit IO Feed receiving new data) and an action (like sending a Tweet or writing to a Spreadsheet).

To do this, **navigate to the Zapier Dashboard** (https://adafru.it/DCk) and **click Make a Zap!**

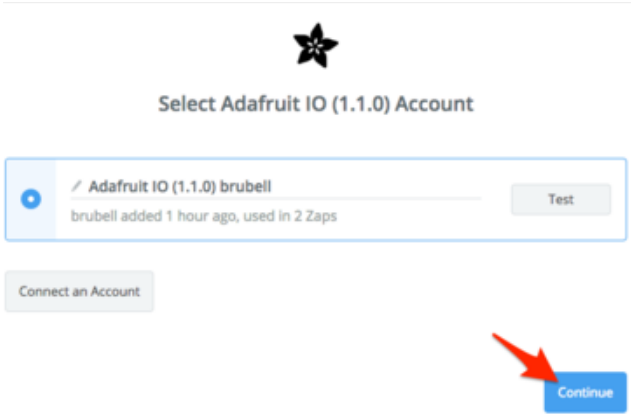




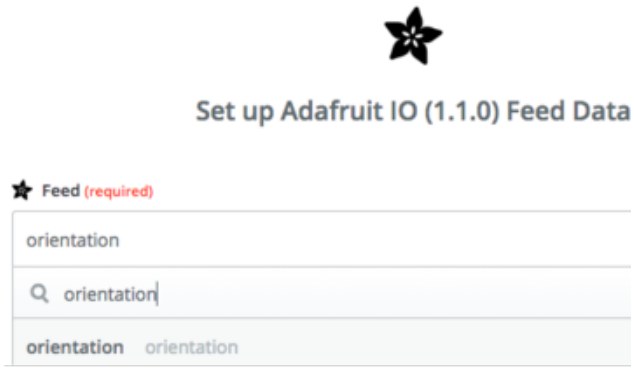
You'll be prompted to choose a trigger app. From the dropdown, select the latest version of Adafruit IO.



Next, you'll want a trigger. Select **Get Feed Data**, which triggers on new feed data.

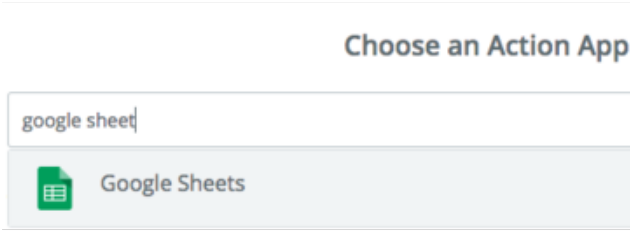


Next, **select your Adafruit IO Account** (or connect it if not done already).

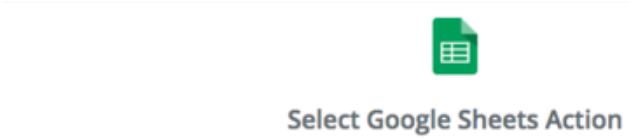


On the Set Up Options step, **Select the Orientation Feed** from the dropdown (if you have a lot of feeds - you can search for it).

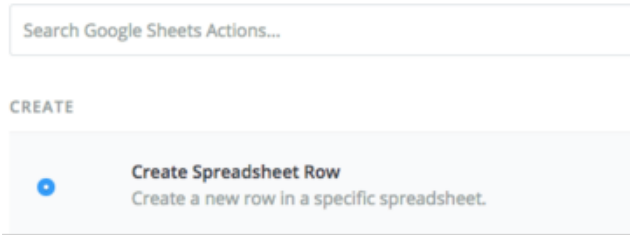
Finally, **select a data sample** (if the feed is newly created with no data, these won't exist yet).



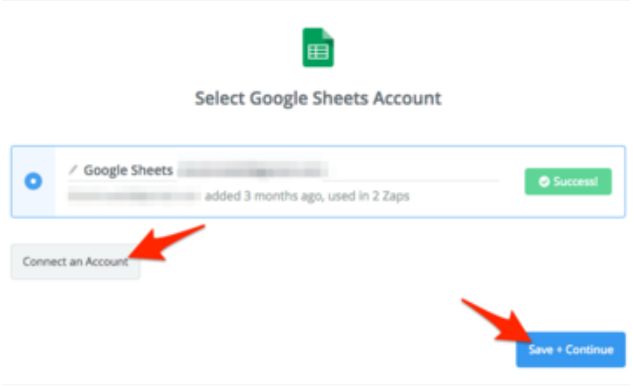
Next, we're going to set up the action. In this case, it'll be writing the value of the feed, and time which the data was sent to the feed, to a Google Sheet.



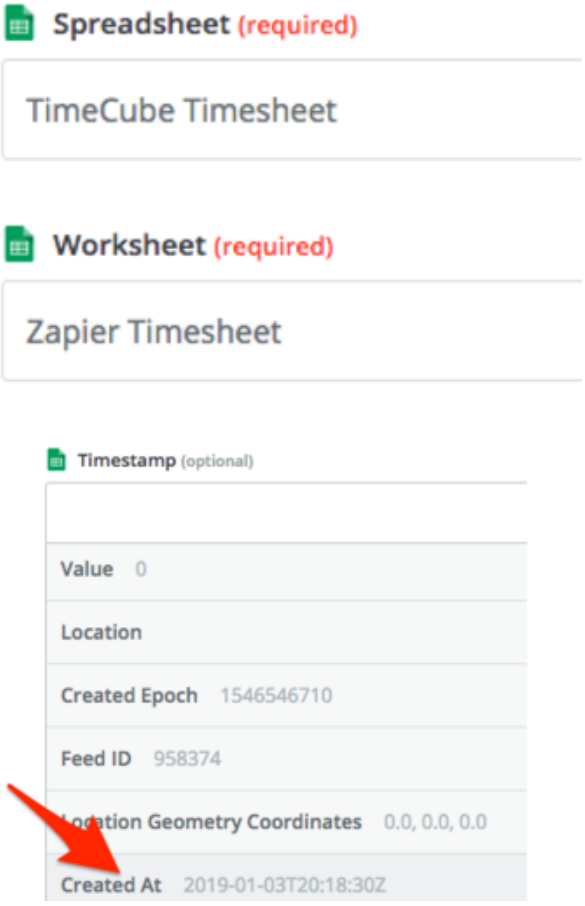
From **Choose an Action App**, **select Google Sheets**.



From **Select Google Sheets Action**, **select Create Spreadsheet Row**.



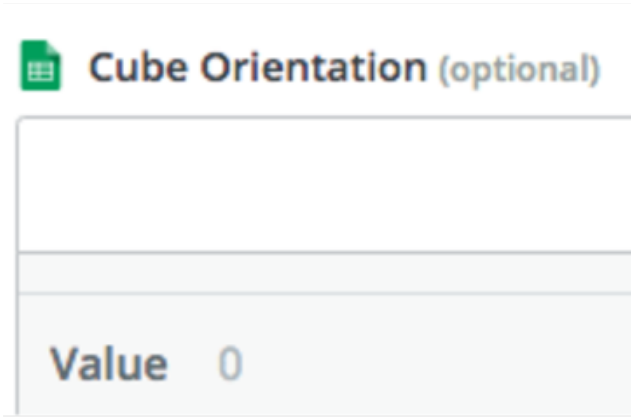
Click Connect an Account to link your Google Sheets Account (the same as your Google Account) with Zapier.



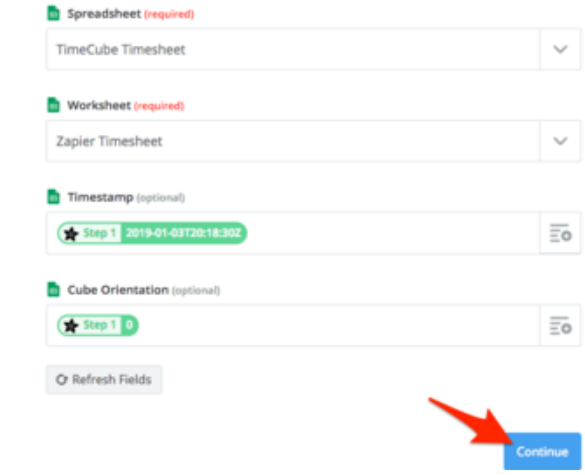
From the Spreadsheet dropdown, **select the spreadsheet** we made earlier from **Worksheet**. Zapier will automatically load in the column names.

Select the Timestamp column.

From the dropdown, **select Created At** as the data values to bring in.

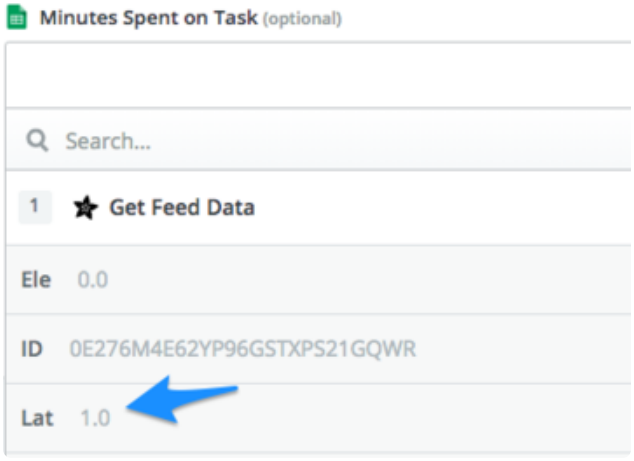


From the Cube Orientation column dropdown, **select the value field**.



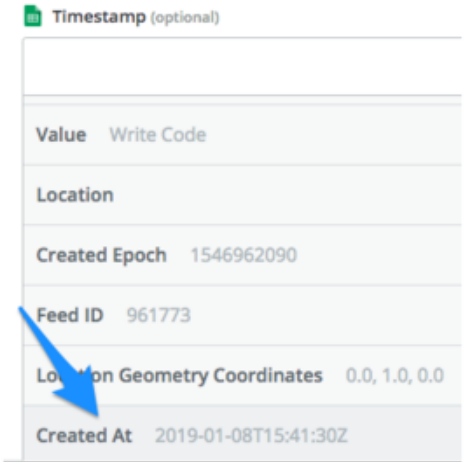
Click Continue to finish setting up the action.

Since we'll want to send the minutes **and** the task as Adafruit IO data, we'll need to get clever.



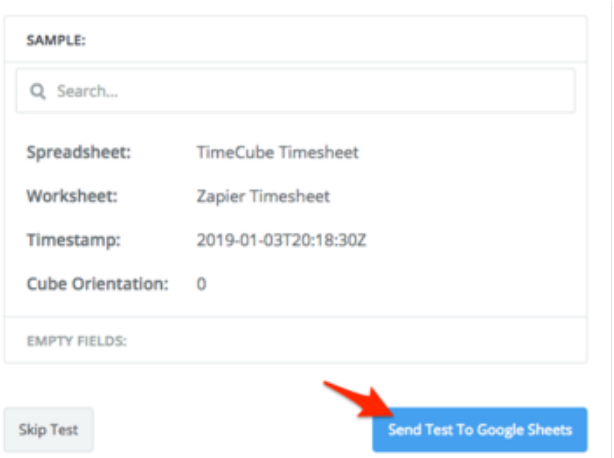
Each value sent to Adafruit IO has associated metadata - such as the time it was sent, and the ID of the data point. To send our timer in the same data point as our task, we'll be using the latitude metadata.

From Zapier, **select Minutes Spent on Task**. Scroll down to **select Lat**.

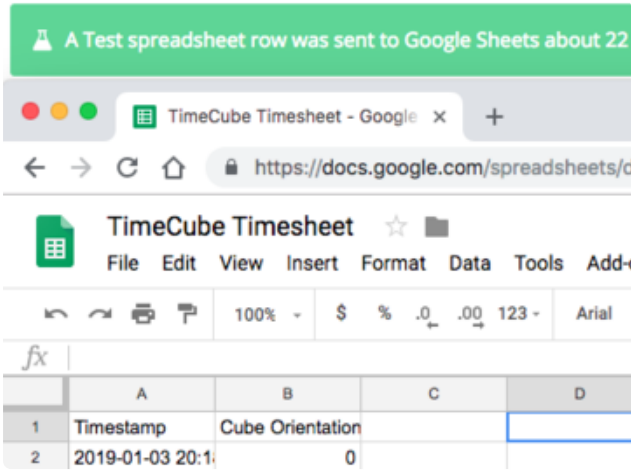


Another piece of metadata which we'll use in our spreadsheet is the timestamp - which is when the data was sent to Adafruit IO. This lets us keep track of our tasks and organize them by data or time.

From Zapier, **select Timestamp**. Scroll down to **select Created At**

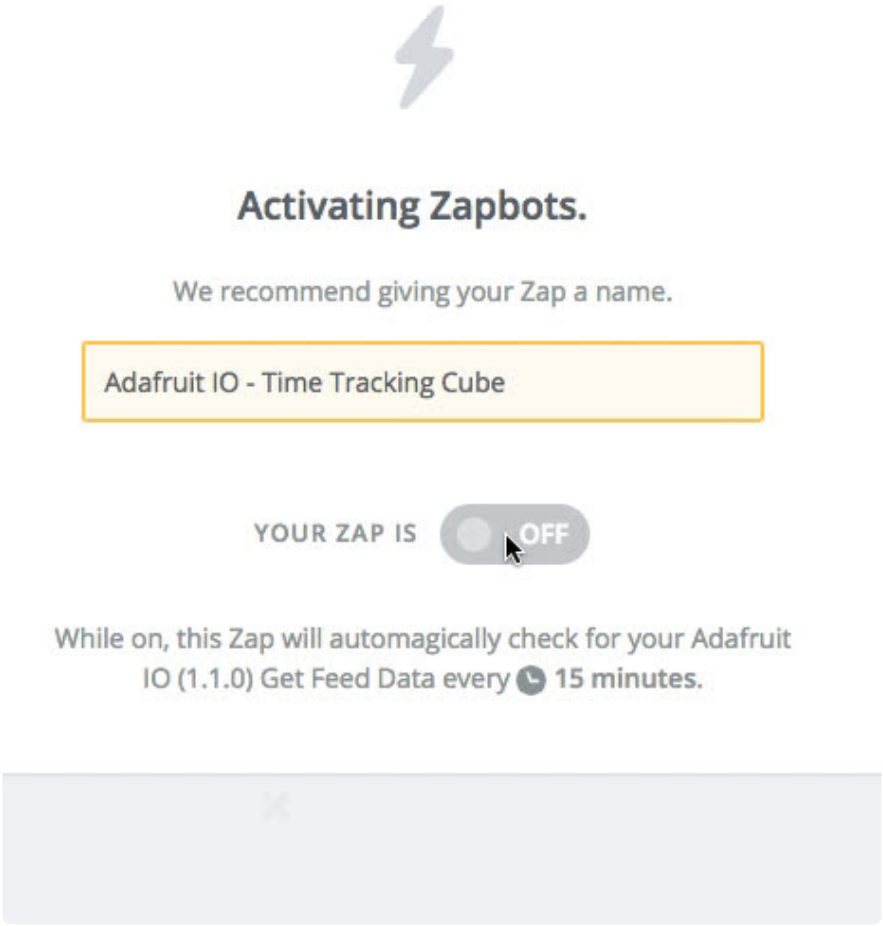


To test if the sheet was set up correctly, **click Send Test to Google Sheets**.



If everything was set up correctly, you'll see an updated row on the Google Sheet you've created earlier.

Finally, **give your zap a name** and **turn it on**.



Next, we'll move onto coding up the Time Cube.

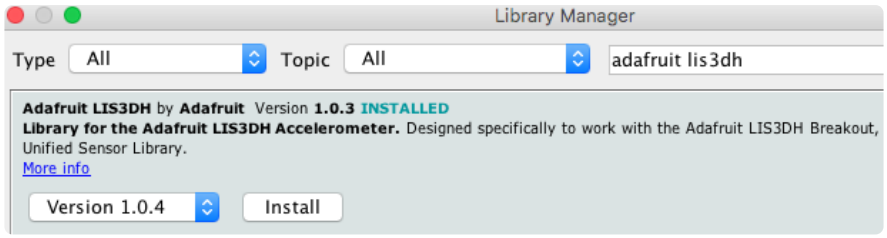
Arduino Setup

First, get the Feather Huzzah ESP8266 set up with the Arduino IDE and Adafruit IO.

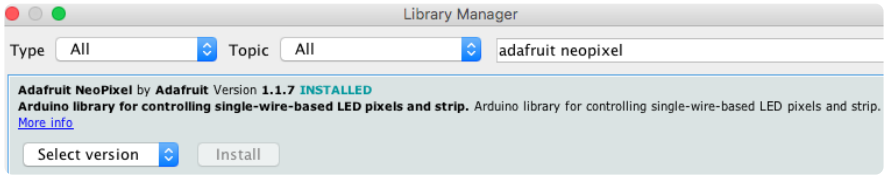
- If you haven't yet set up your ESP8266 for use with Adafruit IO and the Arduino IDE, follow along with this guide (https://adafru.it/DCI). The setup only needs to be performed once.

Installing Libraries

We'll want to use the **LIS3DH** sensor on the Prop-Maker FeatherWing. To do this, we'll install the **Adafruit LIS3DH** library from the Library Manager.



We'll also want to control the NeoPixel strip. From the library manager, install the **Adafruit NeoPixel** library.



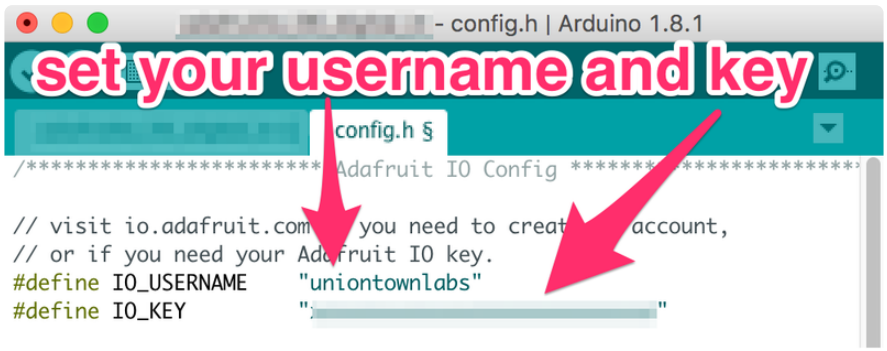
The code for this example is contained within the Adafruit IO Arduino Library (make sure your library version is 2.7.22 or later).

From the Arduino IDE, navigate to **File->Examples->Adafruit IO Arduino->adafruitio_24_zapier**

Next, we'll perform the network configuration required for this sketch.

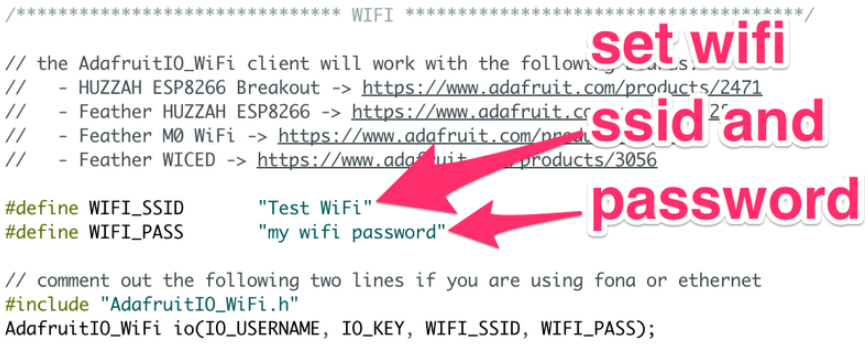
Arduino Network Config

To configure the network settings, click on the **config.h** tab in the sketch. You will need to set your Adafruit IO username in the **IO_USERNAME** define, and your Adafruit IO key in the **IO_KEY** define.



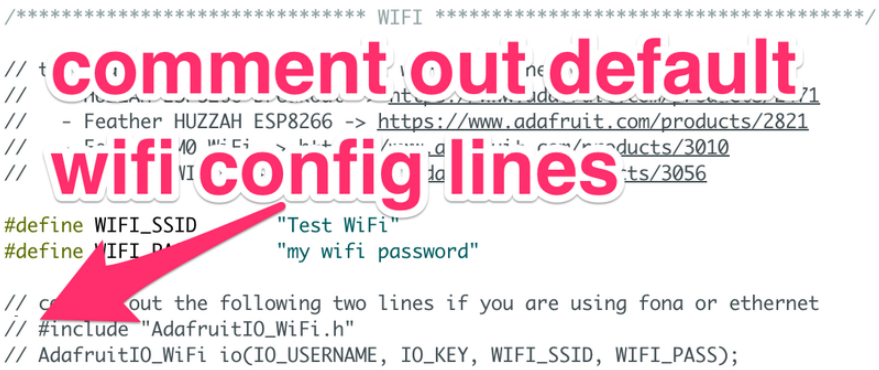
WiFi Config

WiFi is enabled by default in **config.h** so if you are using one of the supported WiFi boards, you will only need to modify the **WIFI_SSID** and **WIFI_PASS** options in the **config.h** tab.



FONA Config

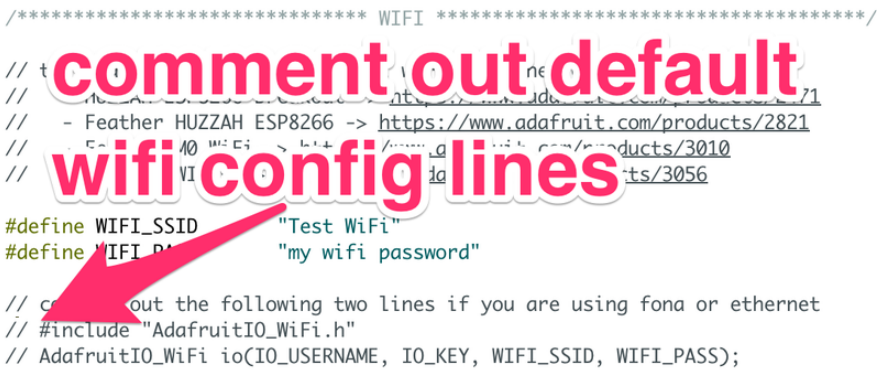
If you wish to use the FONA 32u4 Feather to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**



Next, remove the comments from both of the FONA config lines in the FONA section of **config.h** to enable FONA support.

Ethernet Config

If you wish to use the Ethernet Wing to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**



Next, remove the comments from both of the Ethernet config lines in the Ethernet section of **config.h** to enable Ethernet Wing support.

Next, we will look at how the example sketch works.

Code

Let's take a quick dip into the code powering the Time Tracking Cube. If you're interested in using an accelerometer to send data to Adafruit IO, this code can serve as a good jumping off point.

Within the `loop()`, a normalized sensor reading is taken from the accelerometer and we'll call a function to update the timer.

```
  // Update the timer
  updateTime();

  // Get a normalized sensor reading
  sensors_event_t event;
  lis.getEvent(&amp;event);
```

Then, the face orientation is detected within a conditional statement. For example, if the cube is tilted to the left, we'll detect it by querying the accelerometer's acceleration along the X axis:

```
      // Detect cube face orientation
  if (event.acceleration.x &gt; 9 &amp;&amp; event.acceleration.x &lt; 10) // left-
side up
  {
    //Serial.println("Cube TILTED: Left");
    cubeState = 1;
  }
```

We don't want the cube to register a right-side tilt if we've previously tilted it to the right. To do this, we'll compare the cube's state to its previous state.

```
  // return if the orientation hasn't changed
  if (cubeState == prvCubeState)
    return;
```

Then, we'll send the cube's previous task and the time spent on that task to Adafruit IO based off of the cube's orientation we detected earlier (using a switch-case statement (https://adafru.it/DCs)). On this new task, the NeoPixel strip is updated, the cube will play a tone, and we'll reset the timer. The idle state (case 3) will play a tone and update the NeoPixels, but it won't send to Adafruit IO.

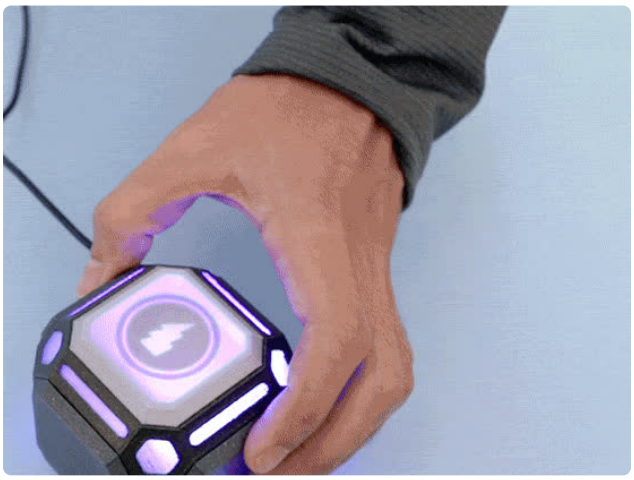
```
  // Send to Adafruit IO based off of the orientation of the cube
  switch (cubeState)
  {
  case 1:
    Serial.println("Switching to Task 1");
    // update the neopixel strip
    updatePixels(50, 0, 0);
    // play a sound
    tone(PIEZO_PIN, 650, 300);
    Serial.print("Sending to Adafruit IO -&gt; ");
    Serial.println(taskTwo);
    cubetask-&gt;save(taskTwo, minutes);
    // reset the timer
    minutes = 0;
    break;
  case 2:
    Serial.println("Switching to Task 2");
    // update the neopixel strip
    updatePixels(0, 50, 0);
    // play a sound
    tone(PIEZO_PIN, 850, 300);
    Serial.print("Sending to Adafruit IO -&gt; ");
    Serial.println(taskOne);
    cubetask-&gt;save(taskOne, minutes);
```

```
    // reset the timer
    minutes = 0;
    break;
  case 3:
    updatePixels(0, 0, 50);
    tone(PIEZO_PIN, 950, 300);
    break;
  }
```

Using the Time Tracking Cube

Upload the code to your Feather Huzzah. Then, open the Arduino Serial Monitor (**Tools -> Serial Monitor)**.

```
Adafruit IO Time Tracking Cube
LIS3DH found!
Pixels init'd
Connecting to Adafruit IOAdafruitIO::connect()
.
Adafruit IO connected.
```

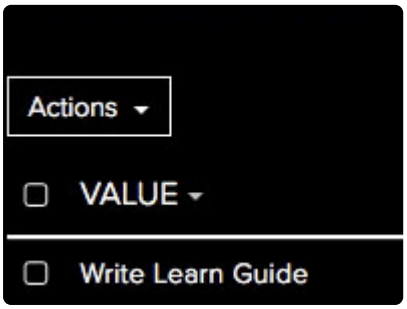


Keep the serial monitor open and tilt the cube the left. The cube will glow **red** and the serial monitor will print out:

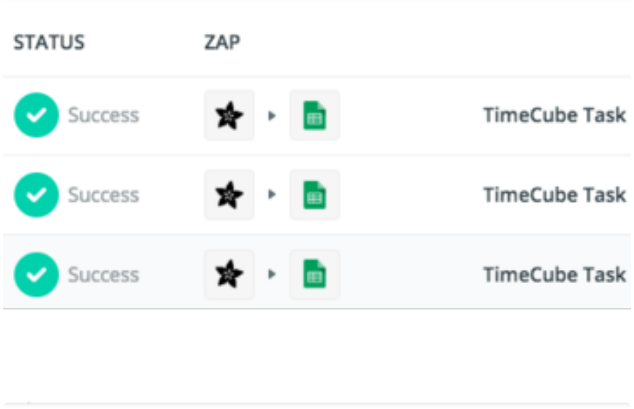
```
Switching to Task 1
Sending to Adafruit IO -> Write Code
```

Tilting the cube to the right will make it switch to the second task and the cube will glow **green**.

Flipping the cube will change the value in real-time.

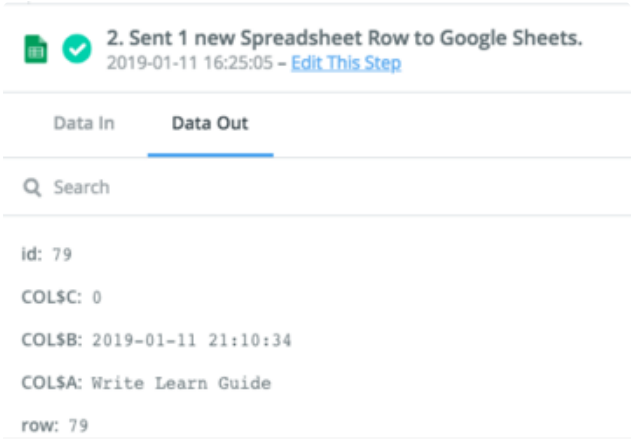


You can check that the task has been sent to Adafruit IO by navigating to the feed on the Adafruit IO website. You should see the value set to the previous task.



Next, we'll want to check that Zapier is working properly.

Navigate to the Task History Page on Zapier. If the Zap executed successfully, it'll display success under its status.

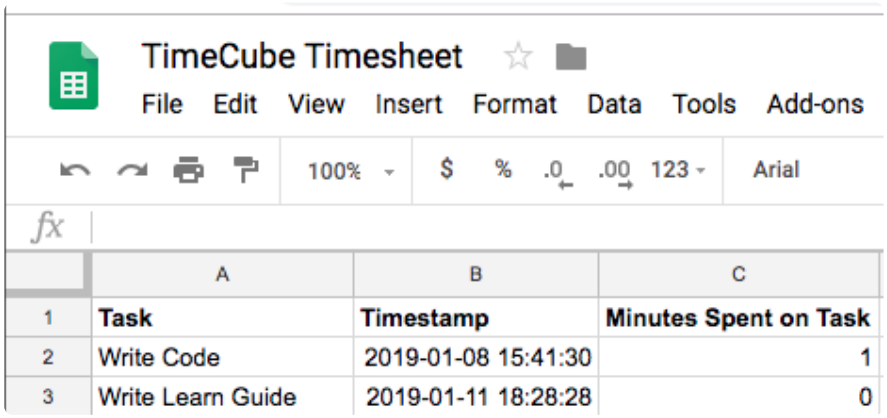


Clicking the task will show you which data was found in Adafruit IO and what was written to the Google Sheet row.

Don't see anything on this page, but the Zap is turned on?

The time cube zap runs every 15 minutes, grabbing the latest data from the Adafruit IO feed.

If the Zap worked properly, you'll see the spreadsheet updated with new values every 15 minutes.



Taking it Further

Now that we have the time cube sending two tasks to Adafruit IO - the code can be extended to sending more tasks to Adafruit IO. How many tasks are possible? The time cube has six sides - so, six unique tasks are possible.

We're also using the Prop-Maker FeatherWing for this project, which opens up more possibilities. You can take this project further by adding in a button (possibly along one the back face of the cube) to power-down the cube when it's not in use.

What about if you have more than six tasks? The Prop-Maker FeatherWing has breakouts for an external switch, which you can use as a mode-selector.

Code

```
// Adafruit IO Time Tracking Cube
// Tutorial Link: https://learn.adafruit.com/time-tracking-cube
//
// Adafruit invests time and resources providing this open source code.
// Please support Adafruit and open source hardware by purchasing
// products from Adafruit!
//
// Written by Brent Rubell for Adafruit Industries
// Copyright (c) 2019 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 <Wire.h>
#include <Adafruit_LIS3DH.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_NeoPixel.h>

// Prop-Maker Wing
#define NEOPIXEL_PIN 2
#define POWER_PIN 15

// Used for Pizeo
#define PIEZO_PIN 0

// # of Pixels Attached
#define NUM_PIXELS 8

// Adafruit_LIS3DH Setup
Adafruit_LIS3DH lis = Adafruit_LIS3DH();

// NeoPixel Setup
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_PIXELS, NEOPIXEL_PIN, NEO_GRB +
NEO_KHZ800);
```

```
// Set up the 'cubeTask' feed
AdafruitIO_Feed *cubetask = io.feed("cubetask");

/* Time Tracking Cube States
 * 1: Cube Tilted Left
 * 2: Cube Tilted Right
 * 3: Cube Neutral, Top
*/
int cubeState = 0;

// Previous cube orientation state
int prvCubeState = 0;

// Tasks (change these to what you're currently working on)
String taskOne = "Write Learn Guide";
String taskTwo = "Write Code";

// Adafruit IO sending delay, in seconds
int sendDelay = 0.5;

// Time-Keeping
unsigned long currentTime;
unsigned long prevTime;
int seconds = 0;
int minutes = 0;

void setup()
{
  // start the serial connection
  Serial.begin(9600);
  // wait for serial monitor to open
  while (!Serial)
    ;
  Serial.println("Adafruit IO Time Tracking Cube");

  // disabling low-power mode on the prop-maker wing
  pinMode(POWER_PIN, OUTPUT);
  digitalWrite(POWER_PIN, HIGH);

  // Initialize LIS3DH
  if (!lis.begin(0x18))
  {
    Serial.println("Couldnt start");
    while (1)
      ;
  }
  Serial.println("LIS3DH found!");
  lis.setRange(LIS3DH_RANGE_4_G);

  // Initialize NeoPixel Strip
  strip.begin();
  Serial.println("Pixels init'd");

  // 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 updateTime()
{
  // grab the current time from millis()
  currentTime = millis() / 1000;
  seconds = currentTime - prevTime;
  // increase mins.
  if (seconds == 60)
  {
    prevTime = currentTime;
    minutes++;
  }
}

void updatePixels(uint8_t red, uint8_t green, uint8_t blue)
{
  for (int p = 0; p < NUM_PIXELS; p++)
  {
    strip.setPixelColor(p, red, green, blue);
  }
  strip.show();
}

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();

  // Update the timer
  updateTime();

  // Get a normalized sensor reading
  sensors_event_t event;
  lis.getEvent(&event);

  // Detect cube face orientation
  if (event.acceleration.x > 9 && event.acceleration.x < 10)
  {
    //Serial.println("Cube TILTED: Left");
    cubeState = 1;
  }
  else if (event.acceleration.x < -9)
  {
    //Serial.println("Cube TILTED: Right");
    cubeState = 2;
  }
  else if (event.acceleration.y < 0 && event.acceleration.y > -1)
  {
    cubeState = 3;
  }
  else
  { // orientation not specified
    //Serial.println("Cube Idle...");
  }

  // return if the orientation hasn't changed
  if (cubeState == prvCubeState)
    return;

  // Send to Adafruit IO based off of the orientation of the cube
  switch (cubeState)
  {
  case 1:
    Serial.println("Switching to Task 1");
    // update the neopixel strip
    updatePixels(50, 0, 0);
```

```
    // play a sound
    #if defined(ARDUINO_ARCH_ESP32)
        ledcWriteTone(PIEZO_PIN, 650);
    #else
        tone(PIEZO_PIN, 650, 300);
    #endif

    Serial.print("Sending to Adafruit IO -> ");
    Serial.println(taskTwo);
    cubetask->save(taskTwo, minutes);
    // reset the timer
    minutes = 0;
    break;
  case 2:
    Serial.println("Switching to Task 2");
    // update the neopixel strip
    updatePixels(0, 50, 0);

    // play a sound
    #if defined(ARDUINO_ARCH_ESP32)
        ledcWriteTone(PIEZO_PIN, 850);
    #else
        tone(PIEZO_PIN, 850, 300);
    #endif

    Serial.print("Sending to Adafruit IO -> ");
    Serial.println(taskOne);
    cubetask->save(taskOne, minutes);
    // reset the timer
    minutes = 0;
    break;
  case 3:
    updatePixels(0, 0, 50);

    // play a sound
    #if defined(ARDUINO_ARCH_ESP32)
        ledcWriteTone(PIEZO_PIN, 950);
    #else
        tone(PIEZO_PIN, 950, 300);
    #endif

    break;
  }

  // save cube state
  prvCubeState = cubeState;

  // Delay the send to Adafruit IO
  delay(sendDelay * 1000);
}
```