Magic Wand

Created by Alex Alves

https://learn.adafruit.com/magic-wand

Last updated on 2024-06-03 02:05:14 PM EDT

Table of Contents

Overview

This project turns a toy hammer into a magic wand that produces different sound and light effects depending on the spell cast based on simple gesture recognition. The amount of components in this project combined with the small room requites a bit of cramfu to get all the parts to fit. The compact nature of this project and the number of components means it's a best for a somewhat experienced maker.

Adafruit Audio FX Sound Board

https://adafru.it/pqa

Adafruit Feather 32u4 Basic Proto

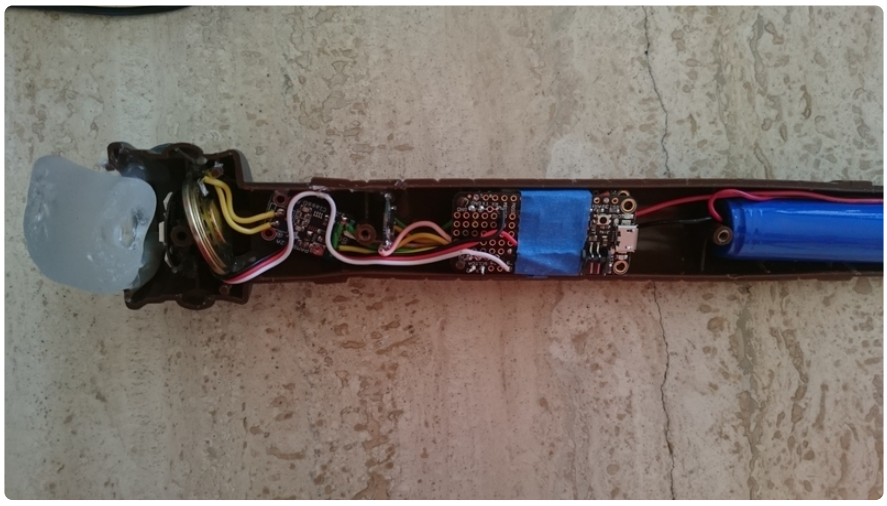
https://adafru.it/keQ

Adafruit NeoPixel Überguide

https://adafru.it/ja1

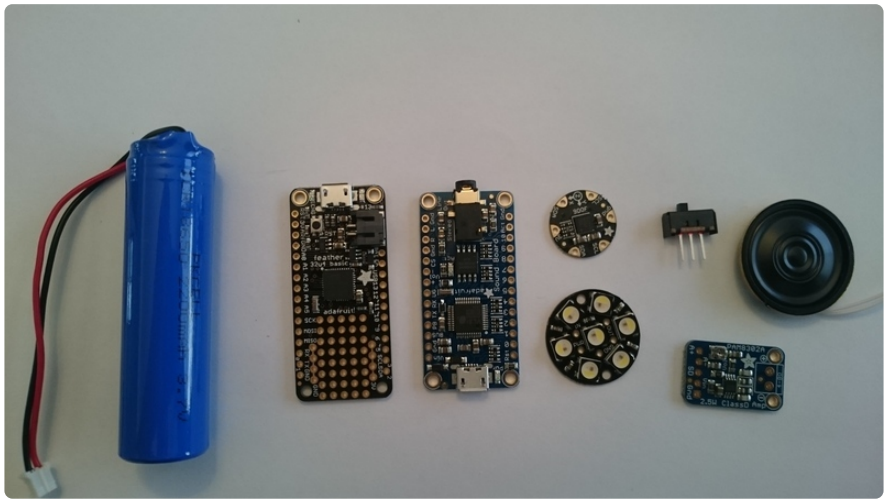
Adafruit LSM9DS0 Accelerometer + Gyro + Magnetometer 9-DOF Breakouts

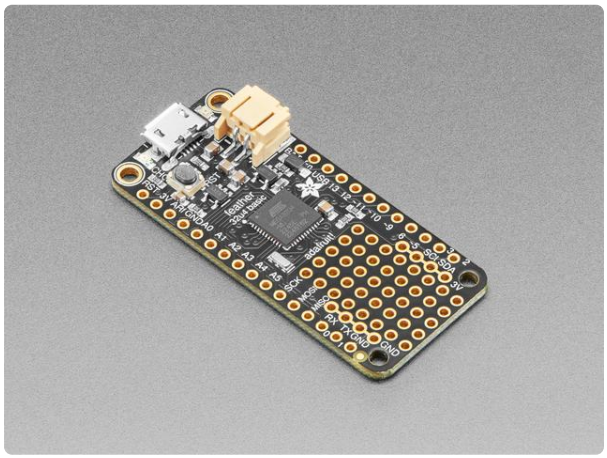
https://adafru.it/dN8



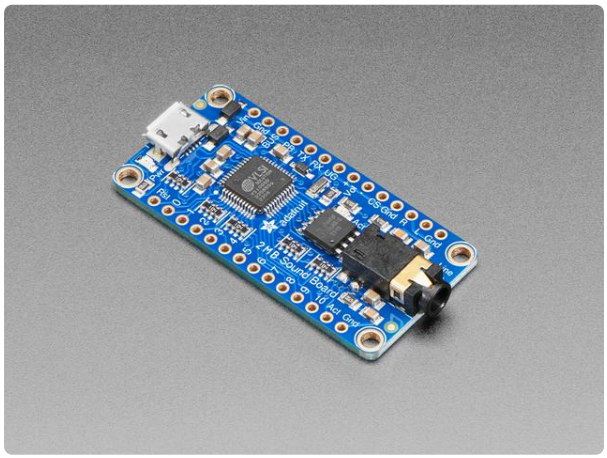
Parts/Tools

Parts



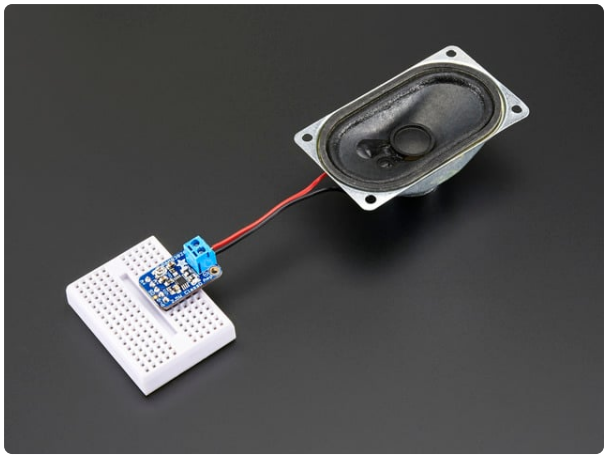


Adafruit Feather 32u4 Basic Proto
Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable...
https://www.adafruit.com/product/2771

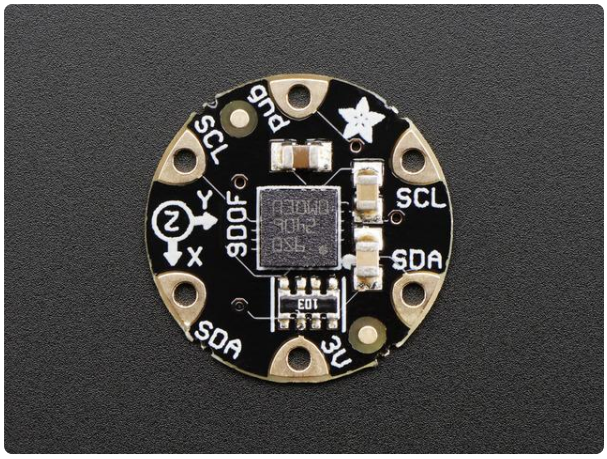


Adafruit Audio FX Sound Board - WAV/OGG Trigger with 2MB Flash
Would you like to add audio/sound effects to your next project, without an Arduino+Shield? Or maybe you don't even know how to use...
https://www.adafruit.com/product/2133

*Note the Audio FX board can be swapped out with any of the series, the Audio FX Mini (http://adafru.it/2342) and the Music Maker FeatherWing (http://adafru.it/3357) would help make a more compact and streamlined setup



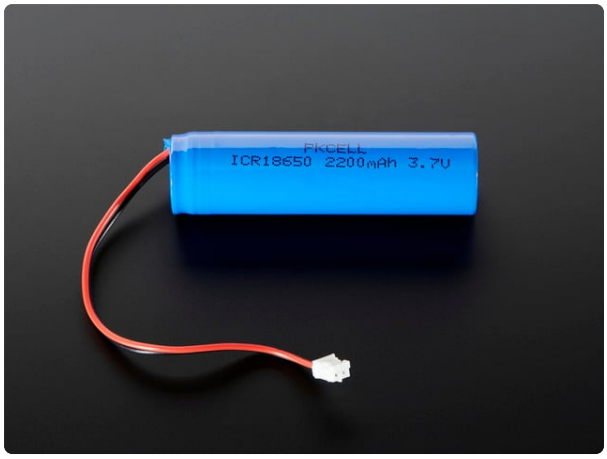
[Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302]
This super small mono amplifier is surprisingly powerful - able to deliver up to 2.5 Watts into 4-8 ohm impedance speakers. Inside the miniature chip is a class D controller, able to...
https://www.adafruit.com/product/2130



[FLORA 9-DOF Accelerometer/Gyroscope/ Magnetometer - LSM9DS0]
Add motion, direction and orientation sensing to your wearable FLORA project with this high precision 9-DOF sensors. Inside are three sensors, one is a classic 3-axis...
https://www.adafruit.com/product/2020

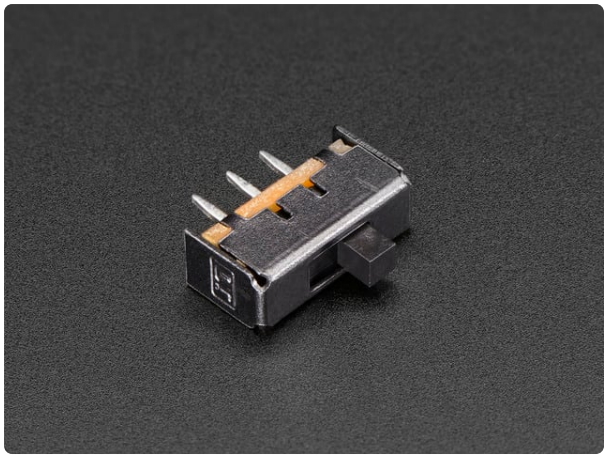
[NeoPixel Jewel - 7 x 5050 RGBW LED w/ Integrated Drivers]
Be the belle of the ball with the NeoPixel Jewel! These NeoPixel Jewels now have 4 LEDs in them (red, green, blue, and white) for excellent lighting...
https://www.adafruit.com/product/2858

[Mini Metal Speaker w/ Wires - 8 ohm 0.5W]
Listen up! This tiny 1" diameter speaker cone is the perfect addition to any small audio project where you need an 8 Ω impedance and will be using 0.5W or less of power. We...
https://www.adafruit.com/product/1890



Lithium Ion Cylindrical Battery - 3.7v 2200mAh

Need a big battery for your project? This lithium-ion battery contains a 2200mAh and a protection circuit that provides over-voltage, under-voltage, and over-current protection. Yet,...
https://www.adafruit.com/product/1781



Breadboard-friendly SPDT Slide Switch

These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They're easy to switch...
https://www.adafruit.com/product/805

Sacrificial Toy

I just looked around a local toy shop and came up with this hammer

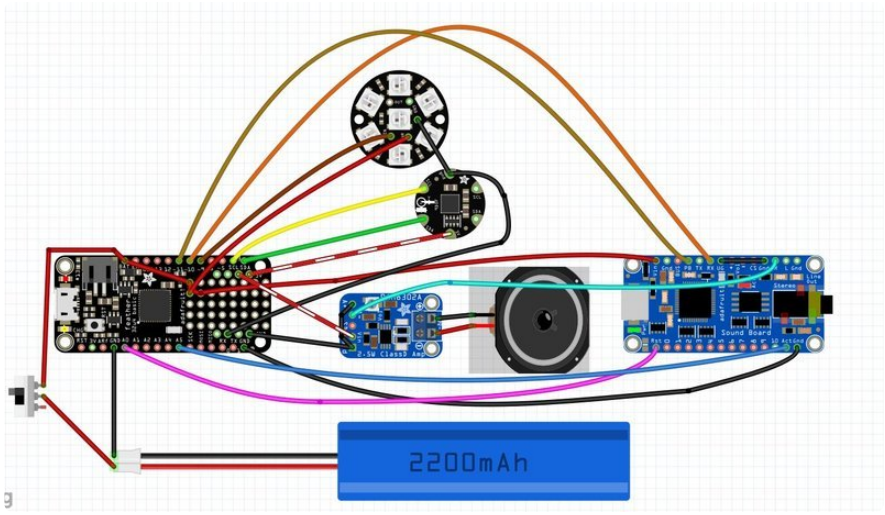
Crystal
I used frosted glass decorative filler for vases

Tools

- Soldering Iron
- Hot glue gun
- Rotary tool
- Sandpaper
- Helping hands
- Wire strippers
- Flush cutters
- Wire

Wiring Diagram

There is a lot of wiring in this so it's best to take it component at a time. Also the exact pins used will depend on the exact placement of parts so keep that in mind as you are prototyping, as neater wiring helps with compact builds. We will use the extra GND and 3.3V pads on the feather's prototyping area. Additionally 3 parts will be connected to the battery directly the Feather, Audio FX board and NeoPixel Jewel. I found that it was easiest to connect all the direct battery connections using the prototype area on the Feather.



For the wiring diagram the black wires are all ground. The solid red is connections to the battery where the dashed red is connected to the regulated 3.3V.

MagicWand.fzz

https://adafru.it/uFO

Wiring Conections

Audio FX board

We will actually use the JST connection at the bottom of this for the battery, meaning Vin on the soundboard will need to be attached to the BAT pin on the feather.

RST	Feather A0
ACT	Feather A5
RX	Feather 10
TX	Feather 11
UG	GND
Vin	Feather Bat
GND	GND
JST -	Battery -
JST +	Battery + through SPDT Switch

Audio Amp

A-	Audio FX R (or L)
A+	GND
Vin	Feather 3.3V
GND	GND
+/-	Speaker

9 DOF IMU

GND	GND
3.3V	Feather 3.3V
SCL	Feather SCL
SDA	Feather SDA

NeoPixel Jewel

GND	GND
PWR	Feather Bat
In	Feather 9

Software

Audio FX board:

First we need to transfer sound files to the Audio FX board, following this guide (https://adafru.it/pqc). The sound files I used are included in a zip file. You can find/ make your own sound effects and add them to the Audio FX board, just make sure you follow the proper format outlined here (https://adafru.it/uCW). This project controls the Audio FX board over serial, so the naming conventions aren't as important, and adding new sounds doesn't involve extra wires.

magicSound.zip

https://adafru.it/uFP

Arduino IDE Set Up:

For interacting with the feather you'll have to have to set up the Arduino IDE to use the Feather. Additionally you'll need to download and install libraries for the 9DOF IMU, Adafruit Sensor, NeoPixels, Audio FX.

Adafruit Feather 32u4 Basic Proto

https://adafru.it/uCY

Adafruit Audio FX Sound Board

https://adafru.it/uCZ

Adafruit NeoPixel Überguide

https://adafru.it/nBF

Adafruit LSM9DS0

https://adafru.it/dNP

Code:

Due to differences in the placement of the IMU, hand placement, and personal preference there is a set of sensor settings defined early in the code. Each of these variables help detect simple parts of the gesture recognition, and are independent. For example FLICKTRIG sets how hard you have to flick your wrist for it to register as a flick action.

The code only has 4 spells currently, but is designed for expandability. Spells are set of by achieving multiple simple condition triggers. For example a fire blast is a wrist flick while horizontal, but doing a wrist flick vertically will summon lightning. Currently there are 3 orientation states, and it can detect flicks and wrist twist. But adding something like a push forward action would allow 3 extra spells with only a few lines of code.

The code includes some useful settings to help debug, first by setting HARDWARETEST to true the wand will play music and test the NeoPixels in start up. Additionally there is PRINTDATA that prints out the sensor data to help calibrate and come up with new actions.

Magic Wand Code & Files

https://adafru.it/AQ4

The startup loop mainly connects to all the different components and if HARDWARETEST is defined as 1, then it also test the sensors, NeoPixels, and the Audio FX board.

The Sensor settings control the sensitivity of when the wand cast spells.

HARDWARETEST and PRINTDATA can be defined as 1 to help with debugging

The other variables set pins for various components and define global variables.

```
//Written by Alex Alves for Adafruit Industries.
//Adafruit invests time and resources providing this open source code,
//please support Adafruit and open-source hardware by purchasing products
//from Adafruit!
//BSD license, all text above must be included in any redistribution.
```

```
#include &lt;SoftwareSerial.h&gt;
#include "Adafruit_Soundboard.h"
#include &lt;Adafruit_NeoPixel.h&gt;
#include &lt;Wire.h&gt;
#include &lt;SPI.h&gt;
#include &lt;Adafruit_LSM9DS0.h&gt;
#include &lt;Adafruit_Sensor.h&gt;  // not used in this demo but required!


#define HARDWARETEST 0 // set to 1 to play sound and test neopixles at start up
#define PRINTDATA 0   // prints out sensor data

////////////    SENSOR SETTINGS     ////////////////////
// these settings may have to be changed and calibrated
// orinetation
#define VERTICALVAL 11000  // decrease for more senstive
#define HORIZVAL 6000     //  increase for more senstive

// flick detection
#define FLICKTRIG 17000 // lower = more senstive

// Wrist twist
#define WRISTDIFF 13000 // lower = more senstive
#define WRISTGY  25000 // higher = more senstive



// Sound
// Choose any two pins that can be used with SoftwareSerial to RX &amp; TX
#define SFX_TX 11
#define SFX_RX 10

// Connect to the RST pin on the Sound Board
#define SFX_RST 18
#define SFX_ACT 23

// we'll be using software serial
SoftwareSerial ss = SoftwareSerial(SFX_TX, SFX_RX);
Adafruit_Soundboard sfx = Adafruit_Soundboard(&amp;ss, NULL, SFX_RST);


// NeoPixels
#define PIXELPIN 9
#define NUM_LEDS 7
#define BRIGHTNESS 255 // max 255
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, PIXELPIN, NEO_GRBW +
NEO_KHZ800);


// IMU
Adafruit_LSM9DS0 lsm = Adafruit_LSM9DS0();


enum orientation {
  horiz,
  vertical,
  mid
};
// Global state variables
bool wristTwist = false;
bool flick = false;
orientation orient = horiz;

/////////////////////////////////////////////////////////////////////////////////////////
// setup
/////////////////////////////////////////////////////////////////////////////////////////

void setup() {
```

```
  Serial.begin(115200);
  connectSensor();
  setupSensor();
  setupNeoPixel();
  pinMode(SFX_ACT, INPUT);

  // softwareserial at 9600 baud
  ss.begin(9600);
  // can also do Serial1.begin(9600)

  while (!sfx.reset()) {
    Serial.println("Sound Board not found");
    delay(1000);
  }
  if (HARDWARETEST) {
    if (! sfx.playTrack("BLAST01 WAV") ) {
      Serial.println("Failed to play track1");
    }
    waitSFXFinish();
  }

}
```

Most of the heavy lifting occurs in reciveData1() which is called every 10ms. When this is called the feather samples and the data from the IMU and detects simple triggers such as writs flicks where meet.

```
void reciveData1()
{
  static long int accelCombineL = 0;
  static int accelZL = 0;
  static long int gyroCombineL = 0;
  static int gyroZL = 0;


  lsm.read();
  // combine data from x and y axis to make it more simple
  long int accelCombine = sqrt( (sq((long int)lsm.accelData.x) + sq((long
int)lsm.accelData.y)));
  int accelZ = (int)lsm.accelData.z;

  long int gyroCombine = sqrt( (sq((long int)lsm.gyroData.x) + sq((long
int)lsm.gyroData.y)));
  int gyroZ = (int)lsm.gyroData.z;



  // Numerical differentiante
  int accelCombineDiff = abs((accelCombine - accelCombineL));
  int accelZDiff = abs((accelZ - accelZL));
  int gyroCombineDiff = abs((gyroCombine - gyroCombineL));
  int gyroZDiff = abs((gyroZ - gyroZL));

  int magz = lsm.magData.z;

  //Transfer to static variables for next time
  accelCombineL = accelCombine;
  accelZL = accelZ;
  gyroCombineL = gyroCombine;
  gyroZL = gyroZ;
```

```
  if (PRINTDATA) {
        Serial.print(accelCombine); Serial.print(',');
        Serial.print(accelCombineDiff); Serial.print(',');

        Serial.print(accelZ); Serial.print(',');
        Serial.print(accelZDiff); Serial.print(',');

        Serial.print(gyroCombine);      Serial.print(',');
        Serial.print(gyroCombineDiff); Serial.print(',');

        Serial.print(gyroZ);        Serial.print(',');
        Serial.print(gyroZDiff);    Serial.print(',');

        Serial.print(magz);       Serial.println(' ');
  }

  // orientation detection
  if (magz &lt; -VERTICALVAL) {
    orient = vertical;
  }

  else if (magz &gt; -HORIZVAL) {
    orient = horiz;
  }
  else {
    orient = mid;
  }


  // wrist twist detection
  if ((gyroZDiff &gt; WRISTDIFF ) &amp;&amp; (abs(gyroZ) &gt; WRISTGY)) {
    wristTwist = true;
  }
  else {
    wristTwist = false;
  }

  // Flick detection
  if ((gyroCombineDiff &gt; FLICKTRIG)) { // (gyroCombine &gt; 16000) &amp;&amp;
    flick = true;
  }
  else {
    flick = false;
  }
}
```

The main loop decides where reciveData1() needs to be sampled and then also matches the conditions set from reciveData1() to spells. Once a spell is selected it sets the light and sound effects. No new spells can be cast until the audio of the previous spell has finished.

```
void loop() {
  static bool spellCasting = false;
  static long unsigned int sampleTrig = 0; // trig for when to sample sensors

  // Take new sample
  if (millis() &gt; sampleTrig) {
    reciveData1();
    sampleTrig = millis() + 10;
  }

  // conect basic actions together to make difrent spells
  if (!spellCasting) {
    // trigger spell 1
```

```
    if (wristTwist &amp;&amp; (orient == vertical) ) {
      spellCasting = true;
      sfx.playTrack("LIGHT01 WAV");
      setColor(strip.Color(255, 255, 255, 255));
      delay(5);
    }

    else if (wristTwist &amp;&amp; (orient == horiz) ) {
      spellCasting = true;
      sfx.playTrack("WIND01  WAV");  //temp
      setColor(strip.Color(150, 255, 100));;
      delay(5);
    }

    else if (flick &amp;&amp; (orient == horiz)) {
      spellCasting = true;
      sfx.playTrack("FIRE01  WAV");
      setColor(strip.Color(255, 50, 20));
      delay(5);
    }

    else if (flick &amp;&amp; (orient == vertical)) {
      spellCasting = true;
      sfx.playTrack("THUNDE~1WAV");
      setColor(strip.Color(0, 0, 255, 100));
      delay(5);
    }
  }

  // spell is being cast
  else {
    // spell just finished
    if (!sfxActive()) {
      spellCasting = false;
      setColor(strip.Color(0, 0, 0));
    }
  }

}
```

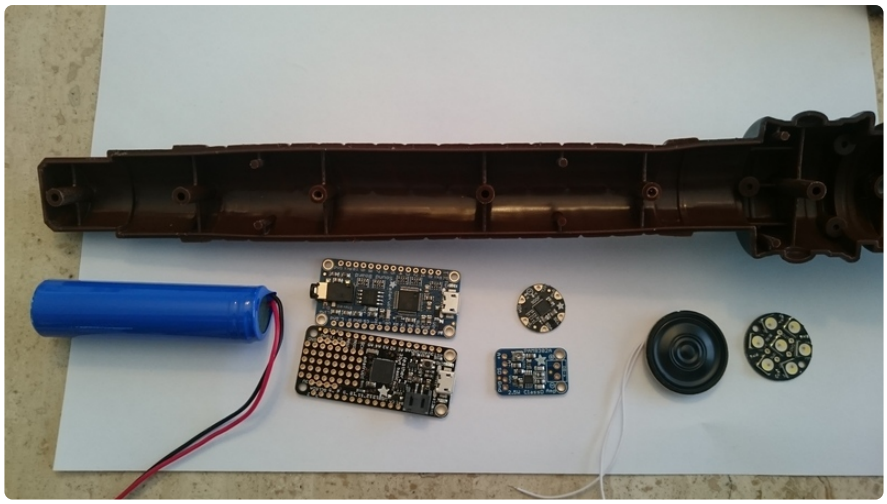
There are also various helper functions that help set up the individual sensors, interface with the Audio FX board and control the NeoPixels. Much of the helpfull debug functions are also here.

```
/************************ MENU HELPERS ***************************/

// Is a sound playing
bool sfxActive() {
  return (! digitalRead(SFX_ACT));
}

// block wait till sound is finished
void waitSFXFinish() {
  delay(5);
  while (sfxActive()) {
    delay(1);
  }
  delay(5);
}


/************************ MENU HELPERS ***************************/

// set colors in sequence
void colorWipe(uint32_t c, uint8_t wait) {
```

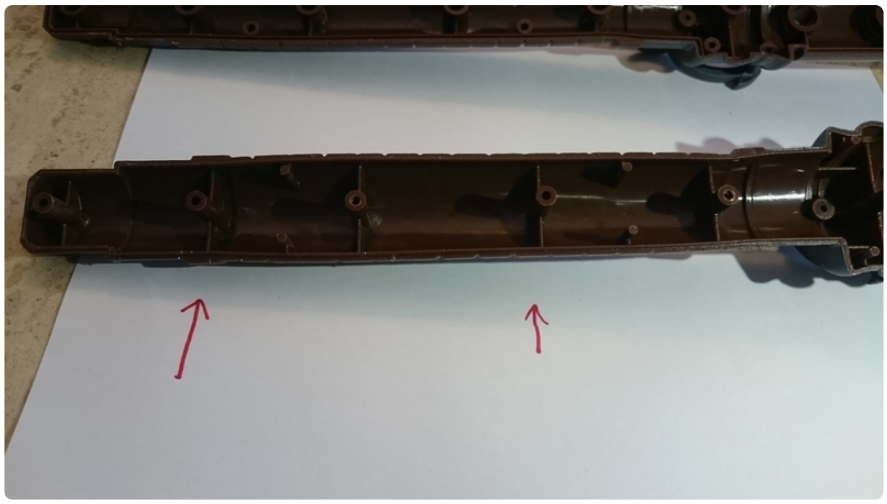
```
  for (uint16_t i = 0; i &lt; strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}

// set all of pixles to one color
void setColor(uint32_t c) {
  for (uint16_t i = 0; i &lt; strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
  }
  strip.show();
}

// set intial pixles status and set brightness
void setupNeoPixel() {
  strip.setBrightness(BRIGHTNESS);
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'

  if (HARDWARETEST) {
    Serial.println("Testing NeoPixels");
    colorWipe(strip.Color(255, 0, 0), 50); // Red
    colorWipe(strip.Color(0, 255, 0), 50); // Green
    colorWipe(strip.Color(0, 0, 255), 50); // Blue
    colorWipe(strip.Color(0, 0, 0, 255), 50); // White
    setColor(strip.Color(0, 0, 0));
  }
}

/********************** LSM9DS0 *********************************/

void setupSensor()
{
  // 1.) Set the accelerometer range
  lsm.setupAccel(lsm.LSM9DS0_ACCELRANGE_2G);

  // 2.) Set the magnetometer sensitivity
  lsm.setupMag(lsm.LSM9DS0_MAGGAIN_2GAUSS);


  // 3.) Setup the gyroscope
  lsm.setupGyro(lsm.LSM9DS0_GYROSCALE_245DPS);
}


void connectSensor()
{
  Serial.println("Connecting to Sensor");

  // Try to initialise and warn if we couldn't detect the chip
  while (!lsm.begin())
  {
    Serial.println("Oops ... unable to initialize the LSM9DS0. Trying again");
    delay(1000);
  }
  Serial.println("Found LSM9DS0 9DOF");
```

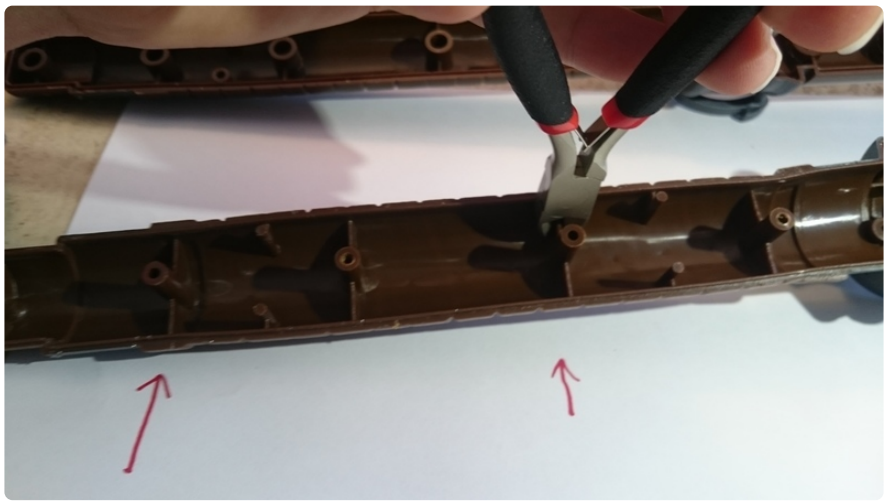
Hardware Shell

We are going to turn this into a wand is to remove the head of the hammer, and replace it with a glowing piece of frosted glass. First you will need to take the hammer apart and see what you have to work with. Once open start to lay out the components and figure out where all the electronics and wires will go such as the layout below.



Once you have your parts laid out next to the shell, you'll discover that there are are support plastic and guides that have to be removed. I find that a combination of snips and using a rotary tool makes fast work removing them. Sanding and an xacto blade flow up to clean up any rough edges.

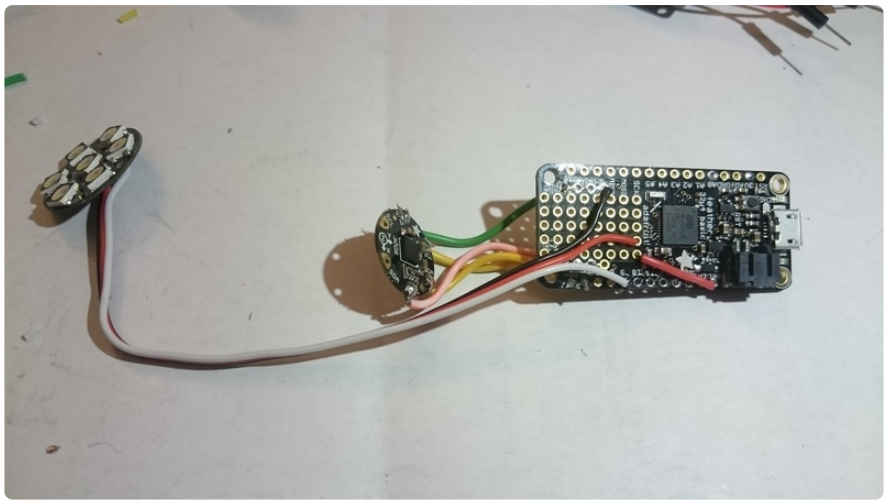




Assembly

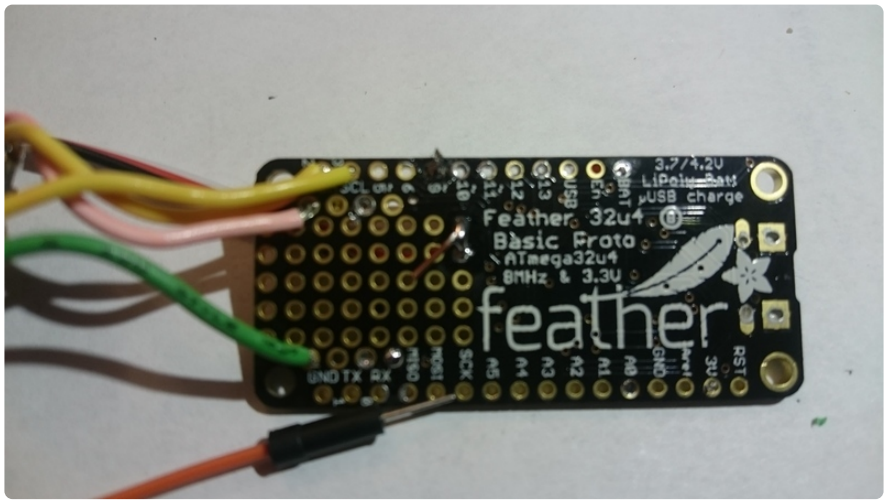
For every wire you cut keep in mind where that wire will go and how long it needs to be. The two most complicated pieces to connect are the Audio FX board and the Feather. Essentially there will be two separate assemblies that will come together in the last few steps. The two boards will be close to one other with little room for wiring, so keep that in mind.

Precut wires of the appropriate length for the NeoPixels and the IMU, and solder them to the IMU and NeoPixels respectively. Solder a wire from the Feather **BAT** pin to the prototyping area, as the short red wire shows below. I found it was easier to solder the wires of the NeoPixels on top of the feather and IMU on the back side. The IMU will be able to attach well at the back of the Feather connecting the **SDA** to **pin 2** and **SCL** to **pin 3.** The **3V** and **GND** on the IMU conect to the the prototyping area. The NeoPixels connect to **pin 9**, **GND** in the prototyping rail and the area on the prototyping area connected to the battery. Once this is done I recommend testing both the NeoPixels and the IMU to ensure they work and the solder joints are good.

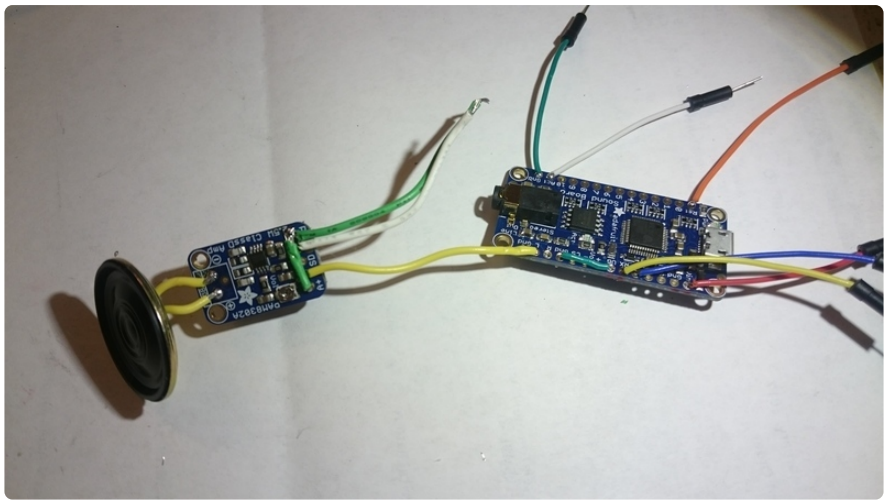


Bridge the back prototypic area so that the **5V** from the NeoPixels connects to the **BAT** connection you made earlier.

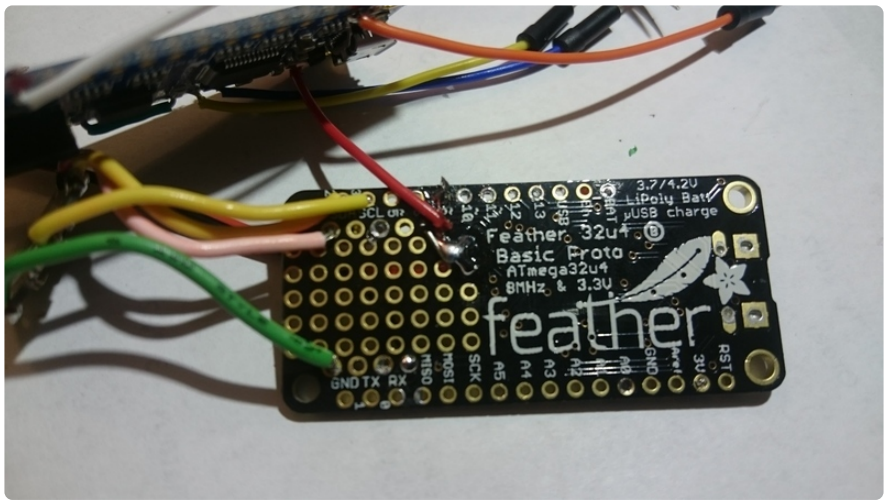
Load Arduino sketch with **HARDWARETEST** and **PRINTDATA** defined as 1. This makes is so that on boot up the NeoPixels will light in sequence, and then print out the sensor data to the serial port that can be viewed in the serial plotter. Make sure you have a battery plugged into the feather, the NeoPixles may give you problems running off just the usb.



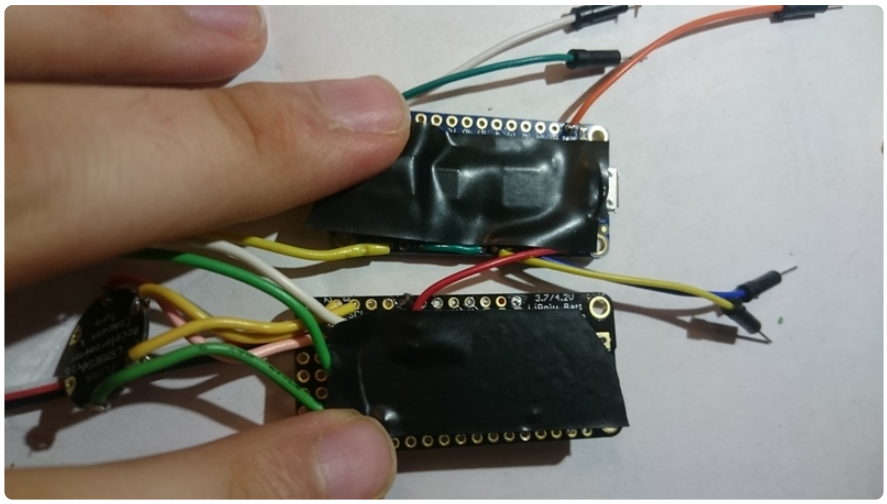
Next up is the Audio FX board. Solder short wires to **Vin, Rst, Act, Rx, Tx,** and **GND.** Also solder a jumper from **GND** to **UG** so the board works in serial mode. At this point I recommend using clips to temporarily test the connections for the Audio FX board, the serial communication and the audio transfer. Next remove the wires from the speaker and solder new wires from the speaker to the audio amp, remember plan out the wire lengths first. Measure and solder wires for the **Vin** and **GND** of the audio amp, these won't be connected till a bit latter on the feather. Finally add a jumper from **GND** to **A-,** and a wire from the audio output (**L** or **R**) of the Audio FX board to the the **A+** input of the amp.

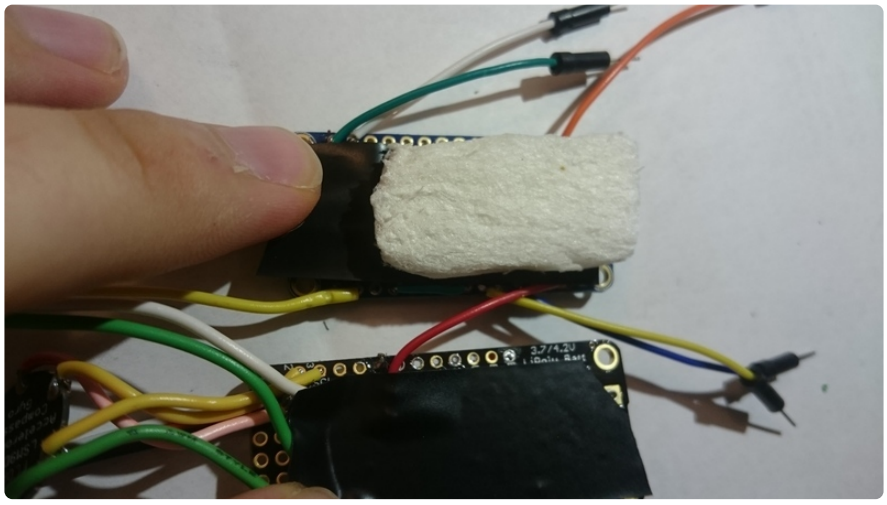


The final step is soldering the two sets modules together. This soldering is a bit more complex and it is essential that you cut the wires to the correct length, too long and they won't bend, too short and you can't solder them. First solder the **Vin** and **GND** on the audio amp to the underside of rails of the prototype area. Next solder the **Vin** of the Audio FX board to the **BAT** connections you made earlier in the prototype area.

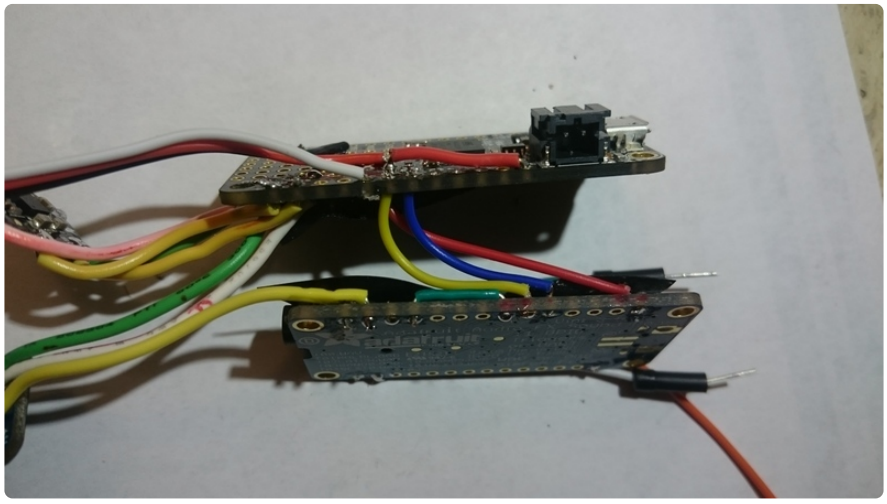


Cover the bottom of the feathers propitiating area and the top of the Audio FX board in electrical tape, this will help avoid any shorts. Also find something that can help cushion the boards from one another, I found that a cut up packing peanut works well.





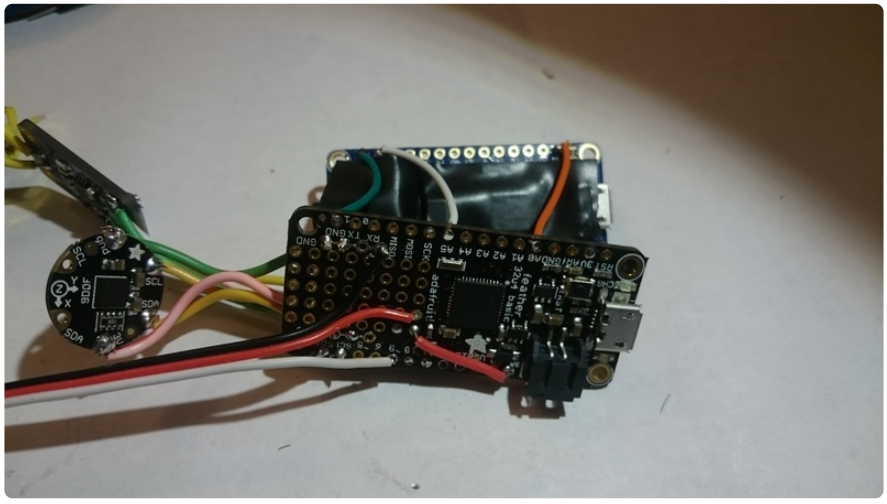
Since Vin is connected next connect **TX** and **RX** from the Audio FX to the feather, since these are all on one side you can keep the two boards open like a glasses case to make the soldering easier.

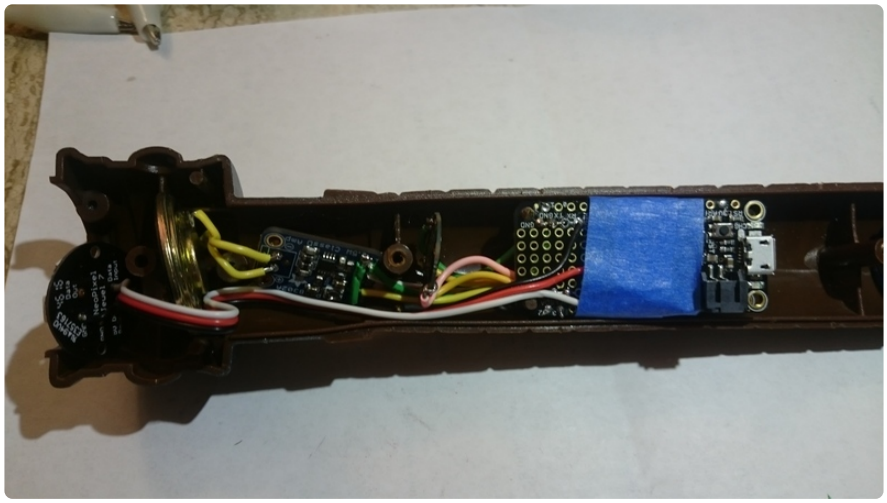


Finally solder the **GND**, **Rst**, and **Act** wires from the Audio FX board to the Feather. I found that by leaving enough wire for the boards to have a 3/4” gap between them it was a good balance between ease of soldering and lack of excesses wire. After those

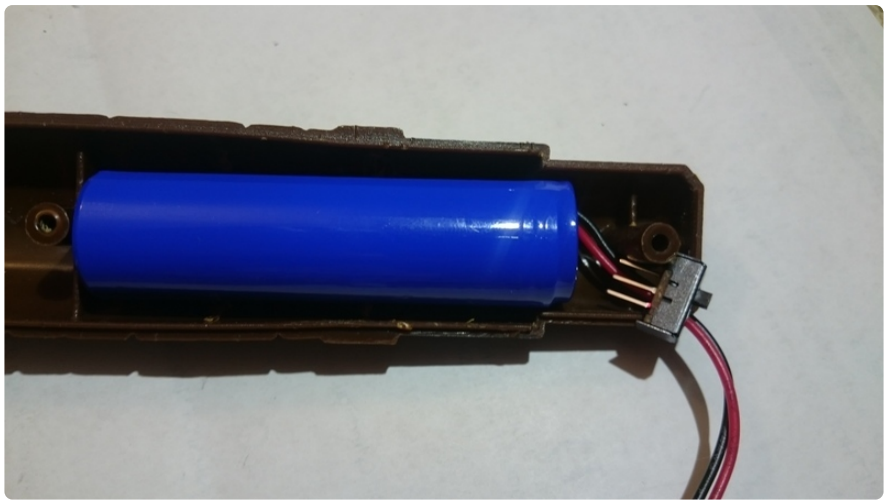
are soldered place the spacer (packing peanut) between the boards and carefully push them together making sure the wires bend neatly. I found that it was helpful to use blue tape to hold the two boards together to avoid risking damaging any wires. Once done perform a fit test with the components and test everything is working.

Now run the Arduino sketch again, with **HARDWARETEST** defined as **1** but **PRINTDATA** as **0**. Now on boot up the Arduino should play BLAST01.WAV during the setup. If it is unable to connect it will print out on the serial port and you should check the **RST**, **TX** and **RX** lines on the Audio FX board. If the error is about how it failed to play the file, make sure you transfered the sound clips to the board correctly and remembered to properly eject it.

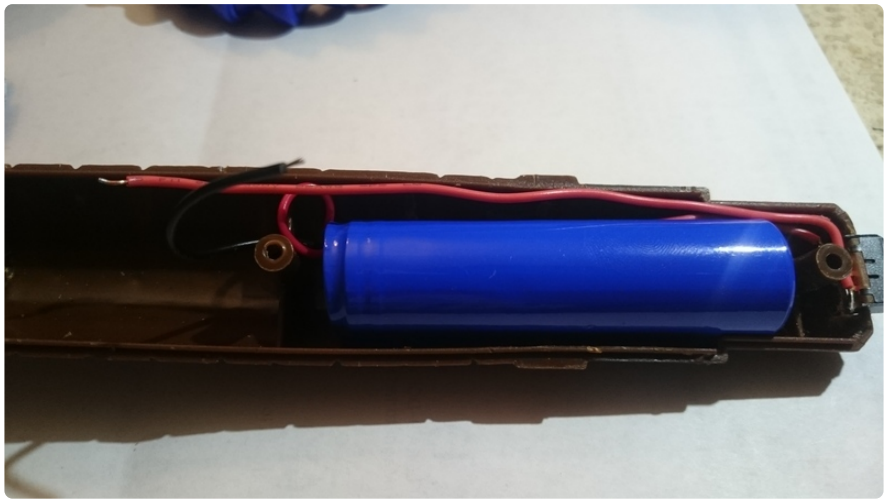


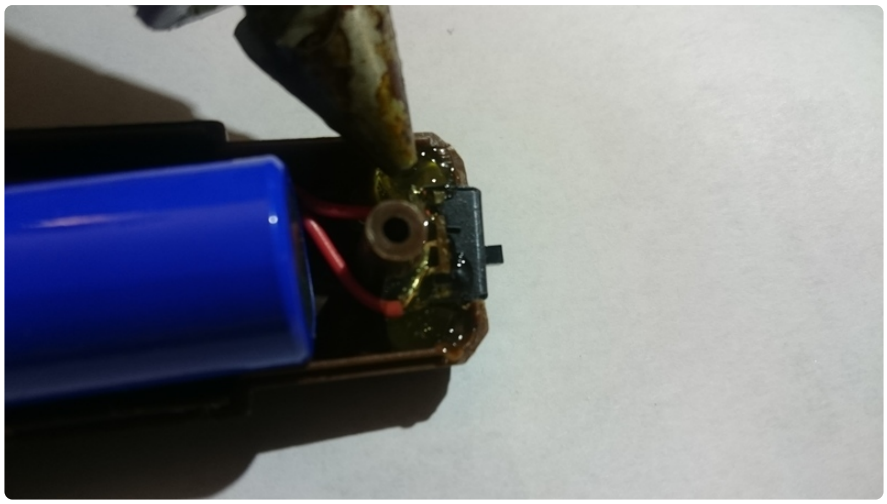


Now to deal with powering the wand. Figure out where you want the power on off switch to be, ideally hidden, the toy I chose has a rubber piece that slides over the bottom that I will use to hide the switch. Cut out room in the toy shell for the switch in the chosen location.

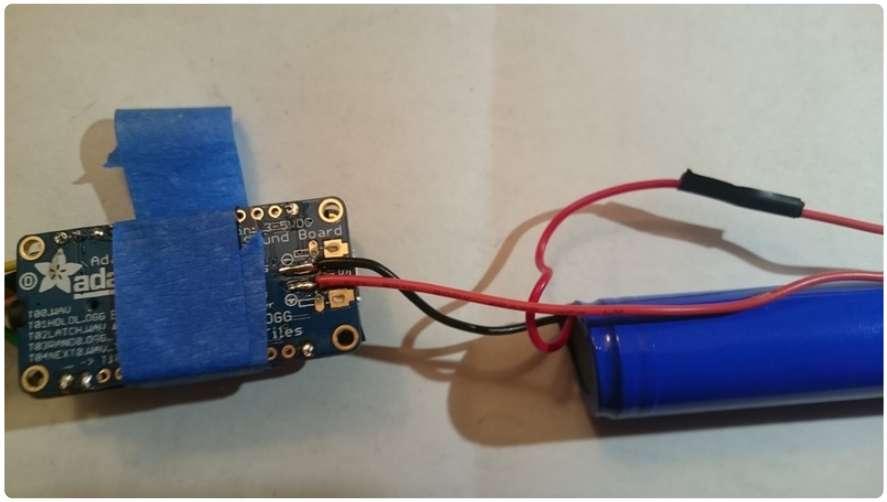


Cut off the JST connector from the battery, unfortunately in this build there isn't enough room for using the JST connector. Take the positive wire from the battery and solder it through the switch so that the switch cuts off the battery from the rest of the project. Guide the wires around the battery and finally hot glue the switch and the battery in place.



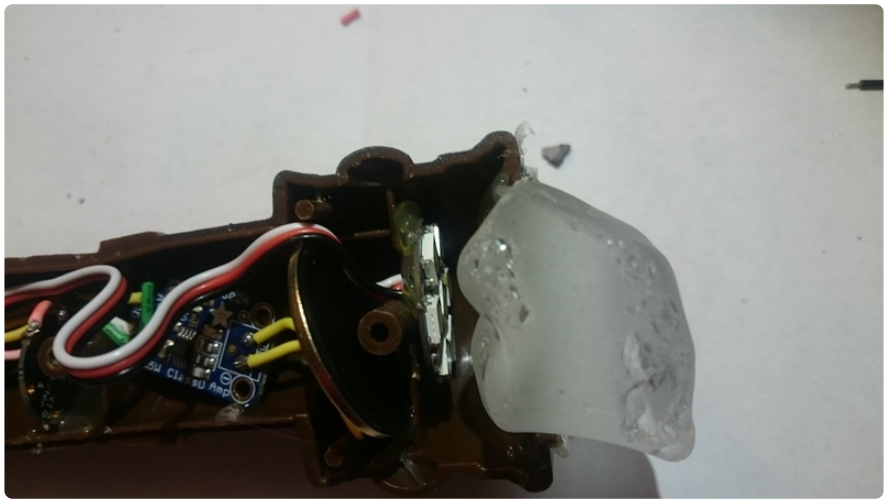


Solder the wires from the switch and battery to the pads for a JST connector on the bottom of the Audio FX board. At this point all of the soldering is done. Test that all the components work and that all the connections are good. The wand should run on battery power, and can still be charged through the usb on the Feather. Once everything is checked out glue down the amp, speaker and IMU. Try and make the IMU and square as possible. With the IMU glued down, it's now possible to calibrate the settings on the motion detection.



Attaching the frosted glass piece will be varied depending on the exact piece of glass you get, I was able to buy a large pack that had various sizes and shapes. I went with a piece that was a bit large so I had to cut away a bit of the shell, but if you went for a smaller piece this wouldn't be necessary. I used clear hot glue to glue the crystal to the side of the shell with electronics. Mess around with different placements for the NeoPixels to get the best light output (you can also use high grit sandpaper to remove frost part of the crystal if necessary). Once figured out glue down the NeoPixels as well. Finally combine the two shells together, I had to remove a bit around the crystal to make it fit. If there are gaps you can use the hot-glue as an opaque filler, just make sure to keep the ability to open the shell. Hot glue doesn't

stick to cooking spray so you can get some on the seem and the crystal and then fill in the gaps.



Charge the battery with the usb port and flip the switch and start casting spells.