# MUNNY Glowing Friend with Bluetooth Control!

Created by John Park

https://learn.adafruit.com/munny-lamp

Last updated on 2024-06-03 02:32:25 PM EDT

# Table of Contents

# Overview

MUNNY DIY is a designer toy figure designed to be modified. So, instead of simply painting him to change his appearance, let's fill him with electronics so he can glow all the colors of the rainbow at the touch of a wireless button!

We'll use the Adafruit Prop-Maker Wing (http://adafru.it/3988) paired with a Feather M0 Bluefruit LE board (http://adafru.it/2995) to drive a 3W RGB LED (http://adafru.it/2530). Communications will come from the free Adafruit Bluefruit Connect app on your iOS or Android device!

## Parts

**1 x** Adafruit Prop-Maker
FeatherWing
https://www.adafruit.com/product/3988

**1 x** Adafruit Feather M0 Bluefruit LE
Bluetooth LE microcontroller of the future
https://www.adafruit.com/product/2995

**1 x** 3W RGB LED
Common Anode
https://www.adafruit.com/product/2530

**1 x** Lithium Ion Cylindrical Battery
3.7v 2200mAh
https://www.adafruit.com/product/1781

Mini Panel Mount

**1 x** SPDT Toggle Switch — https://www.adafruit.com/product/3221
Mini Panel Mount

---

**1 x** Premium Male/Female Raw Jumper Wires — https://www.adafruit.com/product/3633
40 x 6"

---

**1 x** Small Single Row Wire Housing Pack — https://www.adafruit.com/product/3145
for DIY Jumper Cables

---

**1 x** Panel Mount Extension USB Cable — https://www.adafruit.com/product/3258
Micro B Male to Micro B Female

---

**1 x** 5V 2.5A Switching Power Supply — https://www.adafruit.com/product/1995
with 20AWG MicroUSB Cable

---

**1 x** USB Patterned Fabric Cable — https://www.adafruit.com/product/2008
A/MicroB

---

**1 x** Break-away 0.1" 36-pin strip right-angle male header — https://www.adafruit.com/product/1540
10 pack

---

**1 x** Short Feather Male Headers — https://www.adafruit.com/product/3002
12-pin and 16-pin set

---

**1 x** Short Feather Female Headers — https://www.adafruit.com/product/2940
12-pin and 16-pin set

---

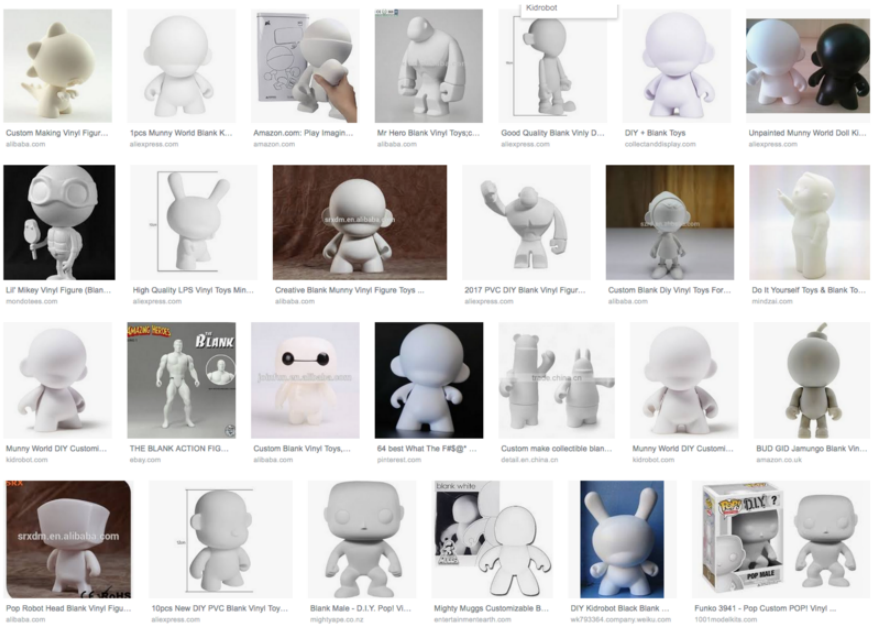
**1 x** Multi-Colored Heat Shrink Pack — https://www.adafruit.com/product/1649
3/32" + 1/8" + 3/16" Diameters

---

## Materials

You'll need to get a plain white urban vinyl toy, such as our good friend MUNNY from Kid Robot (https://adafru.it/CXd). For this guide I used the 7" version. You could use the smaller 4" version, but you would need to leave some of the parts outside the guy!


A quick search for the term **blank vinyl figure** yields all sorts of fun figures that would look great filled with colored light!



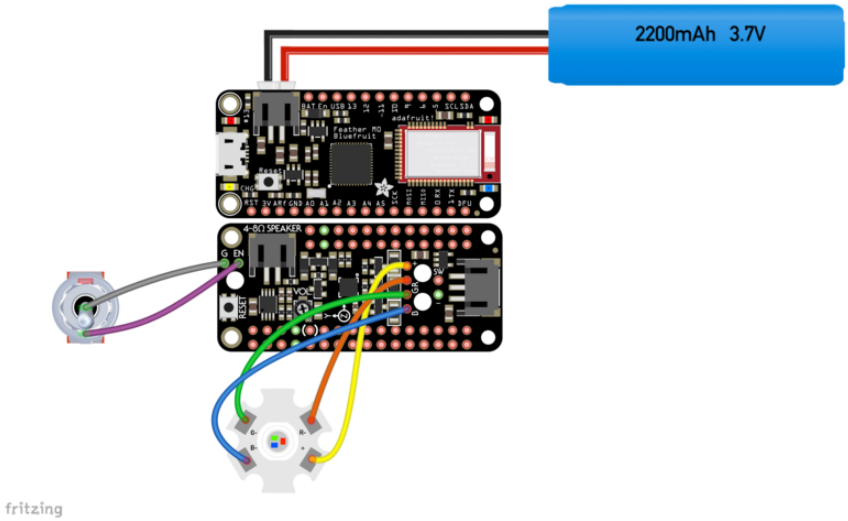
## Tools

The only tool required to operate on your vinyl figure is a hobby knife. Optionally, a heat gun or hair dryer will allow you to soften the vinyl and make it much easier to cut.

For the circuit, you'll need a soldering iron and solder, wire cutters, and wire strippers.

Be sure to be careful with sharp tools and soldering equipment that can get quite hot.

# Build the Circuit

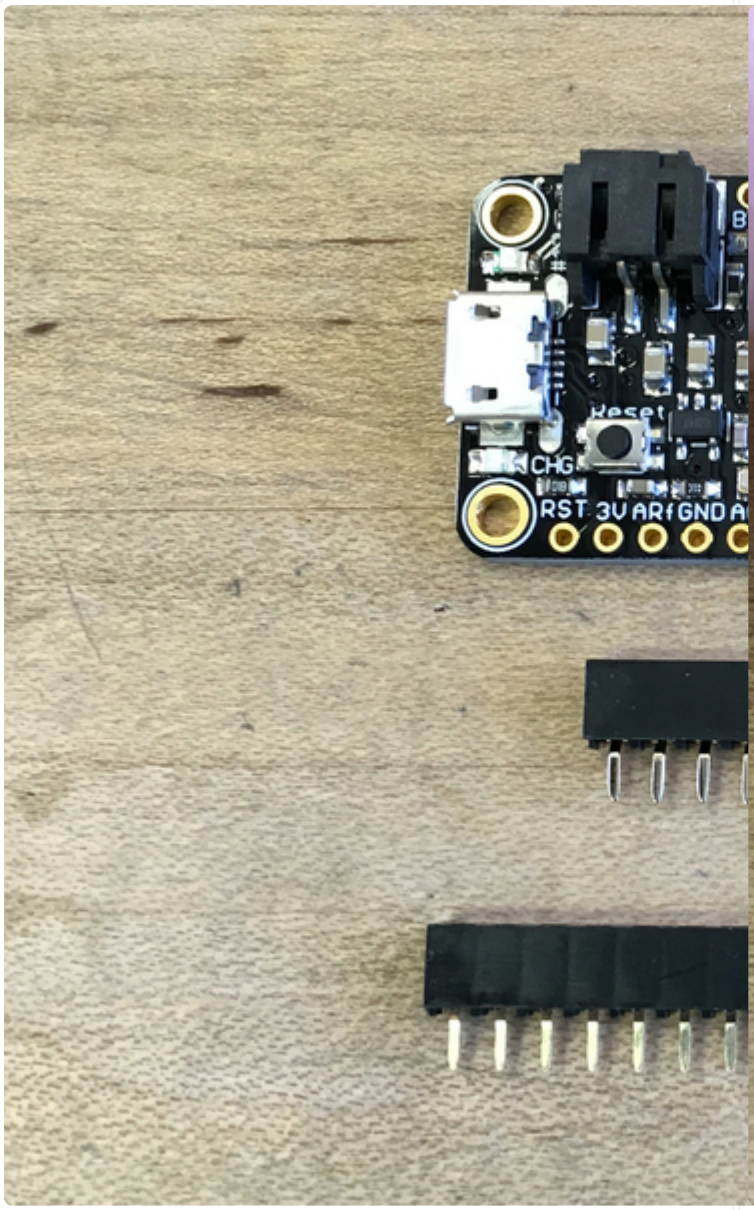


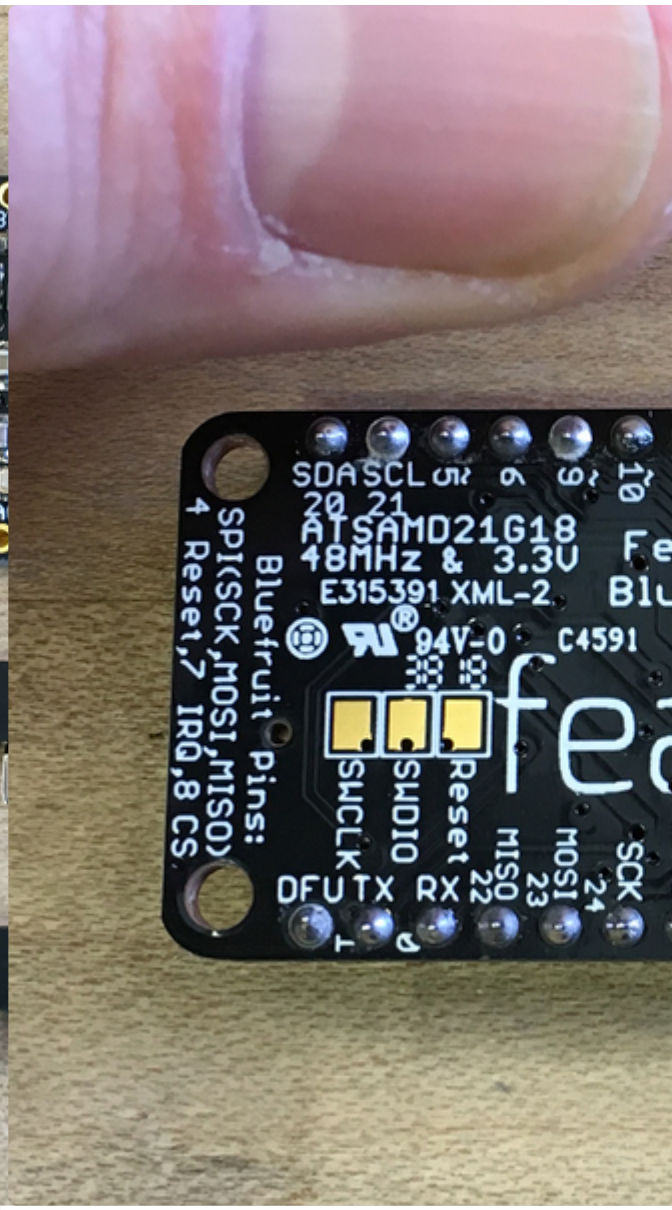

This diagram shows how the circuit will be laid out. The RGB LED and on/off switch will be connected to the Prop-Maker Wing using header pins and shrouded ("DuPont") connectors. We'll have the Prop-Maker Wing plugged into the Feather M0 Bluefruit, and the battery plugged into the Feather.

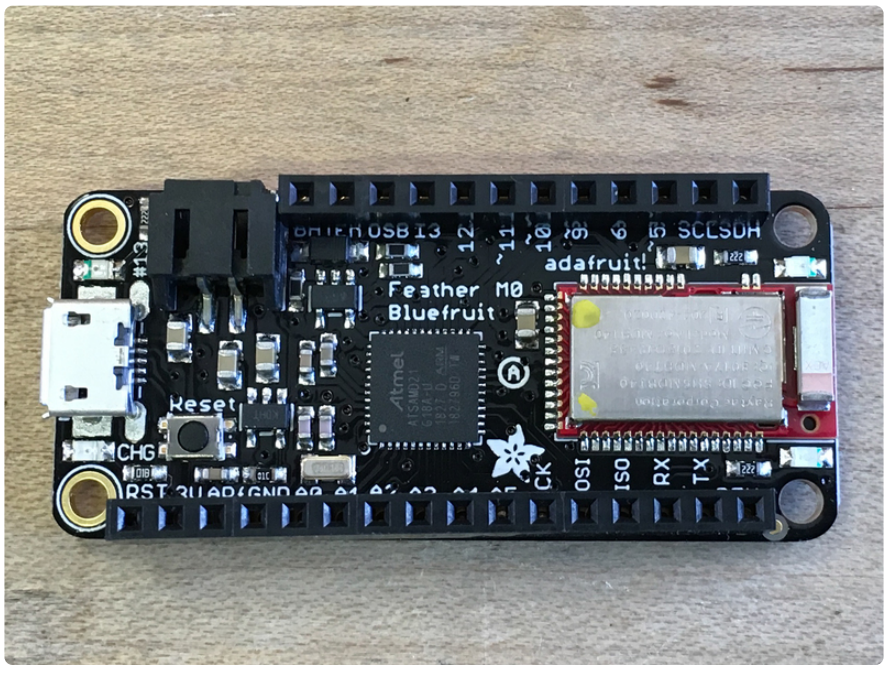
## Feather Headers

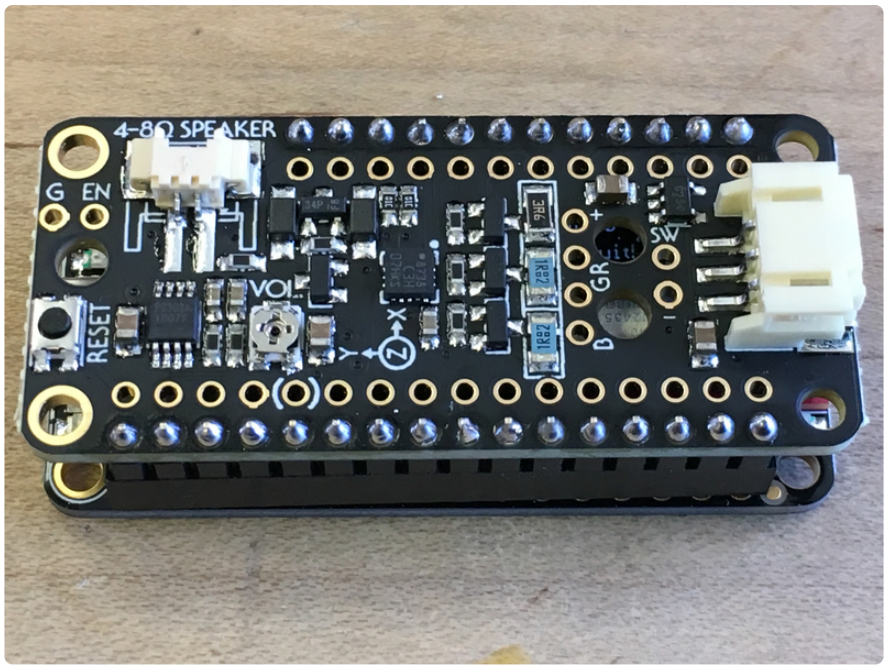
To keep a low profile when fitting the electronics into our MUNNY figure, we'll use low-profile headers to connect the Prop-Maker Wing to the Feather M0 Bluefruit board.

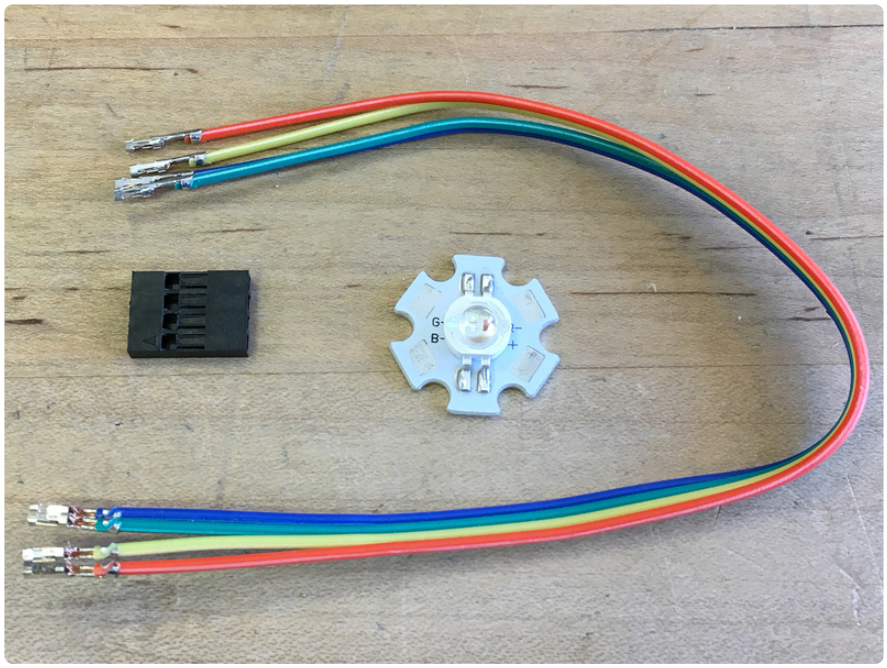
Start by soldering the female headers to the Feather and the male headers to the Prop-Maker Wing as shown below.


Reset
CHG
RST 3V AR GND


SDA SCL
20 21
ATSAMD21G18
48MHz & 3.3V
E315391 XML-2
94V-0
C4591
Bluefruit Pins:
SPI(SCK,MOSI,MISO)
4 Reset,7 IRQ,8 CS
SWCLK
SWDIO
Reset
DFU TX RX
MISO 22
MOSI 23
SCK 24


USB
SCL SDA
adafruit!
Feather M0
Bluefruit
Reset
CHG
RST
SCK
MOSI
MISO
RX
TX



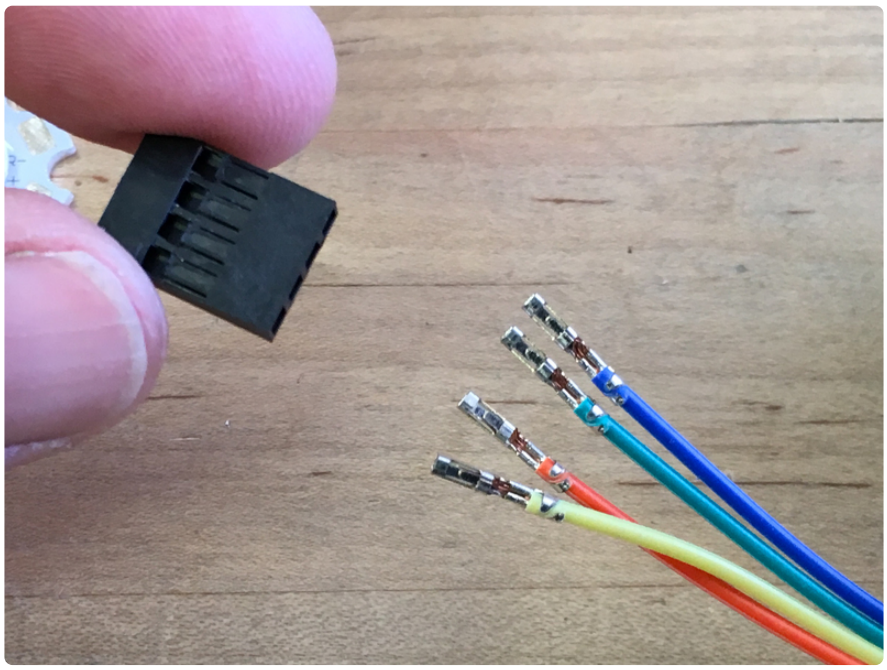
## 3W LED Wiring

We'll use a set of four jumper wires to connect the LED to the board. The Prop-Maker Wing has a common anode RGB LED driver (triple MOSFET action!) and four breakout pads. We'll connect right-angle header pins to the board, and use a single-row wire housing on female jumper wires to make the connect to the LED.

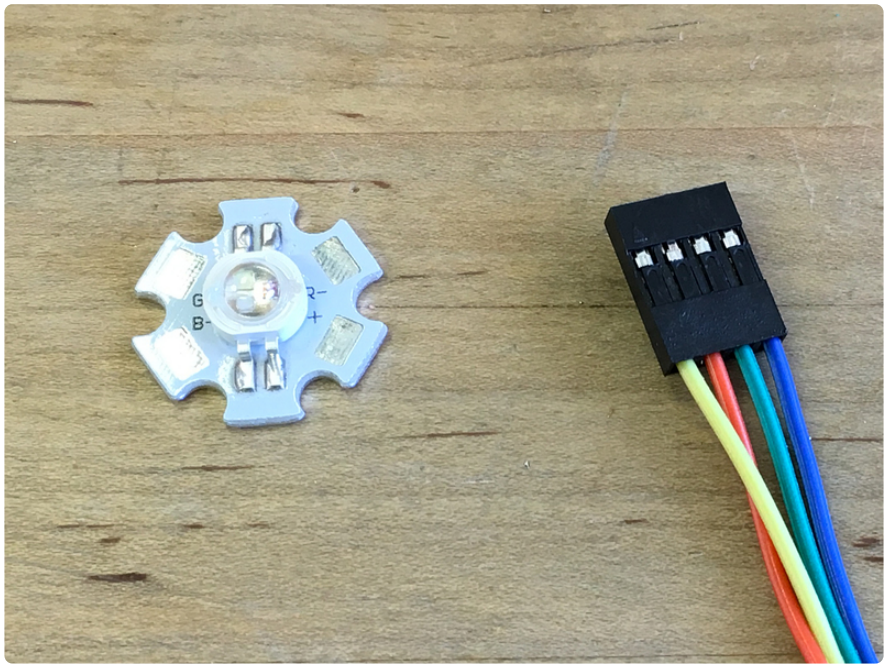
Start by connecting the four wires as shown into the housing. I'm using:

- **yellow** for + **voltage**

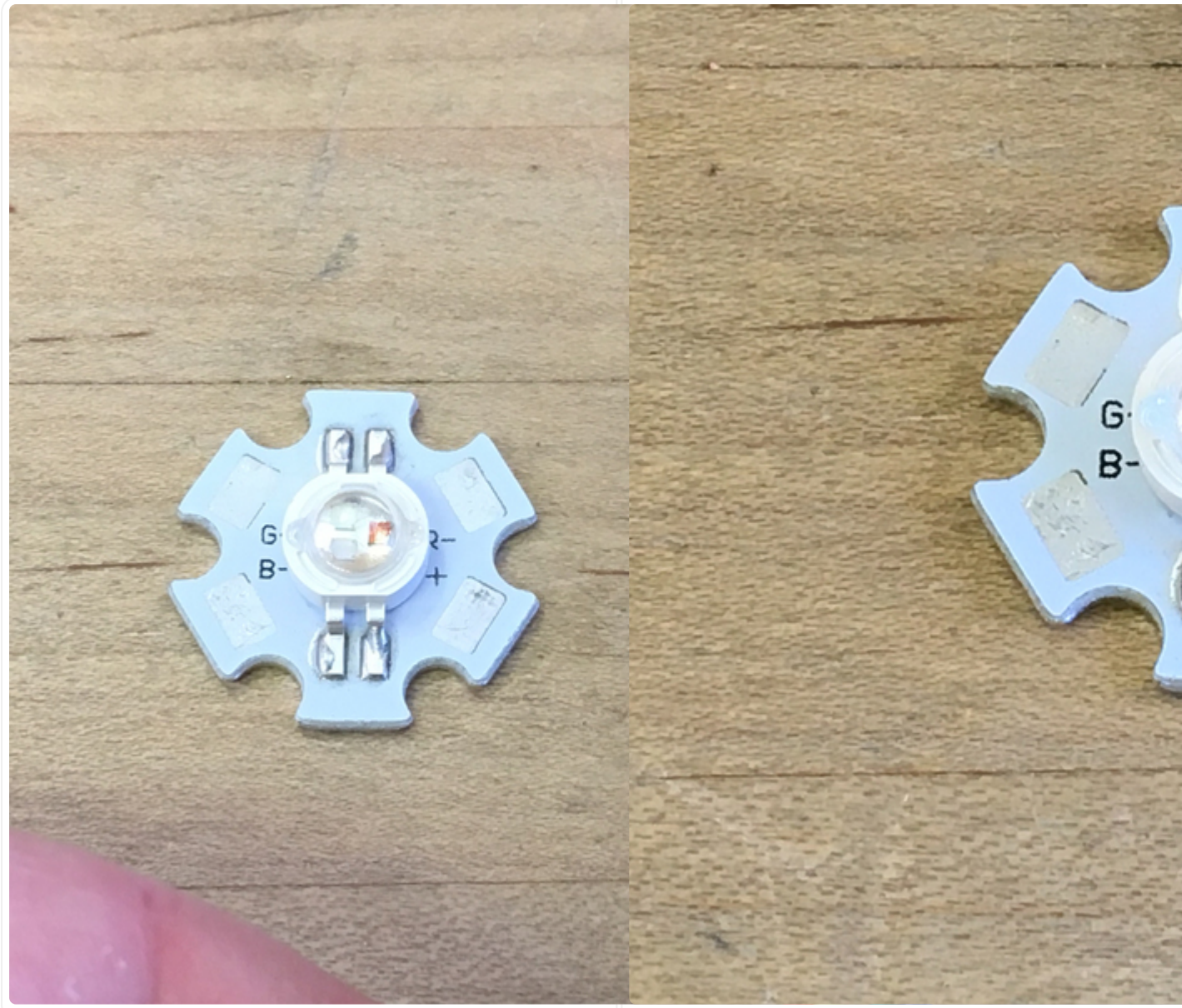
- **orange** for **R**
- **green** for **G**
- **blue** for **B**



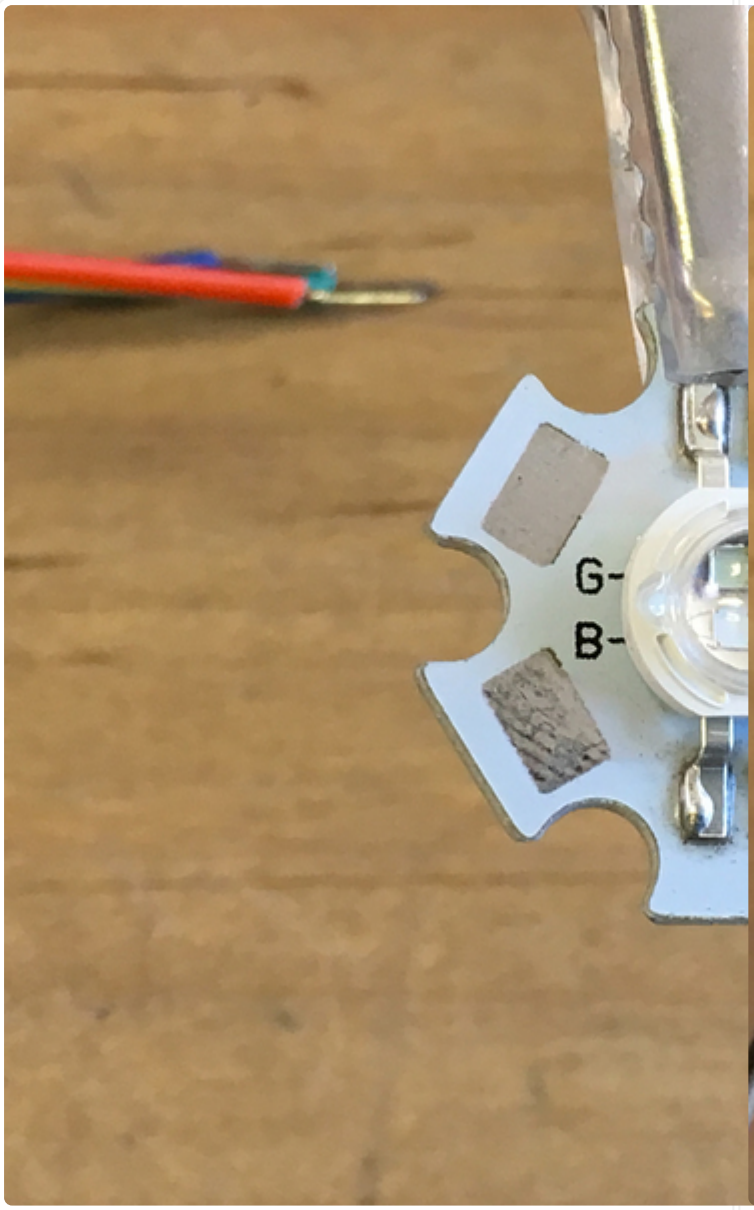
The wire crimp connectors click satisfyingly into place! **Note how the yellow wire is routed to the far left.**

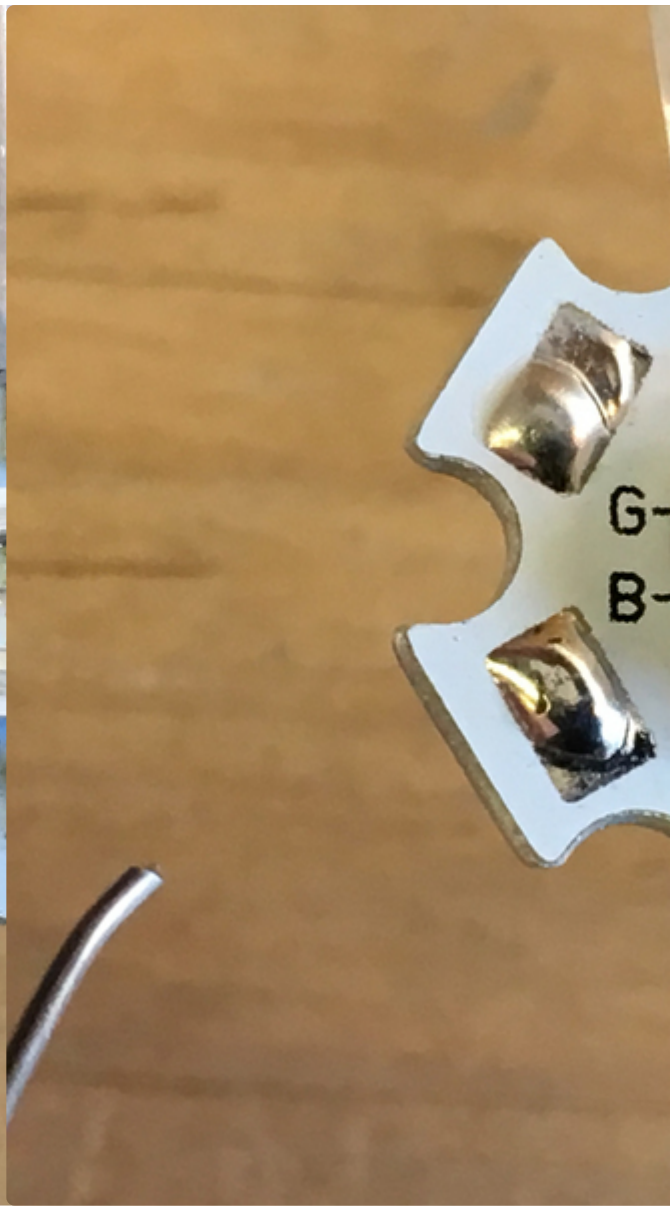


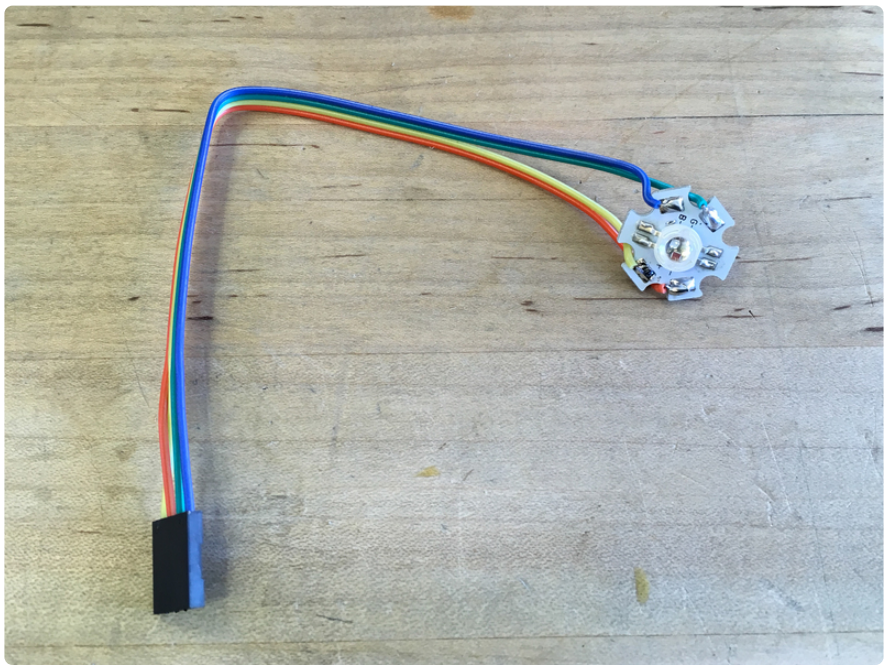

Trim, strip, and tin the other end of the wires as shown in preparation for soldering to the LED.




Next, tin the four pads on the LED and then solder the wires to the corresponding pads.

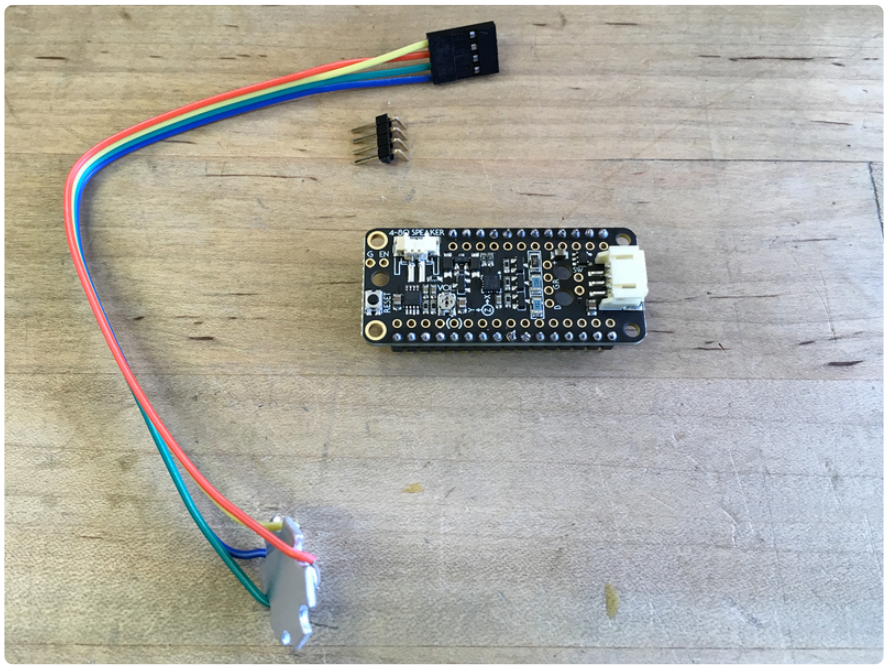

G-
B-

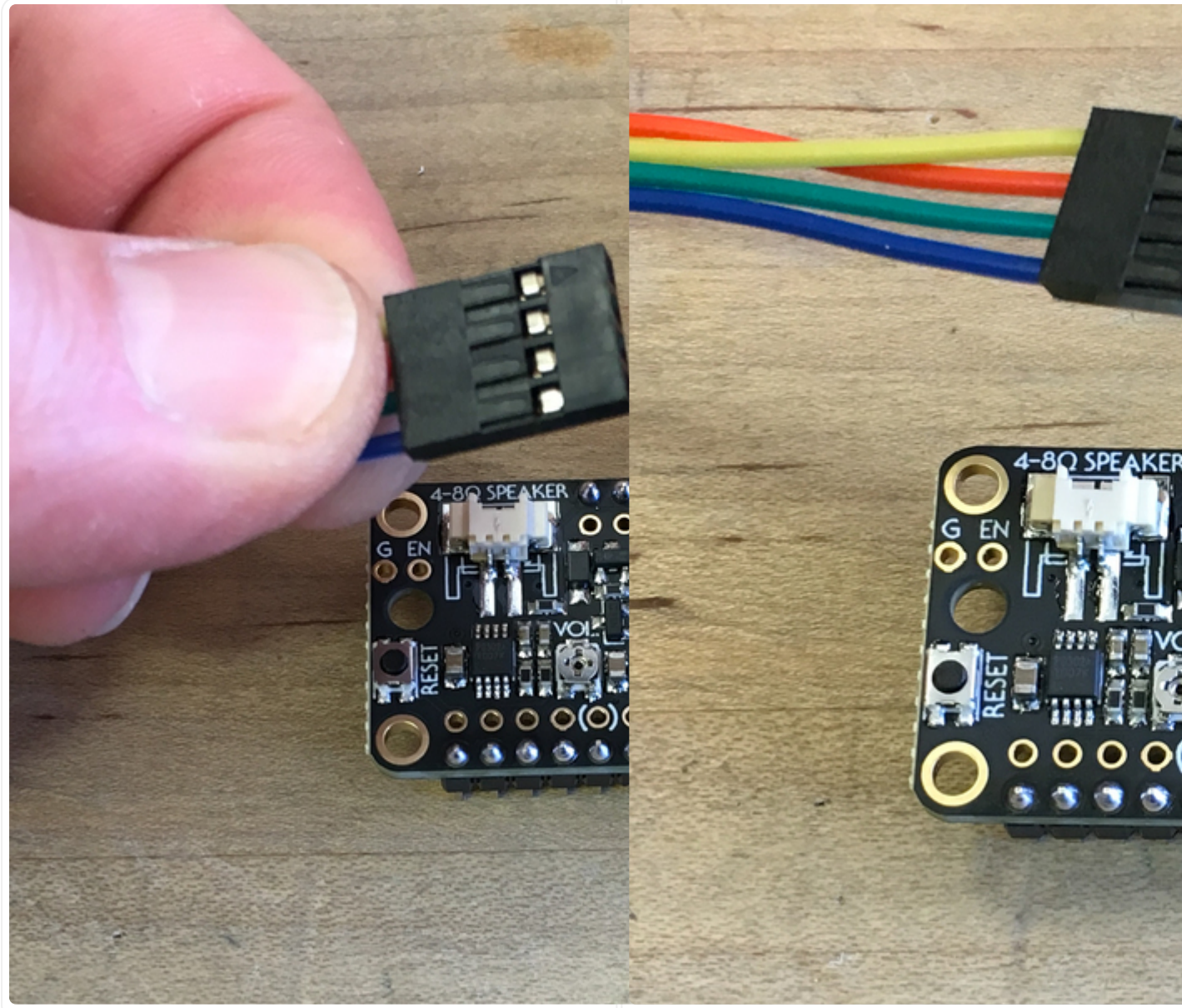

G-
B-



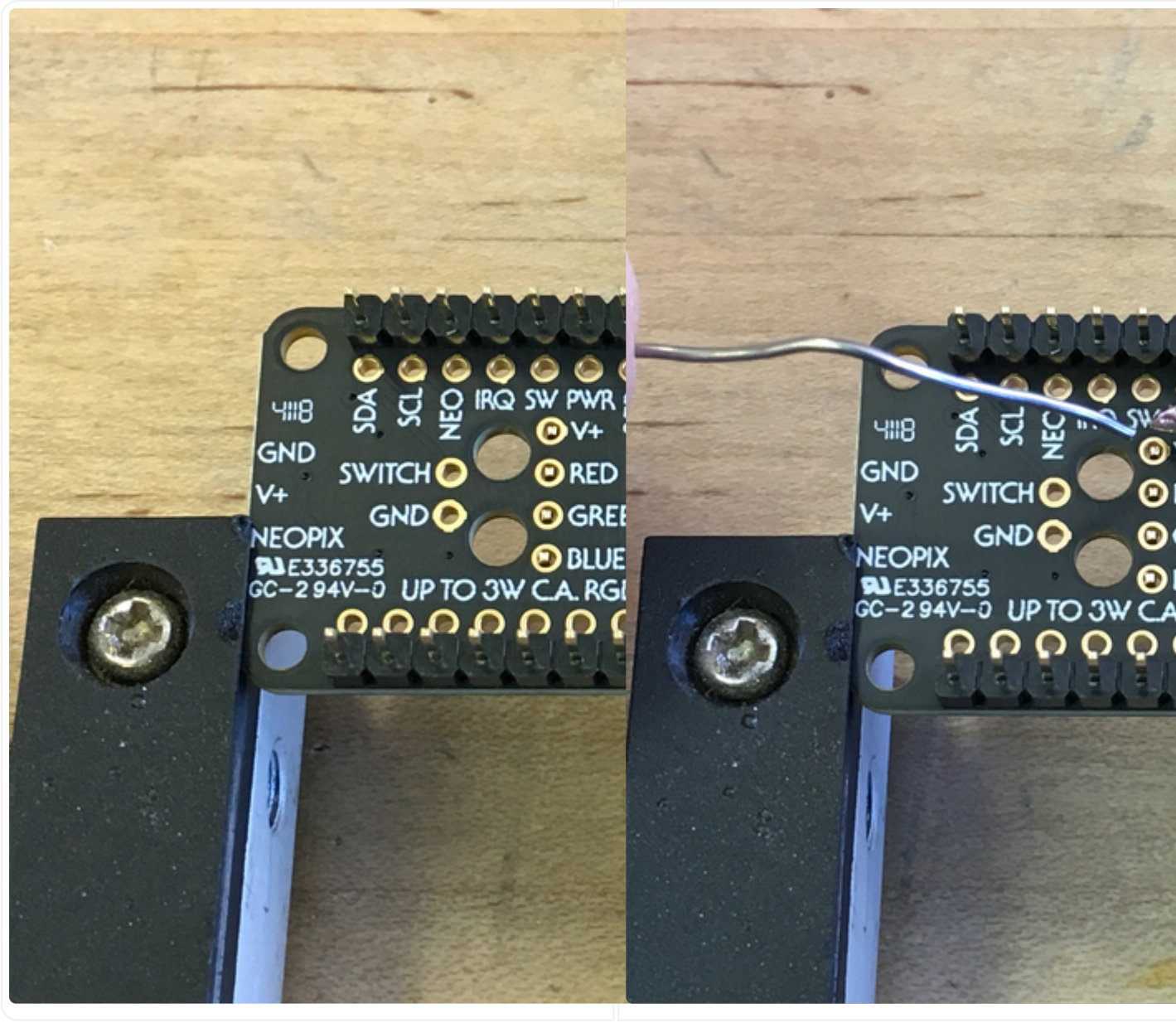
## LED Headers

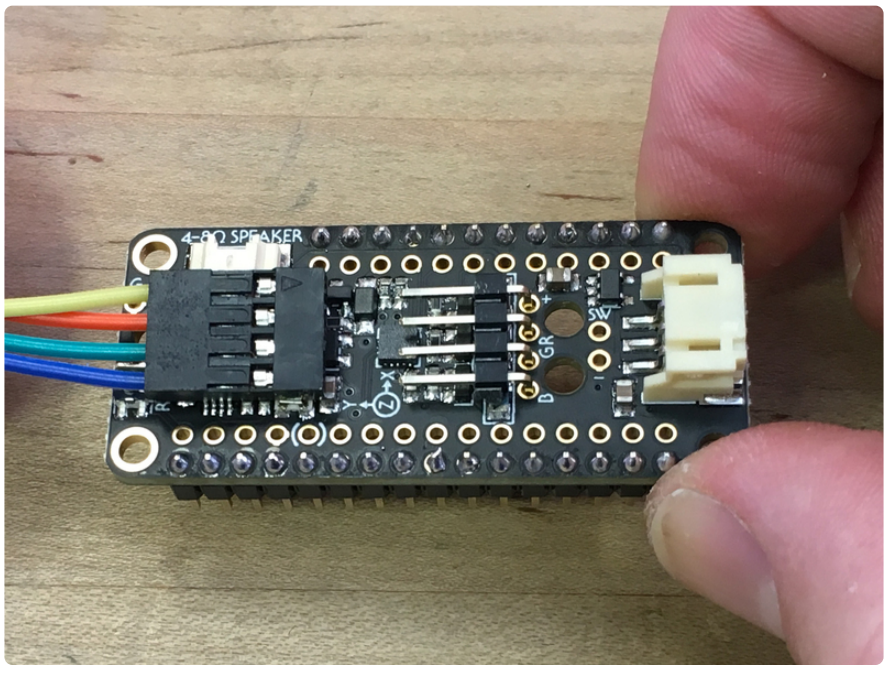
Snap off a 4-pin section of right-angled header pins. To get a good placement, insert them into the LED cable housing, fit them onto the Prop-Maker Wing, and tape it in place.





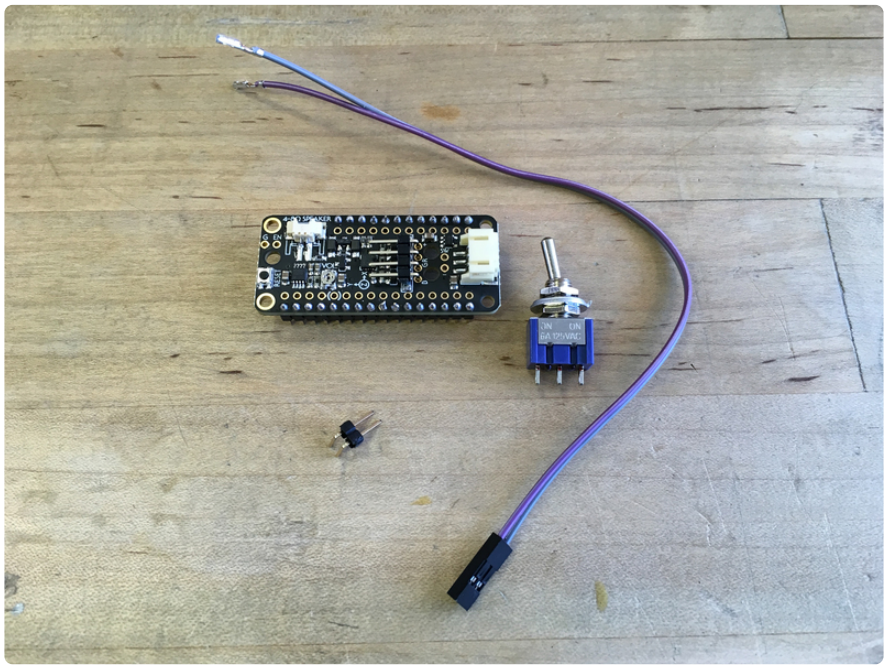

Flip the board over and solder the four pins in place.

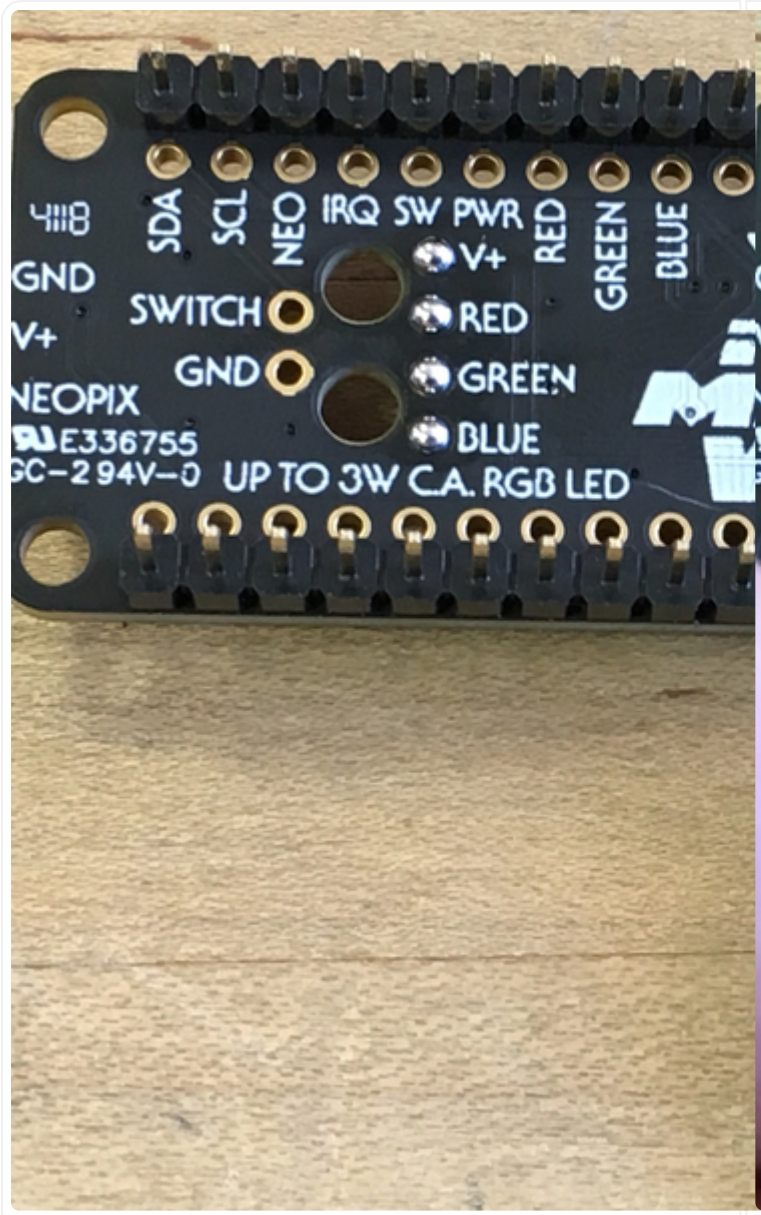

SDA
SCL
NEO
IRQ
SW
PWR
V+
GND
SWITCH
RED
GND
V+
NEOPIX
E336755
GC-2 94V-0
UP TO 3W C.A.

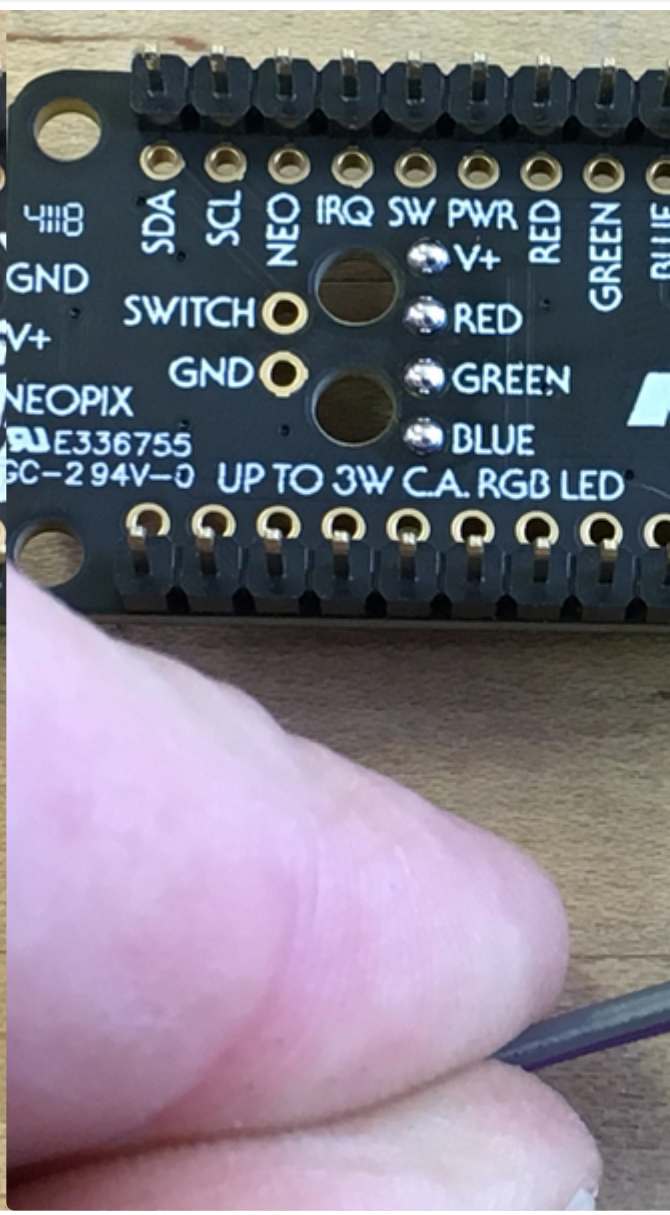

4-8Ω SPEAKER

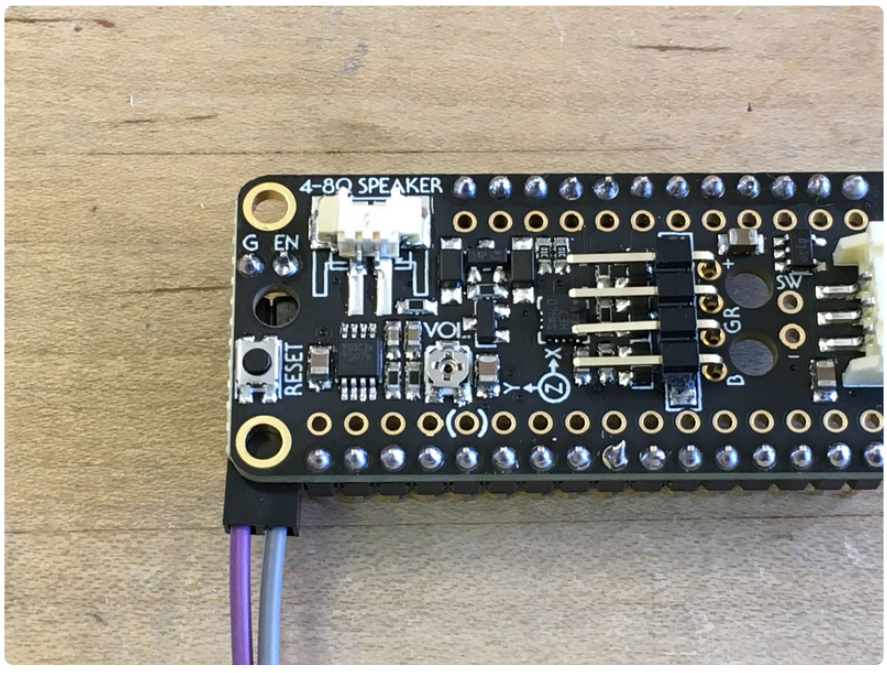
## On / Off Switch Header

We'll use this same technique to make a low profile connection for the On / Off switch to the En(able) and GND pins on the Prop-Maker Wing. We'll connect these to the underside of the board.



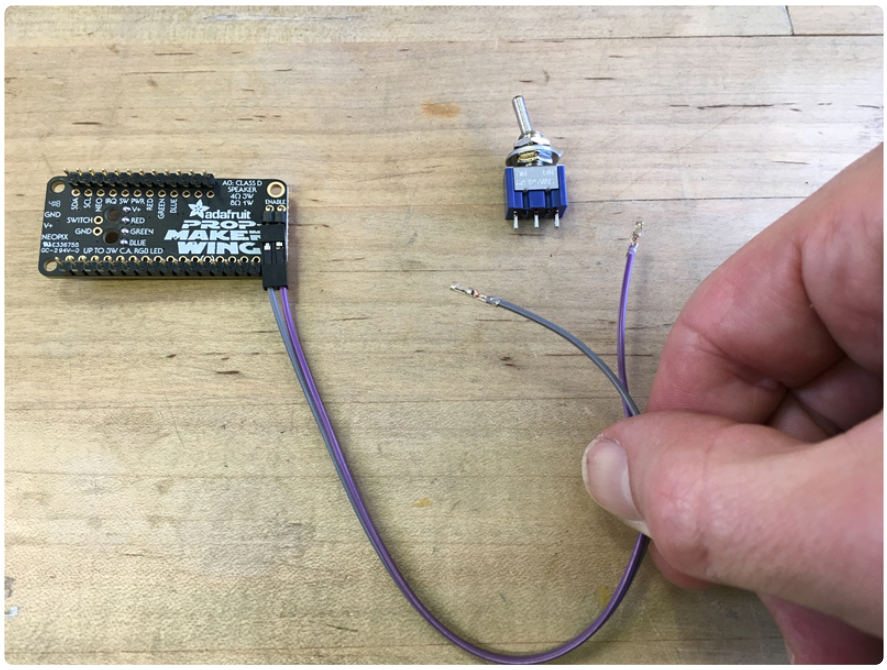

4118
SDA
SCL
NEO
IRQ
SW
PWR
RED
GREEN
BLUE
GND
V+
SWITCH
GND
V+
RED
GREEN
BLUE
NEOPIX
E336755
UP TO 3W C.A. RGB LED

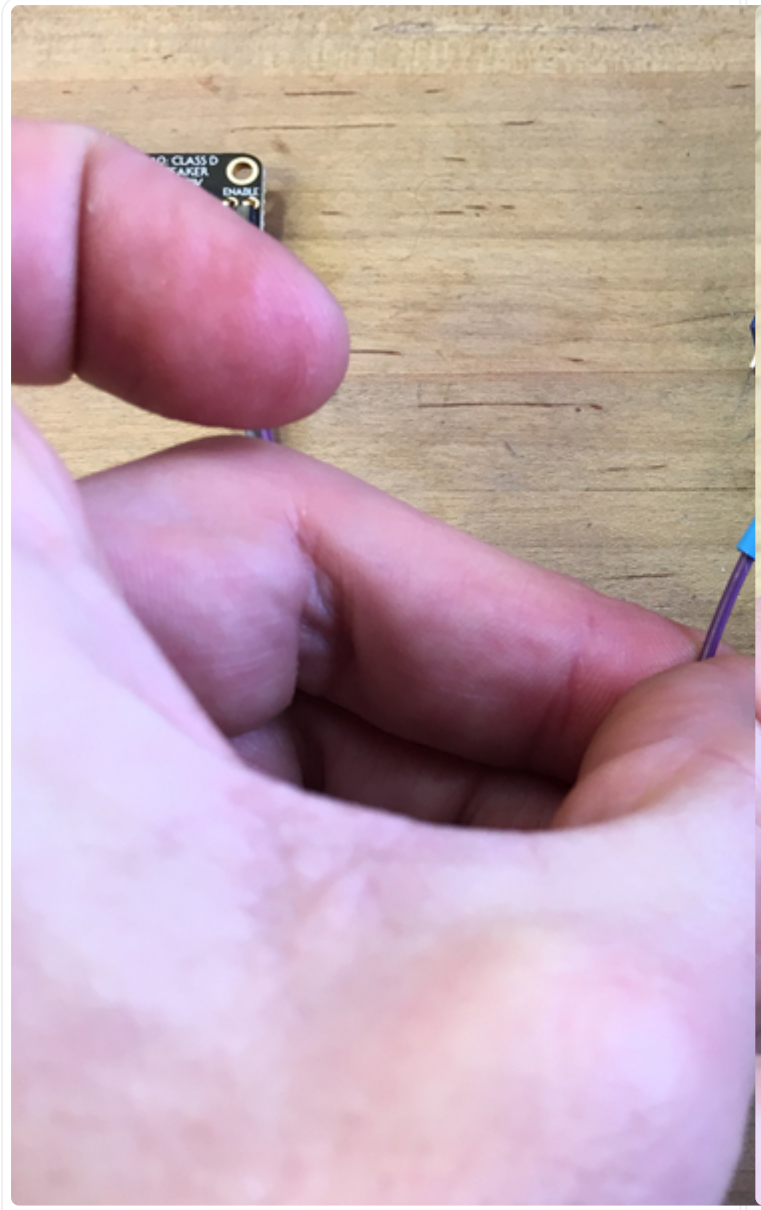

4118
SDA
SCL
NEO
IRQ
SW
PWR
RED
GREEN
GND
V+
SWITCH
GND
V+
RED
GREEN
BLUE
NEOPIX
E336755
UP TO 3W C.A. RGB LED

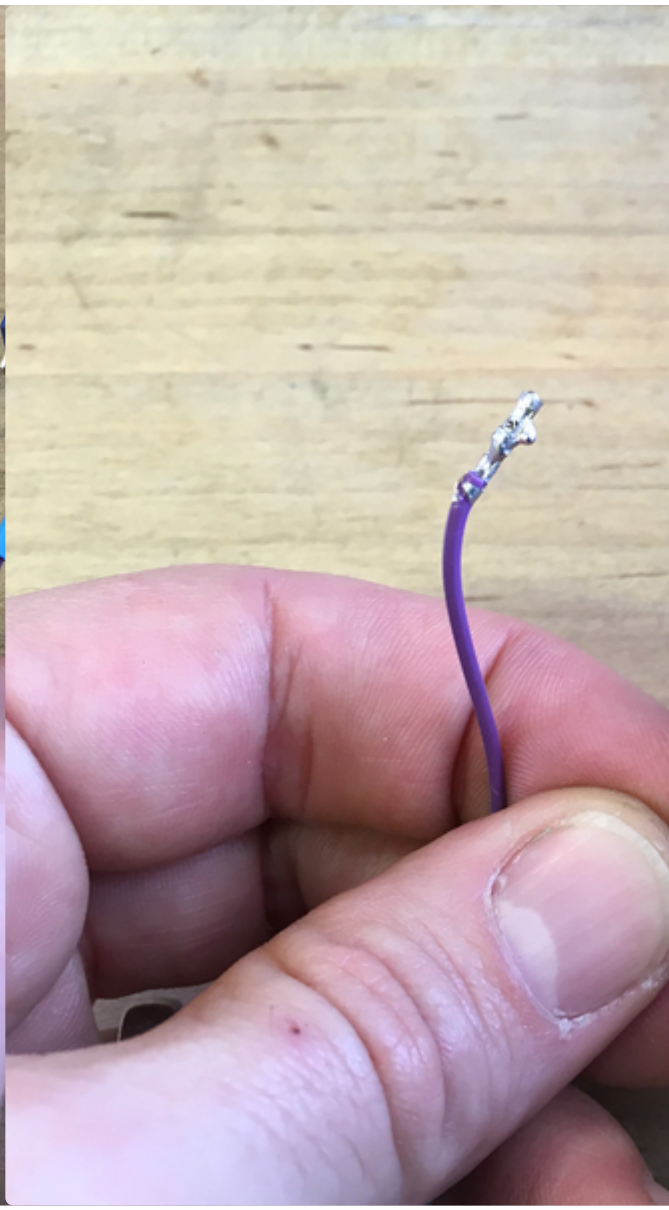

4-8Ω SPEAKER
G
EN
VOL
RESET
SW

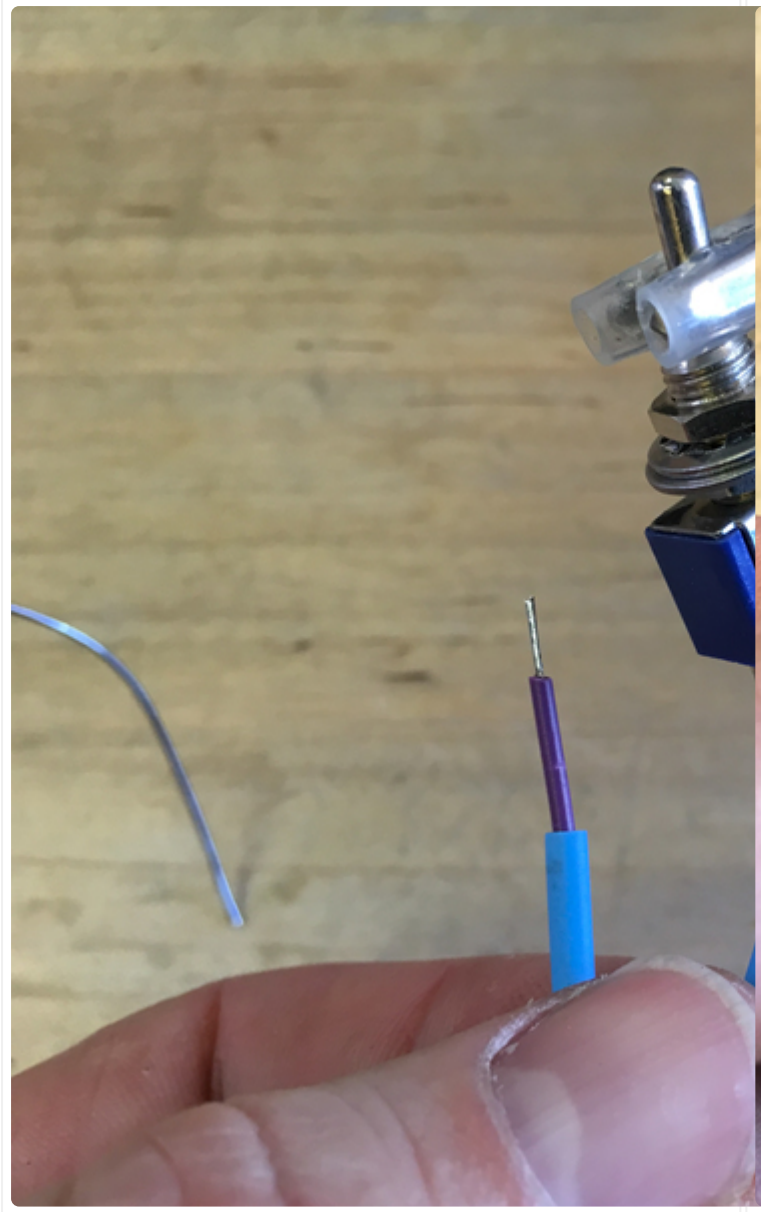
## Button Wiring

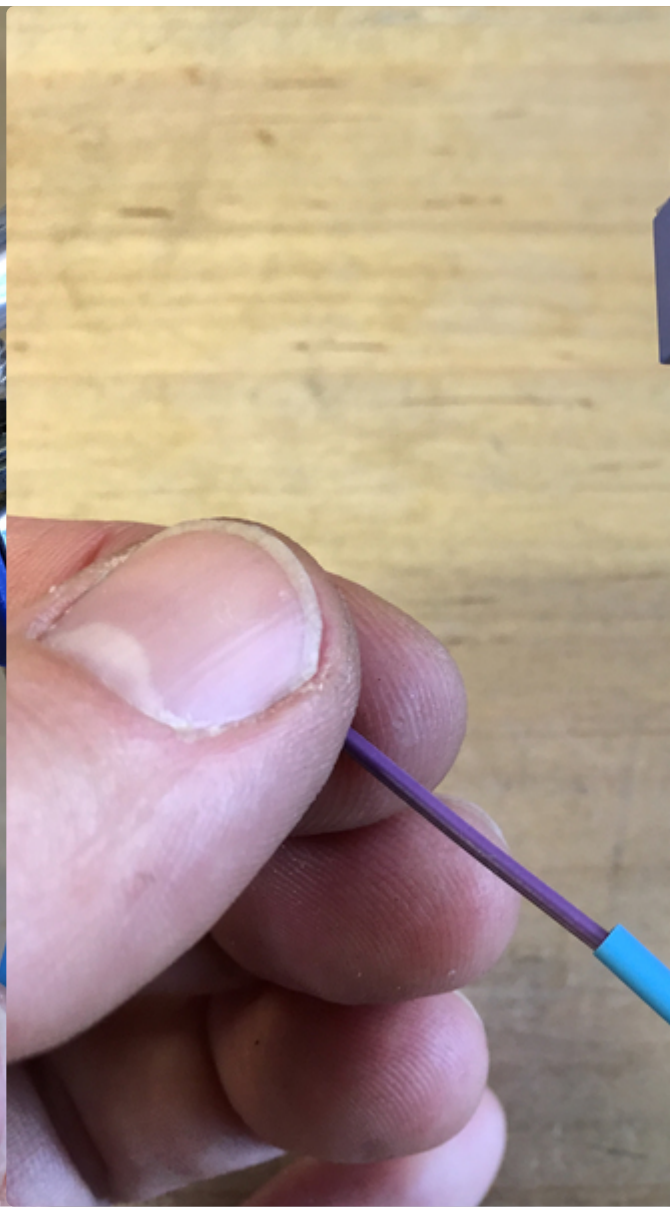
The toggle switch is a SPDT (single pole, dual throw) which means we can connect one wire to the center terminal and the other to either outside terminal. Cut, strip, and tin the two wires, then add some heat shrink tubing (optional), and solder the wires to the switch as shown.

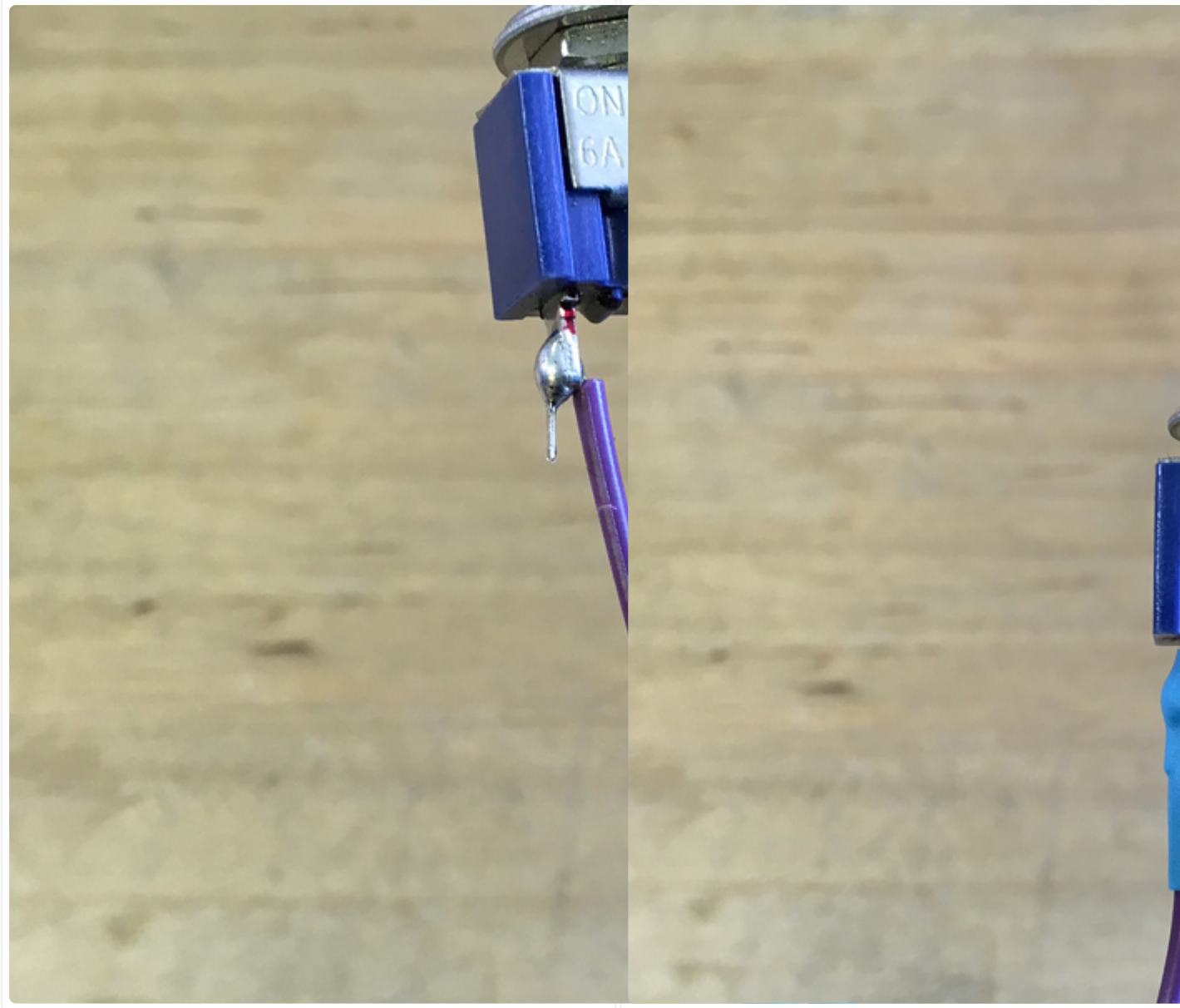






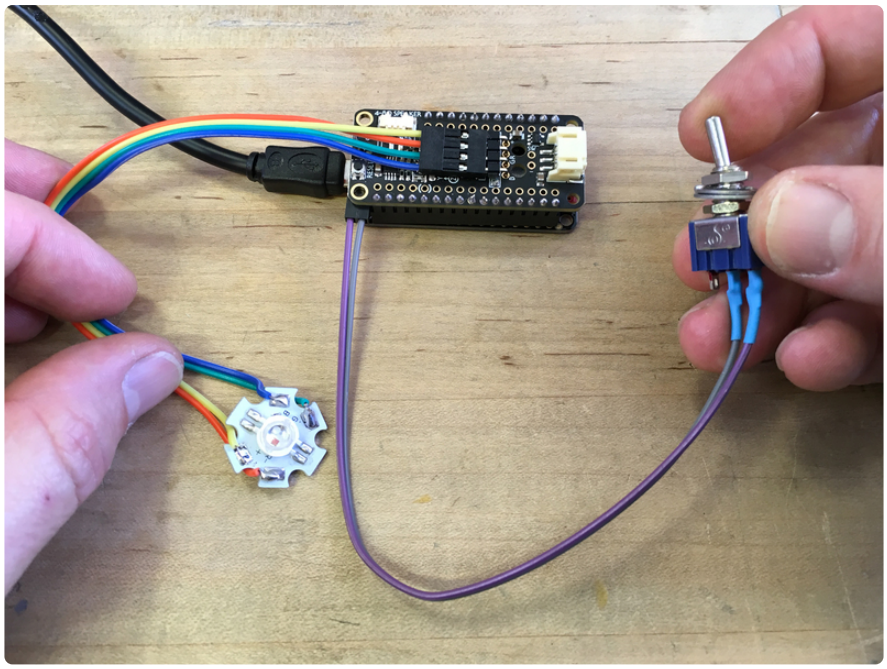







## Plug and Play

You can now plug in the LED cable and on/off switch to the Prop-Maker Wing, and then fit it onto the Feather M0 Bluefruit board.



Next, let's program the Feather!

---

# Code in Arduino

Here we'll code the board in Arduino to change colors using the Bluefruit app on iOS (https://adafru.it/f4H) and Android (https://adafru.it/f4G), as well as to react to tilting your MUNNY to change to a random color.

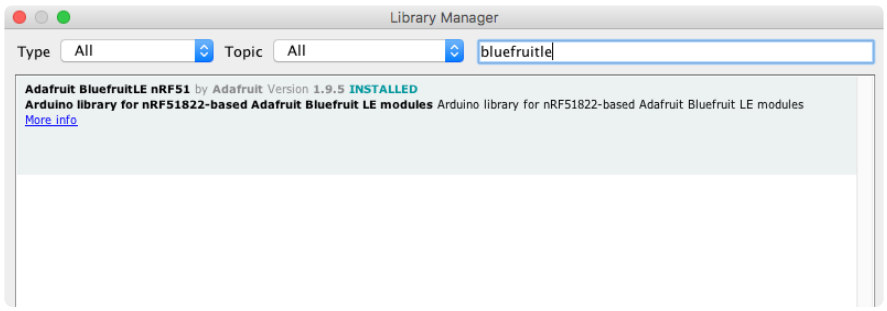
There's a lot of info to digest on the Feather M0 Bluefruit LE, so check out the guide here (https://adafru.it/CXe) if you have questions.

First, get set up with Arduino IDE as detailed here (https://adafru.it/BYk).

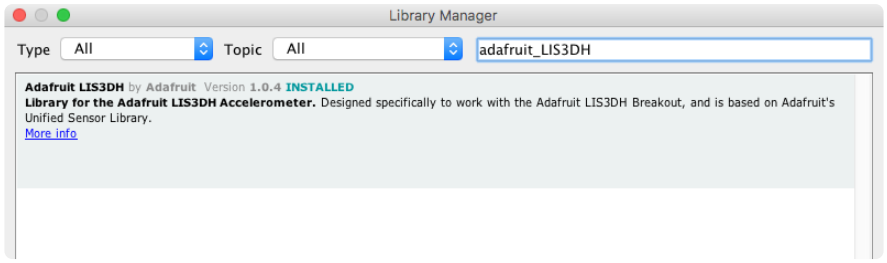
Next, follow these instructions (https://adafru.it/BYm) on getting the proper libraries installed. In this code we use the following libraries:

- **Adafruit_BLE**
- **Adafruit_Bluefruit_SPI**
- **Adafruit_BluefruitLE_UART**
- **Adafruit_LIS3DH**

The first three come from installing the **Adafruit BluefruitLE nRF51** library.




The other library to install is **Adafruit LIS3DH**, which is for the accelerometer built onto the Prop-Maker Wing.




Once you've updated the board definitions as shown, you'll be able to select **Adafruit Feather M0** as your board for compiling and uploading.

Before you continue, make sure you can plug in your Feather over USB and upload the Blink sketch found in Arduino IDE menu **Examples** > **01.Basics** > **Blink**

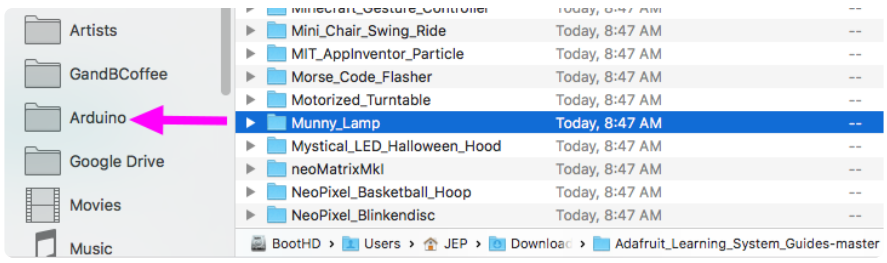
If everything is set up properly, the Feather should now be blinking the onboard LED every second.

Once that's working, try running the Bluefruit example sketch, as detailed here. (https://adafru.it/BYm)

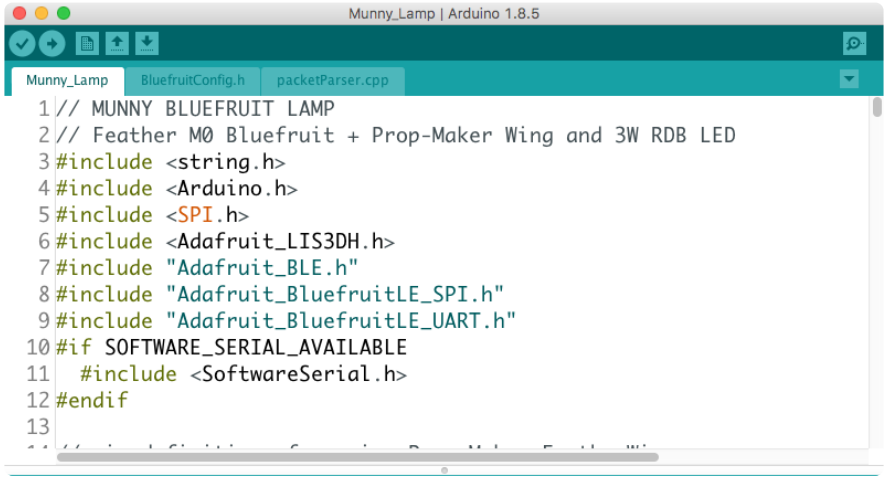
## MUNNY Code

Next, we'll get the **Munny_Lamp.ino** code to upload to the Feather. First, download the Adafruit_Learning_Systems_Guides repo here (https://adafru.it/CJ6).

You'll need to unzip the file and then navigate it to get to the **Munny_Lamp** directory. Move this directory it to your Arduino sketches directory.



Inside of the **Arduino IDE**, open the **Munny_Lamp.ino** sketch.




Note how Arduino automatically opens the associated **BluefruitConfig.h** file and the **packetParser.cpp** file. You won't need to worry about these, but they do need to be there for everything to function!

Upload the code to your Feather M0 Bluefruit LE now, so we can try out the remote color changing and tilt functions!

```
// SPDX-FileCopyrightText: 2018 John Edgar Park for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// MUNNY BLUEFRUIT LAMP
// Feather M0 Bluefruit + Prop-Maker Wing and 3W RDB LED
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include <Adafruit_LIS3DH.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"
#if SOFTWARE_SERIAL_AVAILABLE
  #include <SoftwareSerial.h>
#endif

// pin definitions for using Prop-Maker FeatherWing
//#define NEOPIXEL_PIN 5
//#define SWITCH_PIN   9
#define POWER_PIN    10
#define RED_LED      11
#define GREEN_LED    12
#define BLUE_LED     13
int red = 0;
int green = 0;
int blue = 0;
#include "BluefruitConfig.h"
Adafruit_LIS3DH lis = Adafruit_LIS3DH();

/*=========================================================================
    APPLICATION SETTINGS

    FACTORYRESET_ENABLE       Perform a factory reset when running this sketch

                              Enabling this will put your Bluefruit LE module
                              in a 'known good' state and clear any config
```

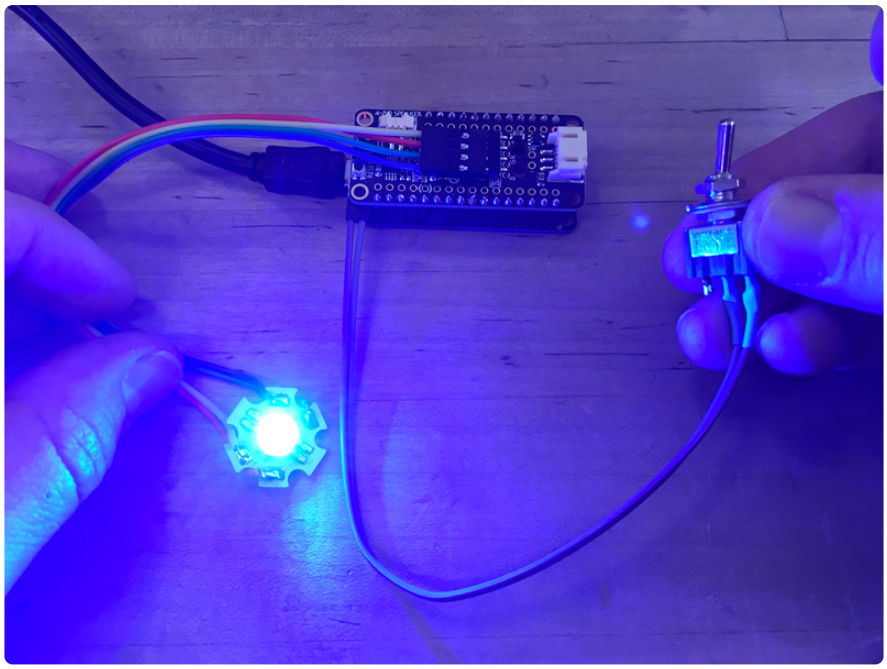
```
                            data set in previous sketches or projects, so
                            running this at least once is a good idea.

                            When deploying your project, however, you will
                            want to disable factory reset by setting this
                            value to 0.  If you are making changes to your
                            Bluefruit LE device via AT commands, and those
                            changes aren't persisting across resets, this
                            is the reason why.  Factory reset will erase
                            the non-volatile memory where config data is
                            stored, setting it back to factory default
                            values.

                            Some sketches that require you to bond to a
                            central device (HID mouse, keyboard, etc.)
                            won't work at all with this feature enabled
                            since the factory reset will clear all of the
                            bonding data stored on the chip, meaning the
                            central device won't be able to reconnect.
    PIN                     Which pin on the Arduino is connected to the
NeoPixels?
    NUMPIXELS               How many NeoPixels are attached to the Arduino?
    -----------------------------------------------------------------------*/
    #define FACTORYRESET_ENABLE     0
/*=========================================================================*/


// Create the bluefruit object, either software serial...uncomment these lines
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ,
BLUEFRUIT_SPI_RST);

// A small helper
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// function prototypes over in packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// the packet buffer
extern uint8_t packetbuffer[];


/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/
void setup(void)
{
  delay(500);
  pinMode(POWER_PIN, OUTPUT);
  digitalWrite(POWER_PIN, HIGH);
  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);

  analogWrite(RED_LED, 0);
  analogWrite(GREEN_LED, 0);
  analogWrite(BLUE_LED, 255);  // startup color, waiting for BLE connection

  if (! lis.begin(0x18)) {   // change this to 0x19 for alternative i2c address
    Serial.println("Couldnt start LIS3DH");
    while (1);
  }
```

```
  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit MUNNY LED Color Picker"));
  Serial.println(F("------------------------------------------------"));

  /* Initialise the module */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check
wiring?"));
  }
  Serial.println( F("OK!") );

  if ( FACTORYRESET_ENABLE )
  {
    /* Perform a factory reset to make sure everything is in a known state */
    Serial.println(F("Performing a factory reset: "));
    if ( ! ble.factoryReset() ){
      error(F("Couldn't factory reset"));
    }
  }

  /* Disable command echo from Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in Controller
mode"));
  Serial.println(F("Then activate/use the sensors, color picker, game controller,
etc!"));
  Serial.println();

  ble.verbose(false);  // debug info is a little annoying after this point!

  /* Wait for connection */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("**********************"));

  // Set Bluefruit to DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("**********************"));

}

/**************************************************************************/
/*!
    @brief  Constantly poll for new command or response data
*/
/**************************************************************************/
void loop(void)
{
  digitalWrite(POWER_PIN, HIGH);

  /* Wait for new data to arrive */
  uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
  if (len == 0) {
    accelerometer_check();
    return;
    delay(10);
```

```
  }

  /* Got a packet! */
  // printHex(packetbuffer, len);

  // Color
  if (packetbuffer[1] == 'C') {
    uint8_t red = packetbuffer[2];
    uint8_t green = packetbuffer[3];
    uint8_t blue = packetbuffer[4];
    Serial.print ("RGB #");
    if (red < 0x10) Serial.print("0");
    Serial.print(red, HEX);
    if (green < 0x10) Serial.print("0");
    Serial.print(green, HEX);
    if (blue < 0x10) Serial.print("0");
    Serial.println(blue, HEX);

    analogWrite(RED_LED, red);
    analogWrite(GREEN_LED, green);
    analogWrite(BLUE_LED, blue);
  }

}

void accelerometer_check() {
  // Accelerometer
  sensors_event_t event;
  lis.getEvent(&event);
  Serial.print("\t\tX: "); Serial.print(event.acceleration.x);
  Serial.print(" \tY: "); Serial.print(event.acceleration.y);
  Serial.print(" \tZ: "); Serial.print(event.acceleration.z);
  Serial.println(" m/s^2 ");

  if (event.acceleration.y < 0) {
    analogWrite(RED_LED, random(0, 255));
    analogWrite(GREEN_LED, random(0, 255));
    analogWrite(BLUE_LED, random(0, 255));
    Serial.println("TILTED");
    delay(100);
  }
}
```

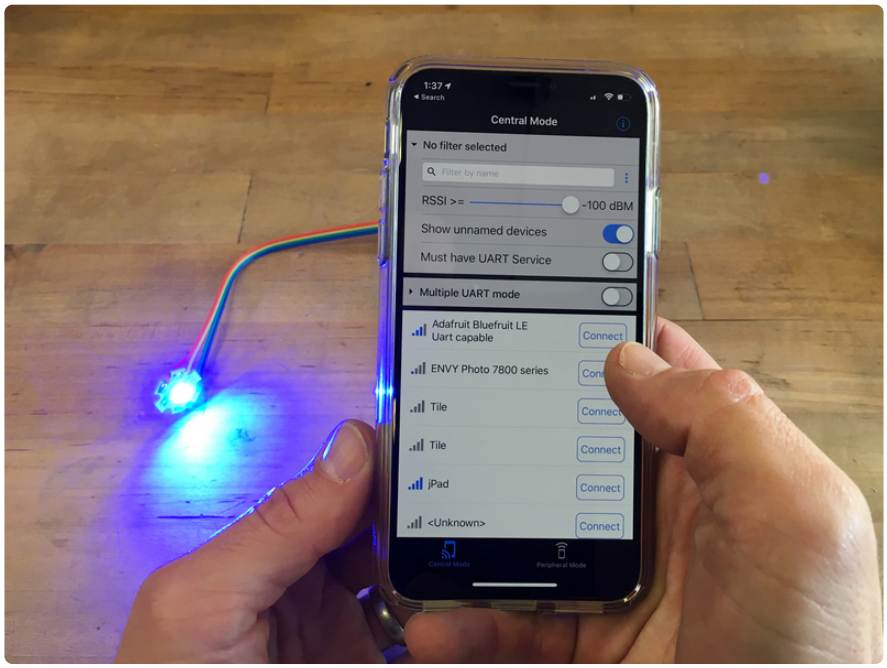
## Testing

The LED will light up blue upon startup. The Feather is now waiting for you to connect to it with the Bluefruit app.



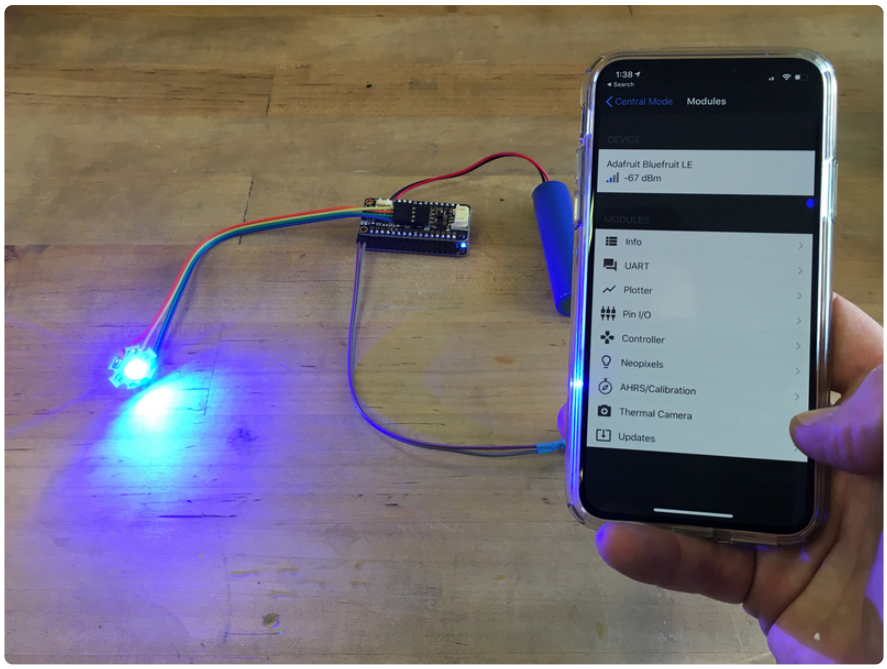
## Connect

Launch the app and after a moment you'll see a list of BLE devices to which you can connect. Click the '**Connect**' button next to the **Adafruit Bluefruit LE** device.



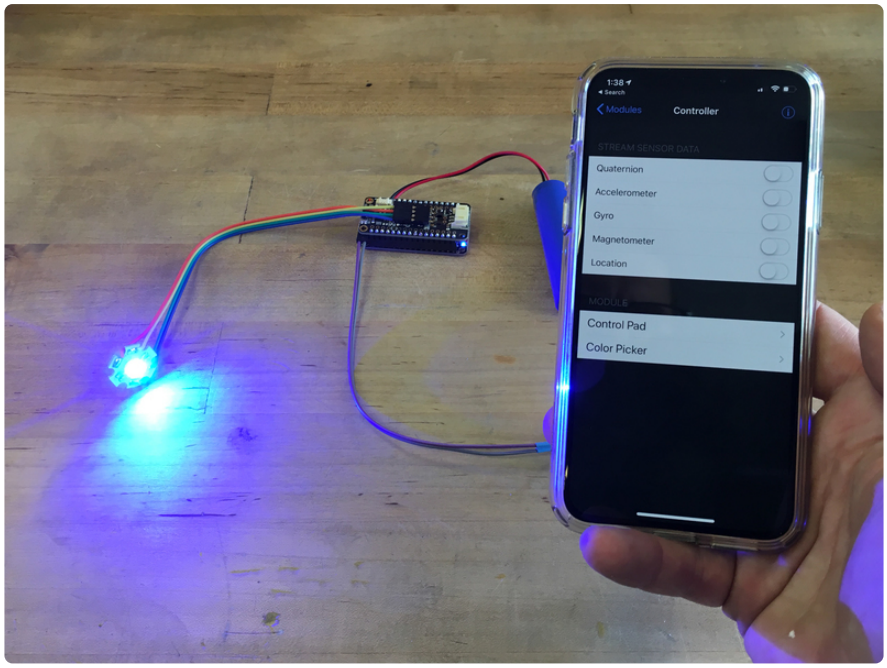

## Controller

Now that you're connected, you can click on the '**Controller**' module.



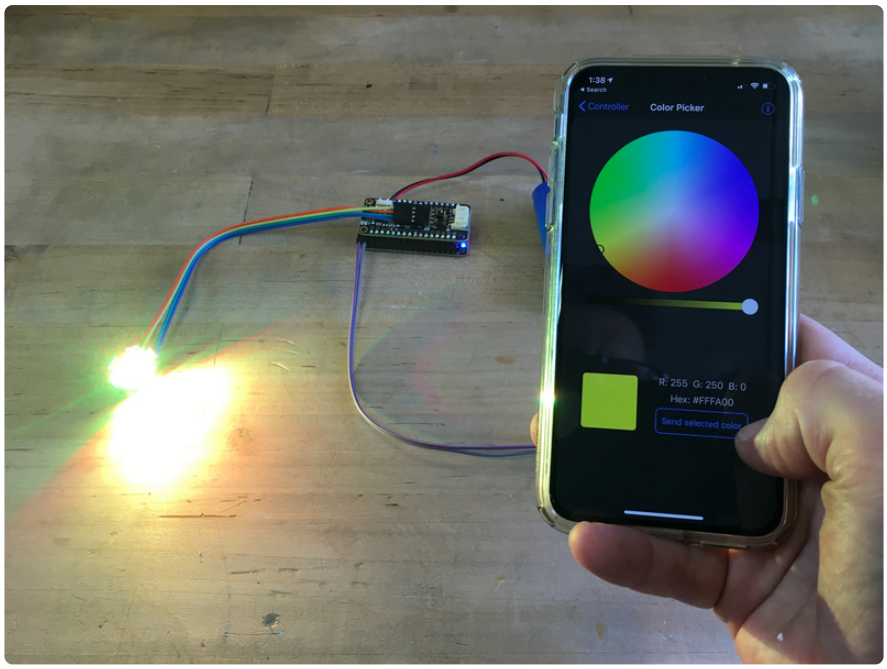

## Color Picker

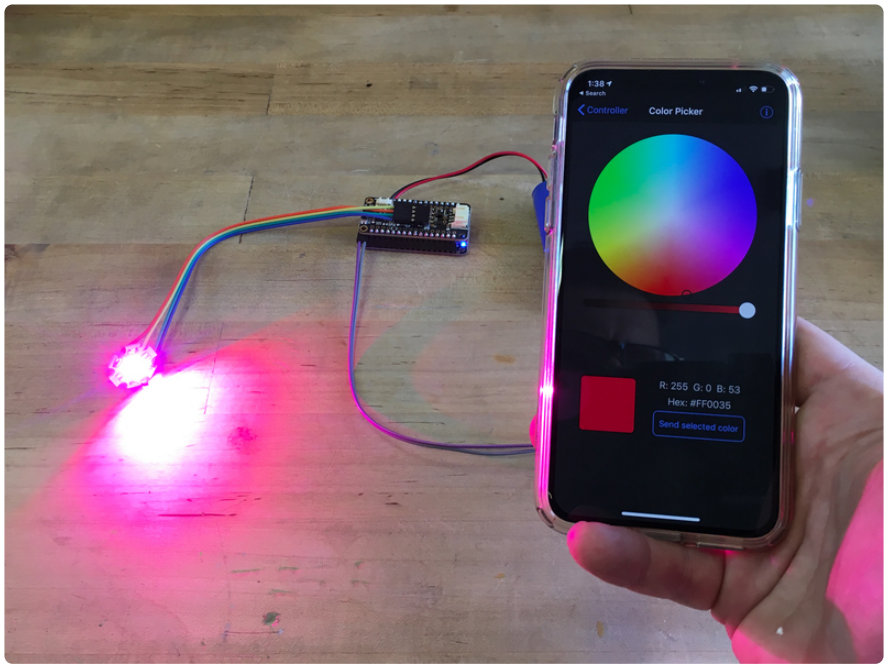
There are a number of choices in the **Controller** module, we'll use the '**Color Picker**'.




## Color Wheel

Now, you can pick on any color in the color wheel! Press '**Send selected color**' when you want to change the RGB LED color wirelessly through the air!!




Note, you can also adjust the brightness by using the slider just below the color wheel.

---

# Code with CircuitPython

You can also code your glowing friend with CircuitPython!

CircuitPython (https://adafru.it/tB7) is designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping

by requiring no upfront desktop software downloads. Simply copy and edit files on the `CIRCUITPY` drive to iterate.

Are you new to using CircuitPython? No worries, there is a full getting started guide here (https://adafru.it/cpy-welcome).

Adafruit suggests using the Mu editor to edit your code and have an interactive REPL in CircuitPython. You can learn about Mu and its installation in this tutorial (https://adafru.it/ANO).

Follow this guide for instructions on installing the latest release version of CircuitPython for the **Feather M0 Adalogger**.

Yes, even though we're using the Feather M0 Bluefruit LE, the .uf2 file we're using is that for the Feather M0 Adalogger!

**CircuitPython Release**

https://adafru.it/Amd

## Libraries

You'll also need to add the following libraries for this project. Follow this guide (https://adafru.it/Cqa) on adding libraries. The ones you'll need are:

- **adafruit_bus_device**
- **adafruit_lis3dh**

Download the latest adafruit-circuitpython-bundle .zip file as linked in the guide. Then, unzip the file and drag those libraries to the **lib** folder on your Feather.

## Code

Here is the code we'll use. Copy it and then paste in Mu. Save it to your Feather as **code.py**

```
# SPDX-FileCopyrightText: 2018 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# RGB Color Picker demo - wire up RGB LEDs and set their color
# using Adafruit Bluefruit Connect App on your phone
# runs on Feather M0 Bluefruit LE running the Feather M0 Adalogger build
# of CircuitPython with Prop-Maker Wing and 3W RGB LED
```

```
import time
import random
import board
import busio
import pwmio
from digitalio import DigitalInOut, Direction
from adafruit_bluefruitspi import BluefruitSPI
import adafruit_lis3dh

ADVERT_NAME = b'BlinkaNeoLamp'

# RGB LED on D11, 12, 13, we're using a Prop Maker wing
red_led = pwmio.PWMOut(board.D11, frequency=50000, duty_cycle=0)
green_led = pwmio.PWMOut(board.D12, frequency=50000, duty_cycle=0)
blue_led = pwmio.PWMOut(board.D13, frequency=50000, duty_cycle=0)
# Prop maker wing has a power pin for the LED!
power_pin = DigitalInOut(board.D10)
power_pin.direction = Direction.OUTPUT
power_pin.value = True

spi_bus = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs = DigitalInOut(board.D8)
irq = DigitalInOut(board.D7)
rst = DigitalInOut(board.D4)
bluefruit = BluefruitSPI(spi_bus, cs, irq, rst, debug=False)

i2c = busio.I2C(board.SCL, board.SDA)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c)

def init_bluefruit():
    # Initialize the device and perform a factory reset
    print("Initializing the Bluefruit LE SPI Friend module")
    bluefruit.init()
    bluefruit.command_check_OK(b'AT+FACTORYRESET', delay=1)
    # Print the response to 'ATI' (info request) as a string
    print(str(bluefruit.command_check_OK(b'ATI'), 'utf-8'))
    # Change advertised name
    bluefruit.command_check_OK(b'AT+GAPDEVNAME='+ADVERT_NAME)

def wait_for_connection():
    print("Waiting for a connection to Bluefruit LE Connect ...")
    # Wait for a connection ...
    dotcount = 0
    while not bluefruit.connected:
        print(".", end="")
        dotcount = (dotcount + 1) % 80
        if dotcount == 79:
            print("")
        time.sleep(0.5)

# This code will check the connection but only query the module if it has been
# at least 'n_sec' seconds. Otherwise it 'caches' the response, to keep from
# hogging the Bluefruit connection with constant queries
connection_timestamp = None
is_connected = None
def check_connection(n_sec):
    # pylint: disable=global-statement
    global connection_timestamp, is_connected
    if (not connection_timestamp) or (time.monotonic() - connection_timestamp >
n_sec):
        connection_timestamp = time.monotonic()
        is_connected = bluefruit.connected
    return is_connected

# Unlike most circuitpython code, this runs in two loops
# one outer loop manages reconnecting bluetooth if we lose connection
# then one inner loop for doing what we want when connected!
while True:
```

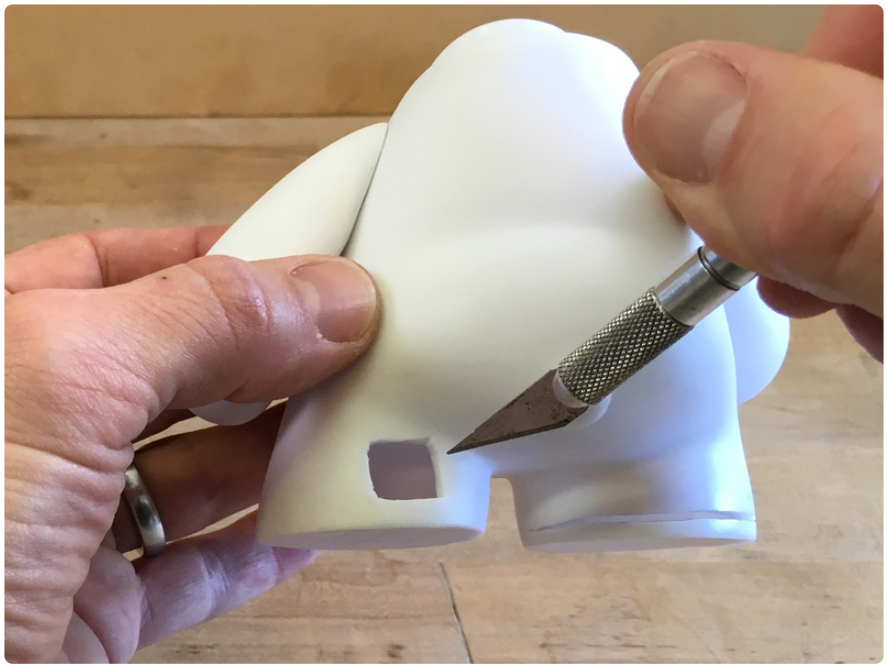
```
    # Initialize the module
    init_bluefruit()
    try:        # Wireless connections can have corrupt data or other runtime
failures
                # This try block will reset the module if that happens
        while True:
            # Once connected, check for incoming BLE UART data
            if check_connection(3):  # Check our connection status every 3 seconds
                # OK we're still connected, see if we have any data waiting
                resp = bluefruit.read_packet()
                if not resp:
                    continue  # nothin'
                print("Read packet", resp)
                # Look for a 'C'olor packet
                if resp[0] != 'C':
                    continue
                # Set the LEDs to the three bytes in the packet
                red_led.duty_cycle = int(resp[1]/255 * 65535)
                green_led.duty_cycle = int(resp[2]/255 * 65535)
                blue_led.duty_cycle = int(resp[3]/255 * 65535)
            else:  # Not connected
                # print(lis3dh.acceleration)
                if lis3dh.acceleration.y < -5:
                    print("Tilted")
                    red_led.duty_cycle = random.randint(0, 65535)
                    green_led.duty_cycle = random.randint(0, 65535)
                    blue_led.duty_cycle = random.randint(0, 65535)
                    time.sleep(0.25)

    except RuntimeError as e:
        print(e)  # Print what happened
        continue  # retry!
```

## Testing

Once you've saved the code.py file to the board, the LED will light up! For info on testing it with the Bluefruit App, head back to the previous page in this guide, to the section on Testing.

# Mod Your MUNNY


Now we've got to stuff the electronics into MUNNY! This procedure will vary depending on the vinyl figure you have. You could mount the Feather like a cool backpack, and run the LED inside the body, for example. Or you can try to get everything inside as shown here. One member of our community even suggested using a reed switch and magnet to enable hidden interface options!

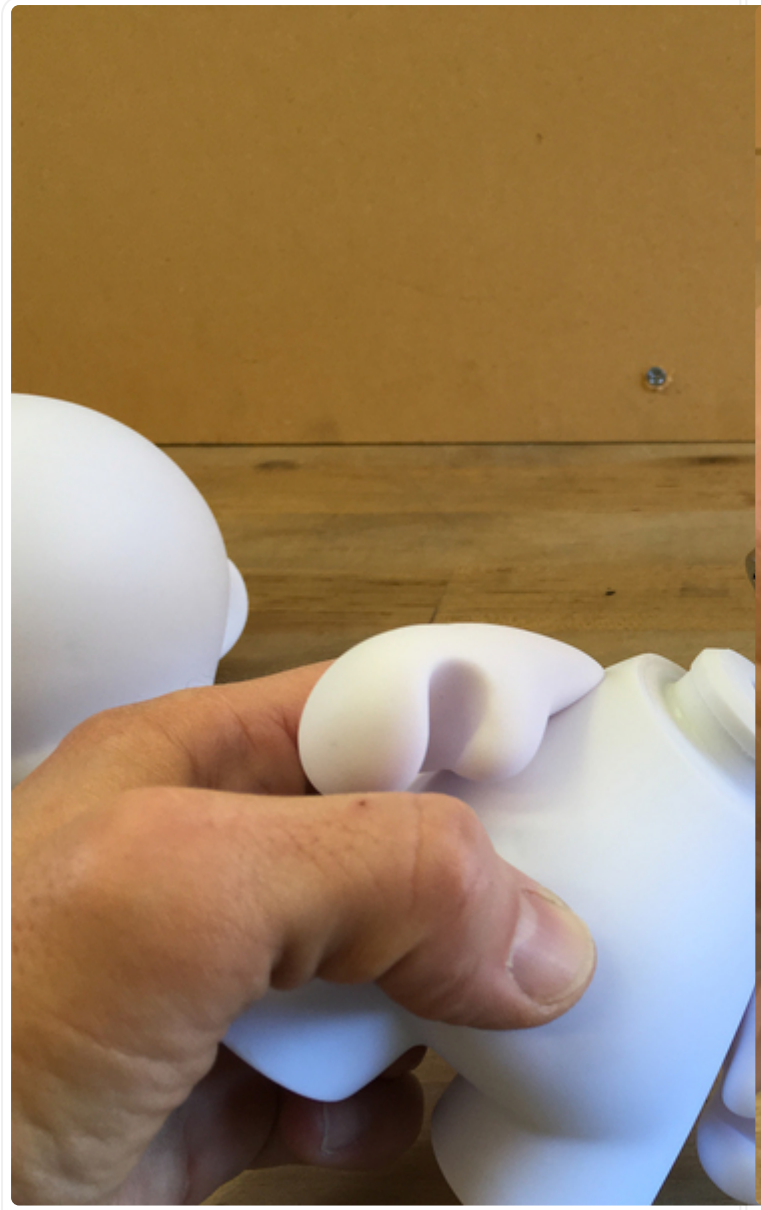


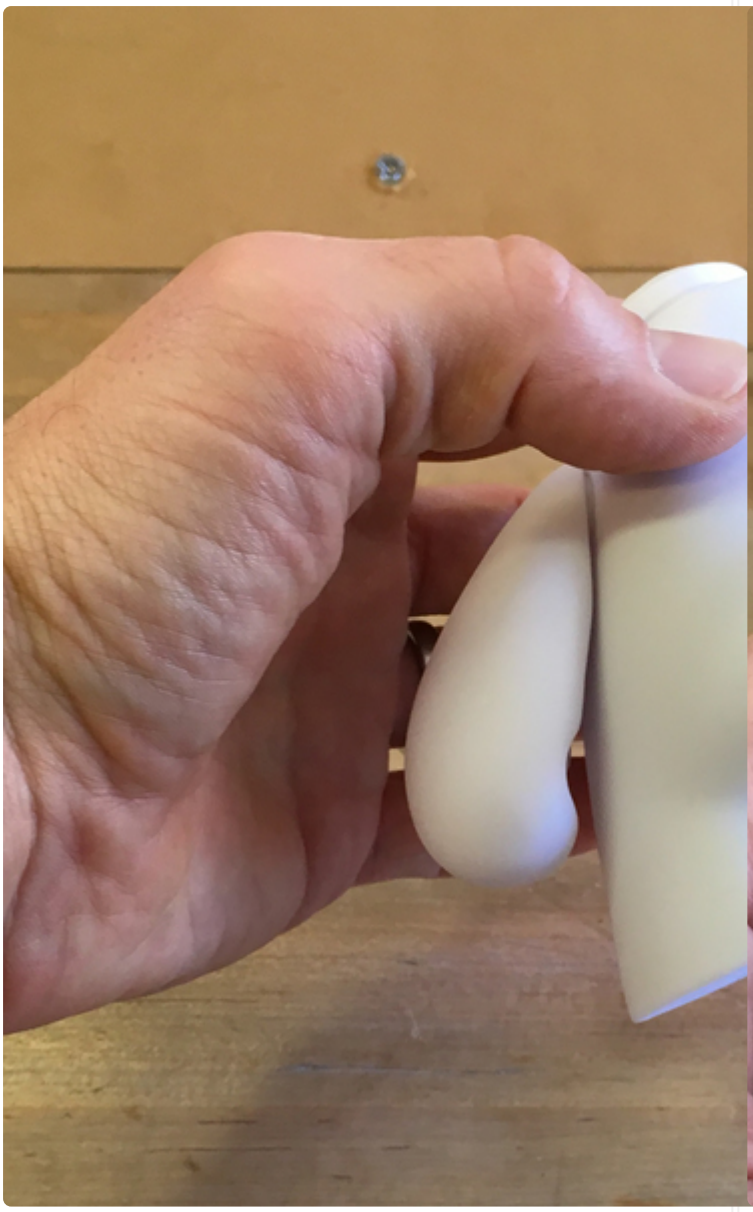
These are the mods I did:

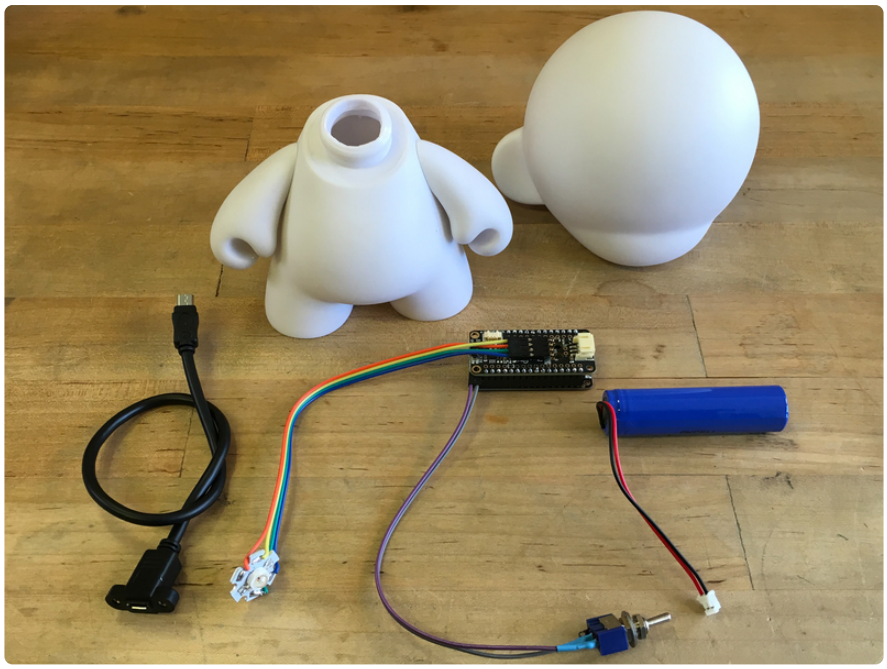
- Remove the head

- Heat the neck with a heatgun, then slice the center open with a hobby knife
- Repeat for the neck socket in the head
- Head and cut open a square in the left leg to insert the USB panel mount
- Heat and slice a flap opening into the right foot to insert the electronics

Please be careful with sharp or hot instruments.



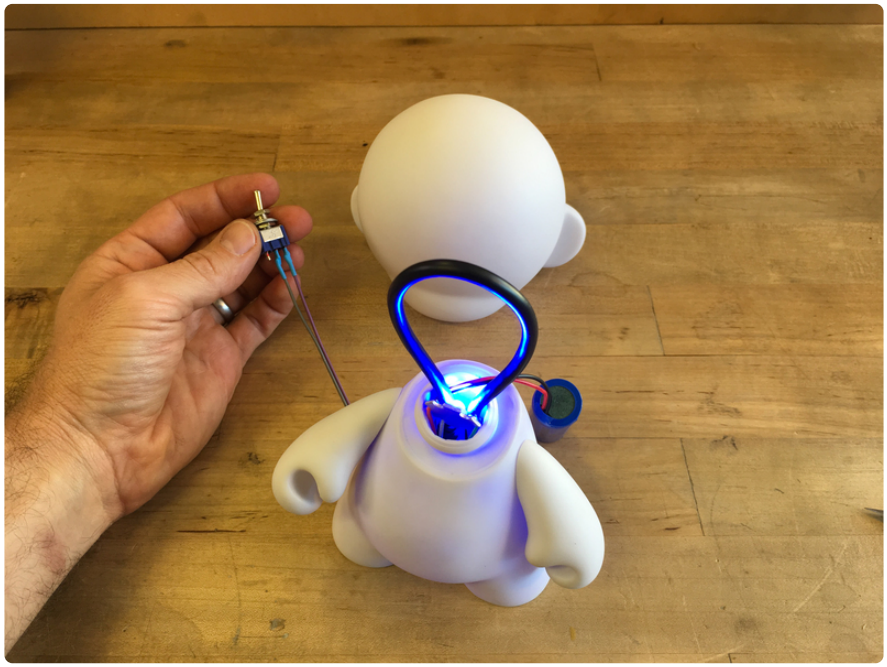




## Insert Electronics

It's a bit like building a ship in a bottle, but now you will insert the battery into the head, the panel mount through the leg, and connect and feed in the Feather and LED assembly! The video slide show will give you a good idea how to proceed.

Before putting the head back on, it's a good idea to test that everything is working!



## Action

Now you can feed the USB cable and battery into MUNNY's head and close it up. I decided to have him hold onto his own on/off switch.

Turn on the switch, and then connect to the Feather using the Adafruit Bluefruit app. You can change his colors any time you like!

## Tilt Action

MUNNY also has a secret mode! You can tip him upside down to randomize his color!!

This uses the accelerometer built right onto the Prop-Maker Wing to detect tilt, and it then picks random color values every half second.

Tip him back upright to stop the madness.

## Recharge

You can bring your little glowing friend with you and use the battery for power. When it's time to recharge, simply plug in the USB power supply into his heel port!