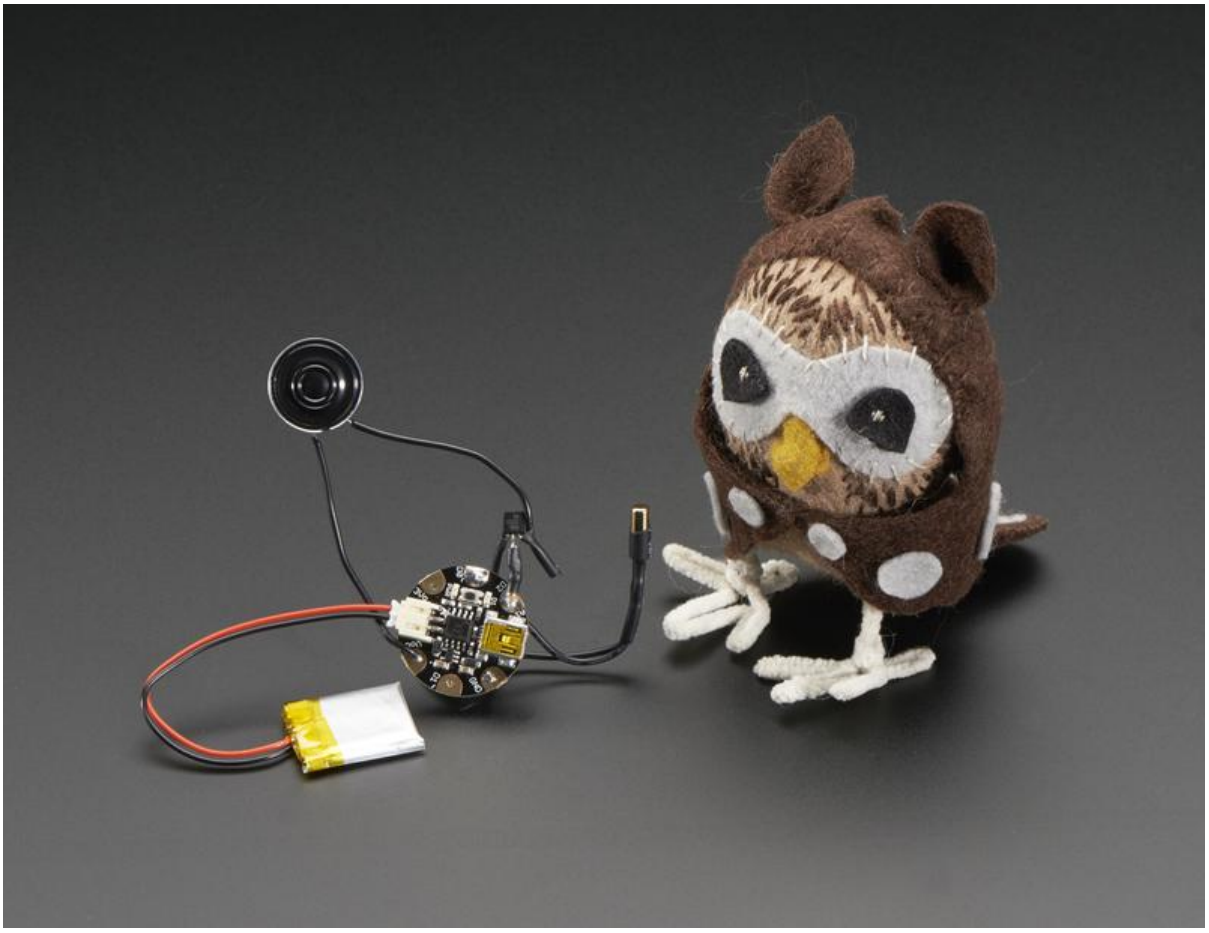
Chirping Plush Owl Toy

Created by Becky Stern



https://learn.adafruit.com/chirping-plush-owl-toy

Last updated on 2022-12-01 02:09:32 PM EST

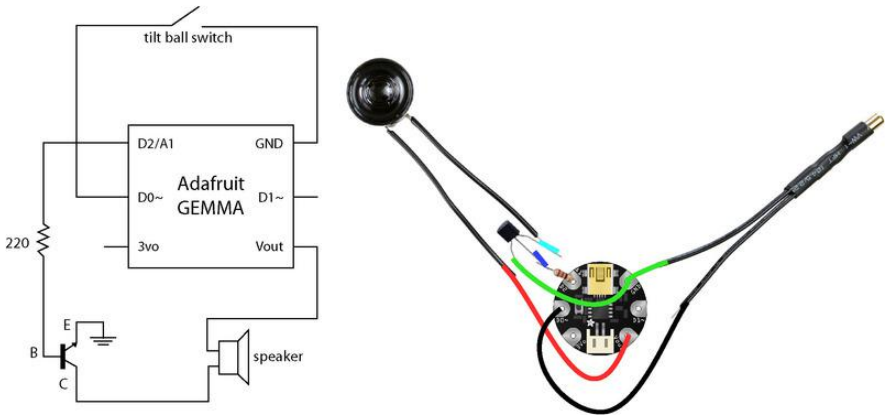
Table of Contents

Overview

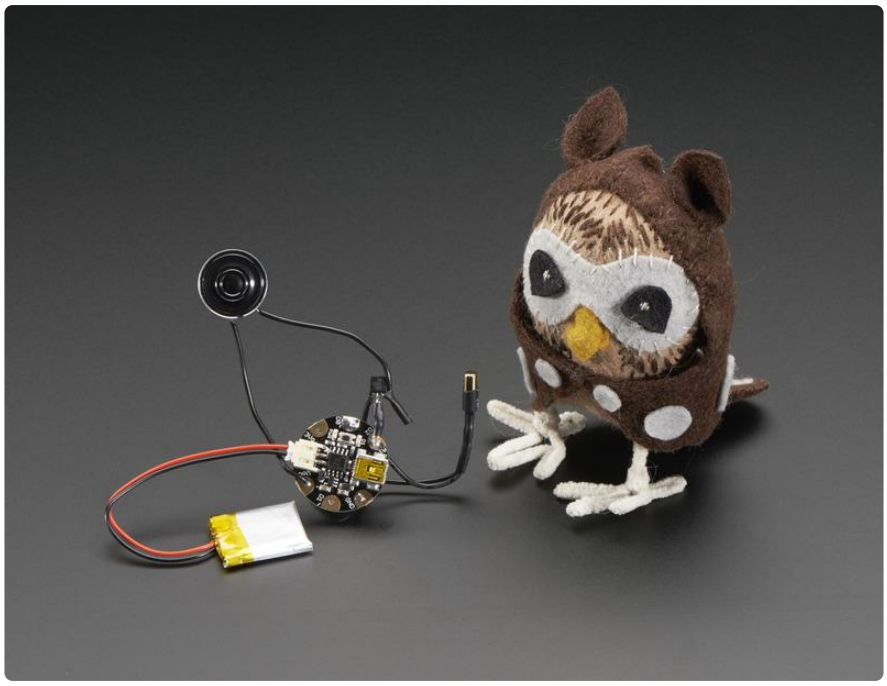
Chirp chirp! Make this delightful sound-making plush toy with the Sew-Your-Own Owl Kit (http://adafru.it/1728) and GEMMA plush toy guts kit (http://adafru.it/1759). This simple circuit uses a tilt ball switch to trigger GEMMA to play a sound on the small speaker. You can program your own circuit to make other sounds too, so you can even mod a plush toy you already have!

This is a beginner level sewing and soldering project. This tutorial has detailed photos for each step of circuit assembly and installing the circuit into the toy, and the owl kit comes with detailed printed instructions recommended for slightly experienced stitchers as young as eight.

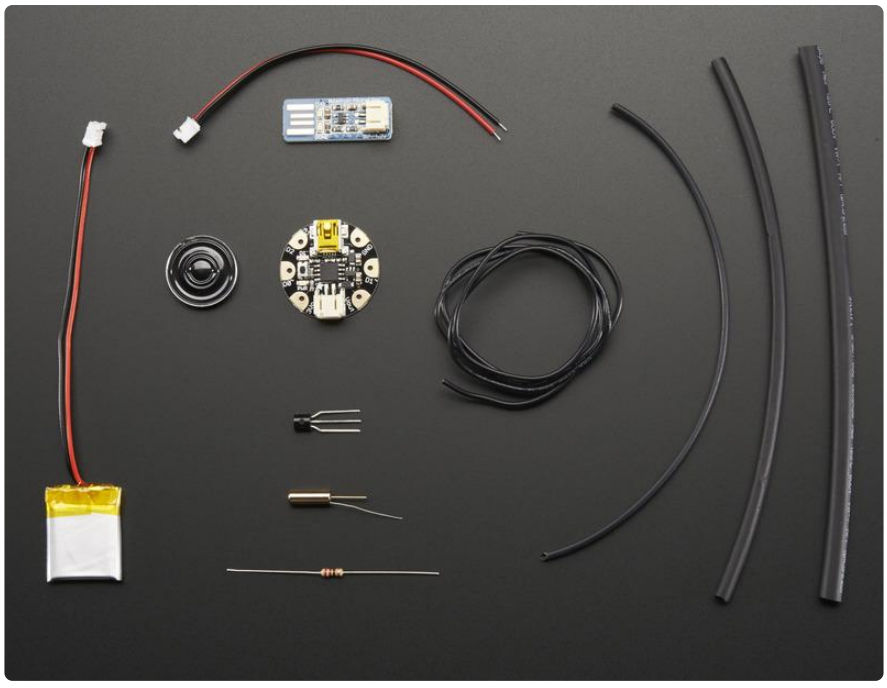
This guide was written for the Gemma v1 and v2 boards. It can also be done with the Gemma M0. We recommend the Gemma M0 as it is easier to use and is more compatible with modern computers! The wiring is the same.



Click to enlarge this circuit diagram, and refer back to it throughout the building of your circuit!



Tools & Supplies



Your GEMMA plush toy guts kit (http://adafru.it/1759) includes:

- Adafruit Gemma M0 () or Adafruit GEMMA ()
- 150mAh lipoly battery (http://adafru.it/1317)
- Micro Lipo charger (http://adafru.it/1304)
- PN2222 transistor (http://adafru.it/756)
- Tilt ball switch (http://adafru.it/173)

- 220 ohm resistor
- small speaker
- 24 inches of solid core wire (http://adafru.it/290)
- three sizes of heat shrink tubing (http://adafru.it/344)

You will also need the Sew-Your-Own Owl Kit (http://adafru.it/1728) or an existing plush toy to mod.

In addition to the above materials, you should have the following tools on hand to complete this project:

Any entry level 'all-in-one' soldering iron that you might find at your local hardware store should work. As with most things in life, you get what you pay for.
Upgrading to a higher end soldering iron setup, like the Hakko FX-888 that we stock in our store (http://adafru.it/180), will make soldering fun and easy.

Do not use a "ColdHeat" soldering iron! They are not suitable for delicate electronics work and can damage the boards (see here ()).

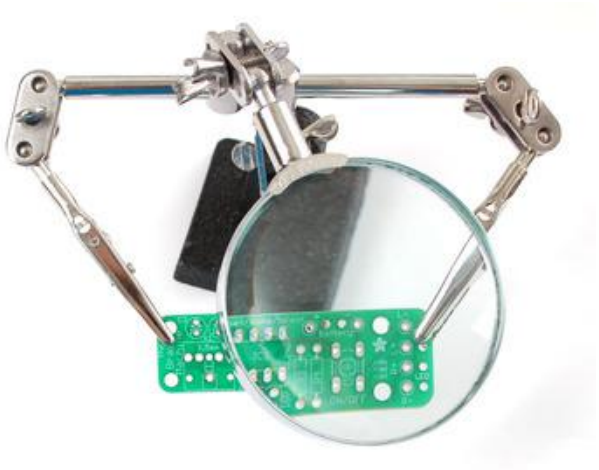
Click here to buy our entry level adjustable 30W 110V soldering iron. (http://adafru.it/180)

Click here to upgrade to a Genuine Hakko FX-888 adjustable temperature soldering iron. (http://adafru.it/303)

Learn how to solder with tons of tutorials! ()

You will want rosin core, 60/40 solder. Good solder is a good thing. Bad solder leads to bridging and cold solder joints which can be tough to find.

Click here to buy a spool of leaded solder (recommended for beginners). (http://adafru.it/145)



A helping third hand tool really makes this project a joy to build.

Click here to buy a helping third hand tool. (http://adafru.it/291)

Don't forget your wire strippers (http://adafru.it/527)!

You'll need a pair of flush snips (http://adafru.it/152).

Sharp scissors are a must!

You will need a good quality basic multimeter that can measure voltage and continuity.

Click here to buy a basic multimeter. (http://adafru.it/71)

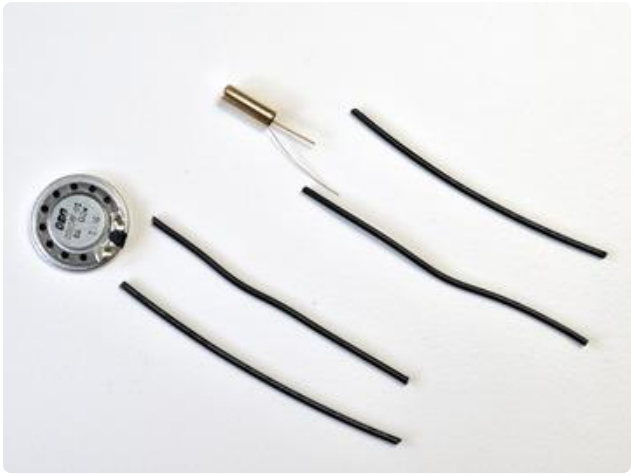
Click here to buy a top of the line multimeter. (http://adafru.it/308)

Click here to buy a pocket multimeter. (http://adafru.it/850)

Don't forget to learn how to use your multimeter too! ()

Heat gun or lighter to shrink your heat shrink tubing!

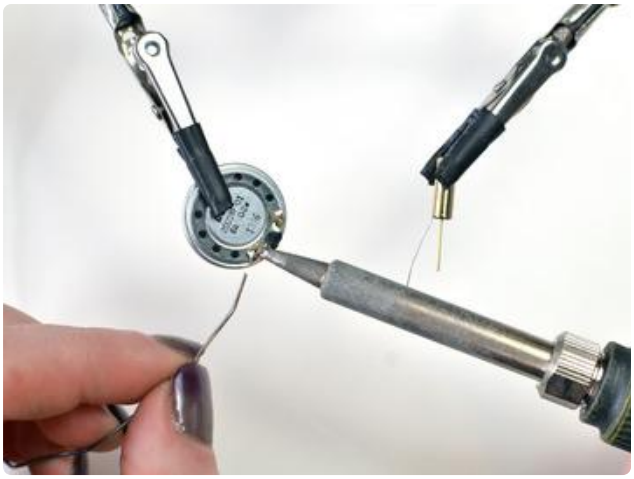
Solder Circuit

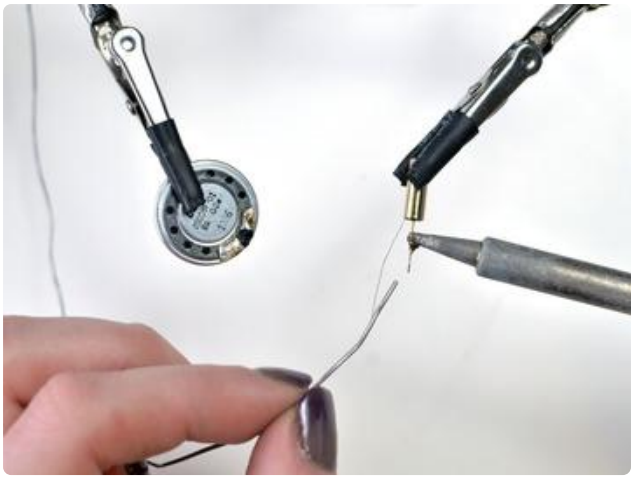


Start by preparing the speaker and tilt ball switch.

Cut five pieces of wire about four inches in length each.

Use wire strippers to strip the insulation off the last quarter inch of each end of each wire.



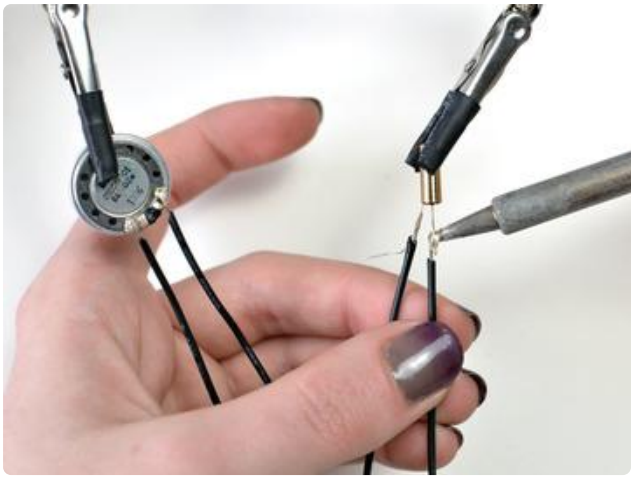


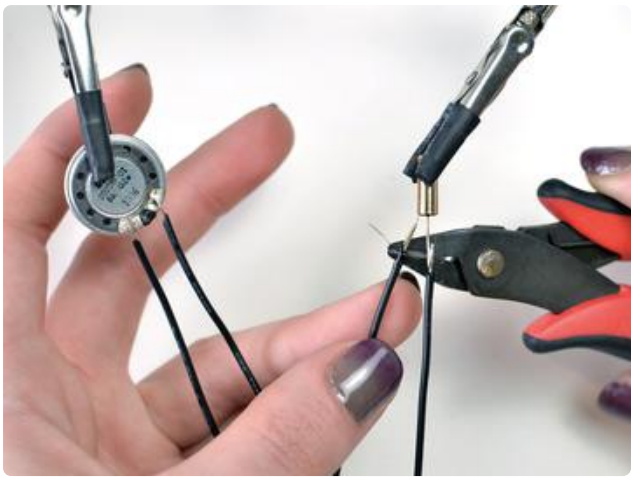
Place the small speaker and tilt ball switch in the jaws of a helping hand tool.

Use a soldering iron to heat up one of the pads of the small speaker.

Apply a small amount of solder to the heated pad to tin it. This will make it easier to attach a wire in the next step.

Repeat to tin the other speaker pad and the two legs of the tilt ball switch.

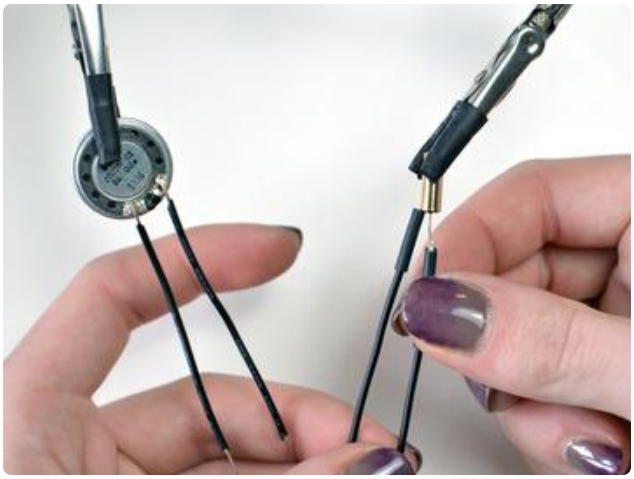


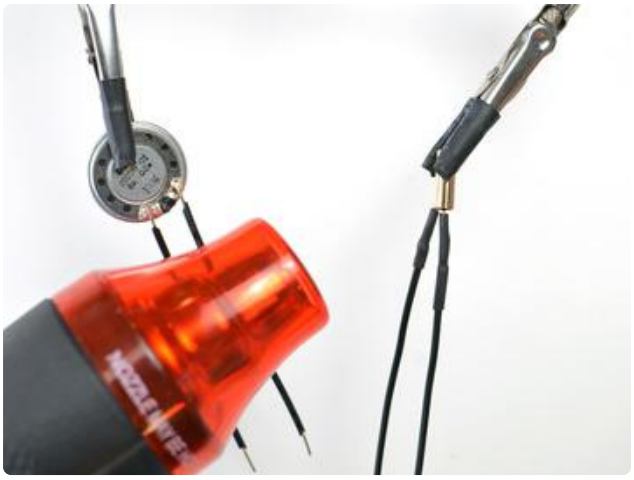


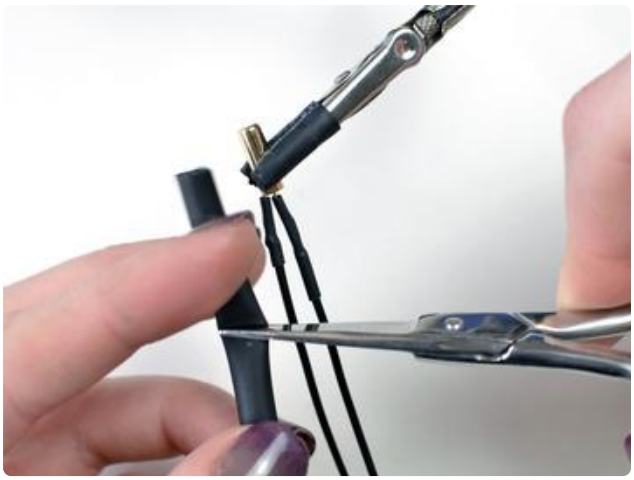
Hold a wire up to the speaker's pad and reheat the solder with your soldering iron. Hold it to the joint until the solder flows around the wire, then remove heat and hold the wire until the solder solidifies.

Repeat to solder wires to the the other speaker pad and both legs of the tilt ball switch.

Snip off any excess wire.





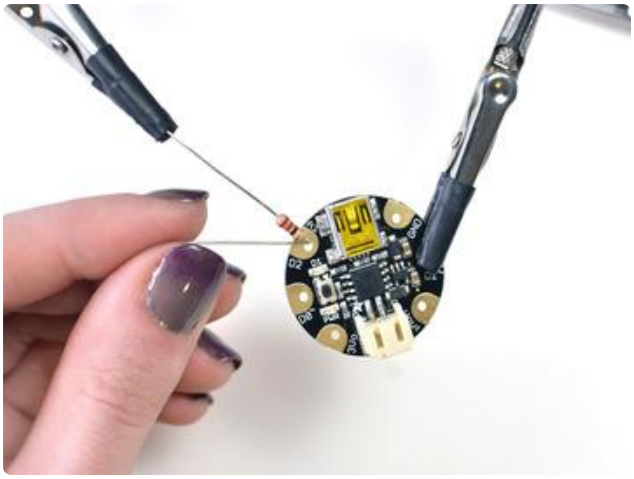


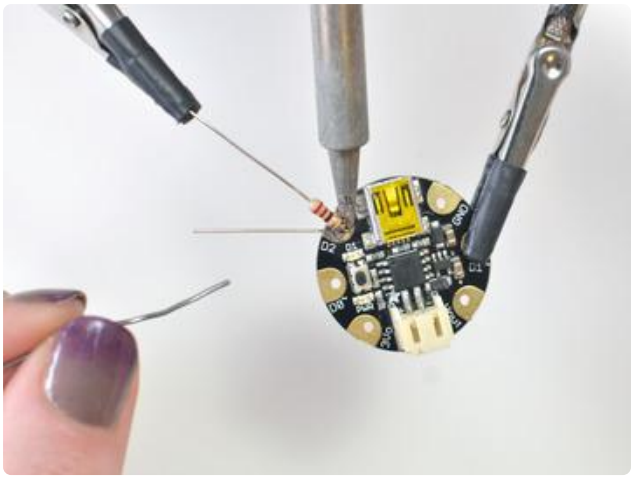
Slide a piece of the smallest diameter heat shrink tubing over each leg of the tilt ball switch to insulate the wires and solder connections.

Use a heat gun to shrink the tubing.

Cut a piece of larger heat shrink tubing and slide over the entire tilt ball switch and wire connections; heat to shrink.

Your speaker and tilt ball switch are now ready for attaching to the rest of your circuit!



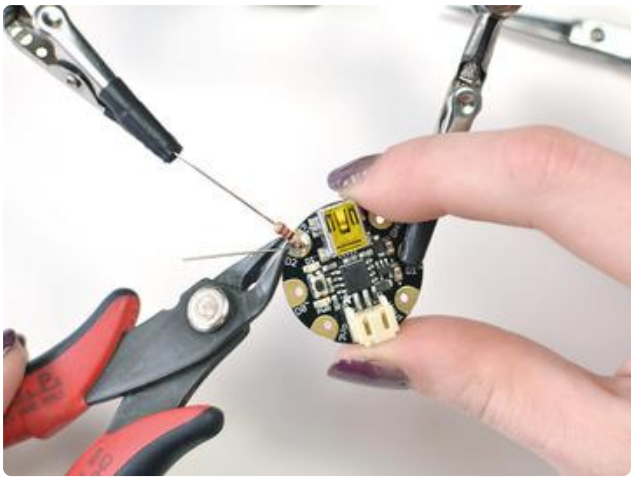


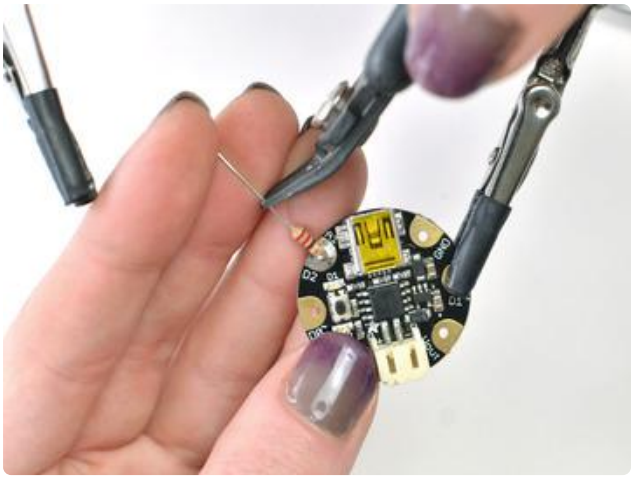
Now get GEMMA set up in your helping hand tool.

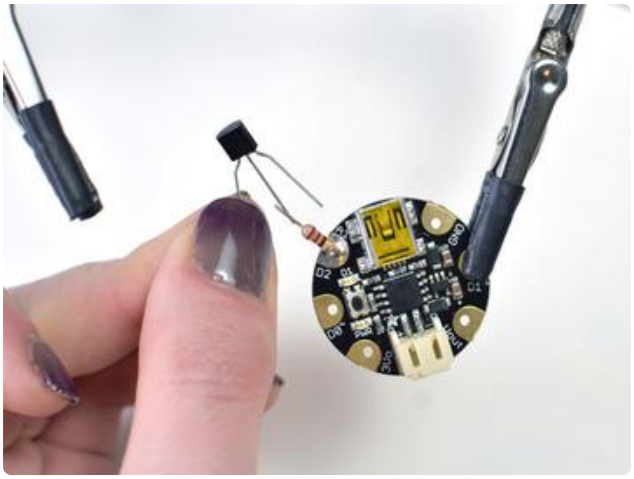
Insert one of the legs of the 220 ohm resistor into the pin marked D2 on GEMMA.

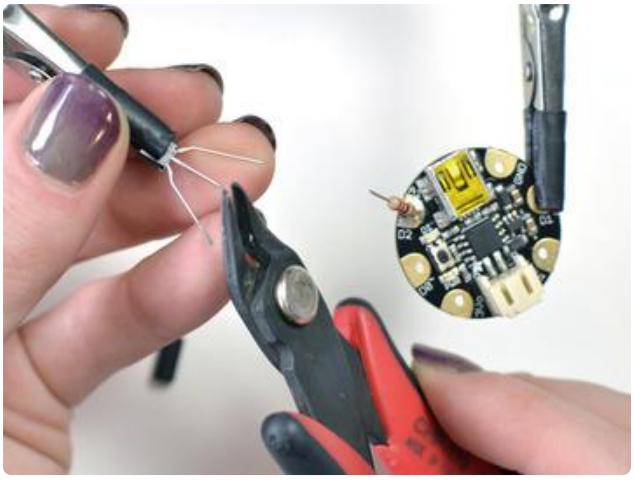
Solder in place.

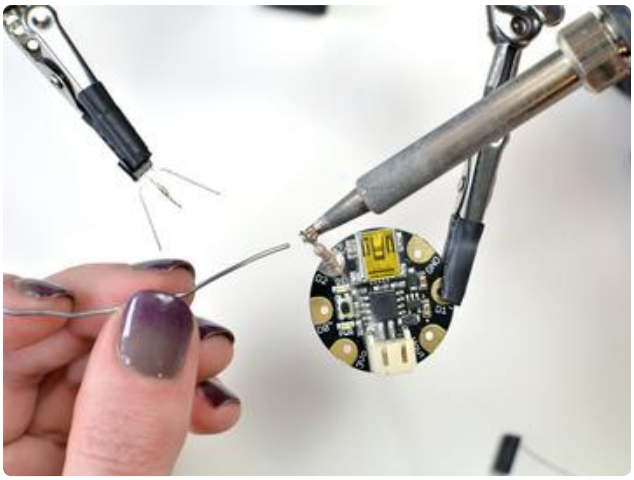
Snip off the extra wire lead and cut the free-hanging wire lead to half its length.









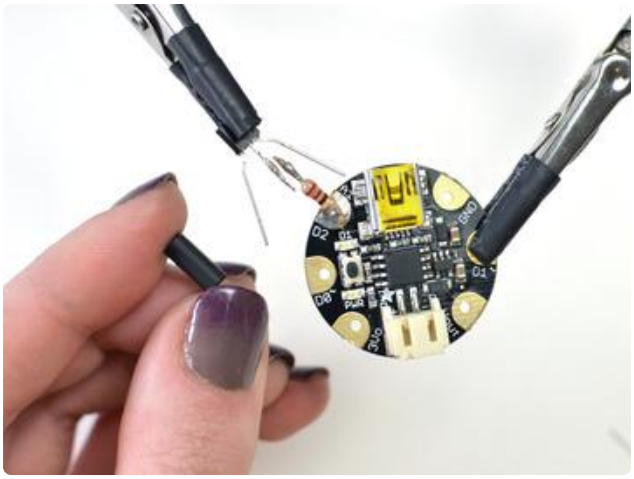


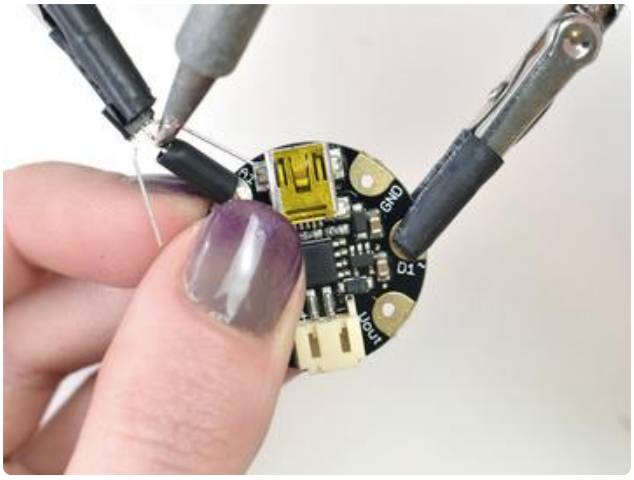
Next it's time to connect up the center leg of the transistor.

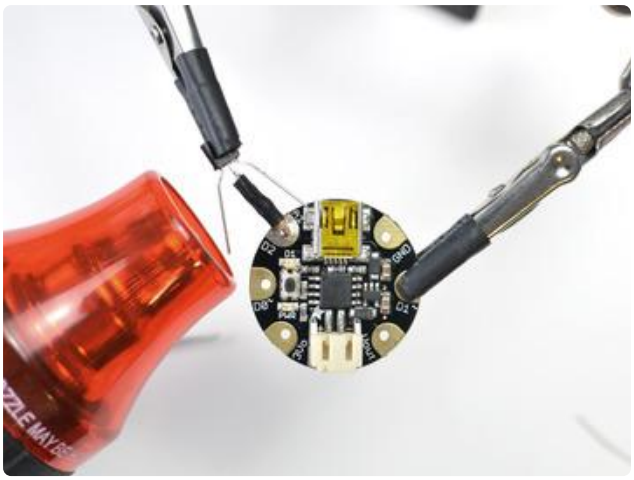
Pay close attention to the orientation of the transistor. The flat side should face the same direction as the front of GEMMA.

Clip the transistor's center lead shorter and set up the component in your helping hand tool.

Tin the lead of the resistor and the center lead of the transistor.



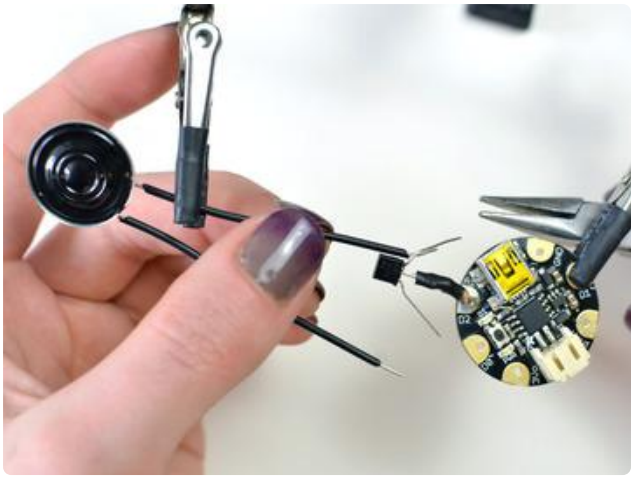


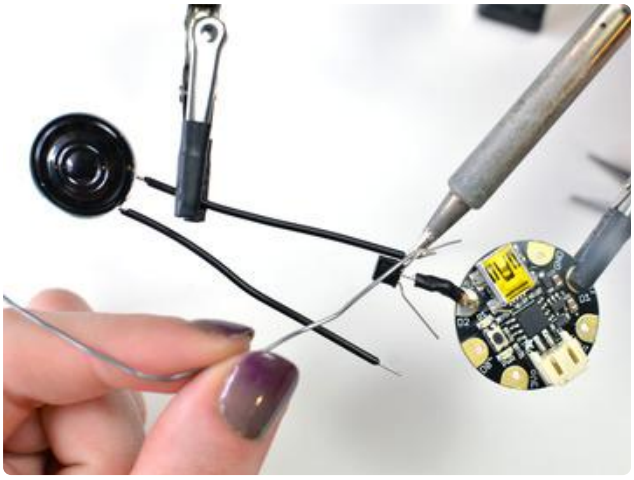


Don't forget heat shrink tubing! Cut a small piece of the medium size and place it over the resistor.

Position the leads next to each other and carefully solder in place.

Heat the tubing to shrink it down.



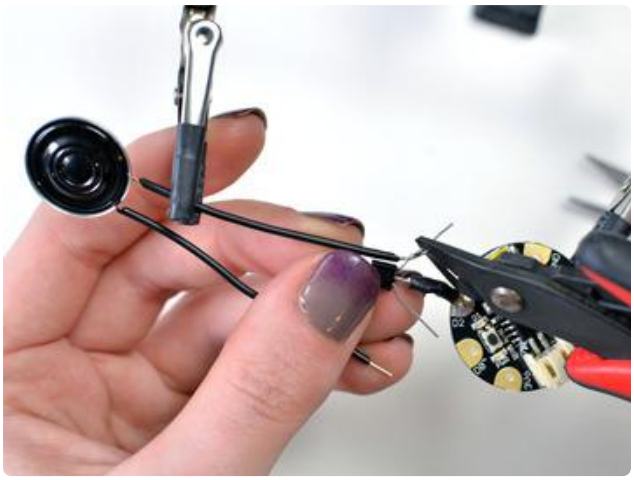


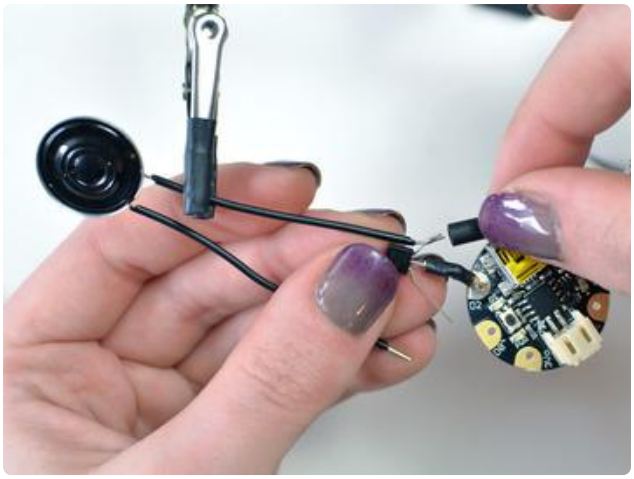
Set up the speaker in your helping hand tool and position one of its wire leads next to the transistor lead on the right.

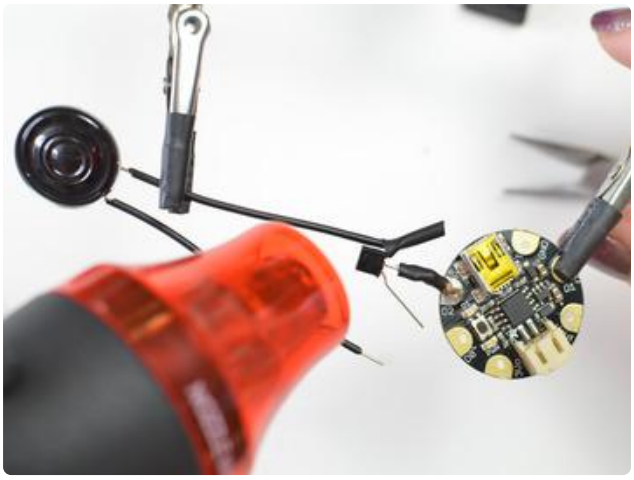
Twist the wires together and solder.

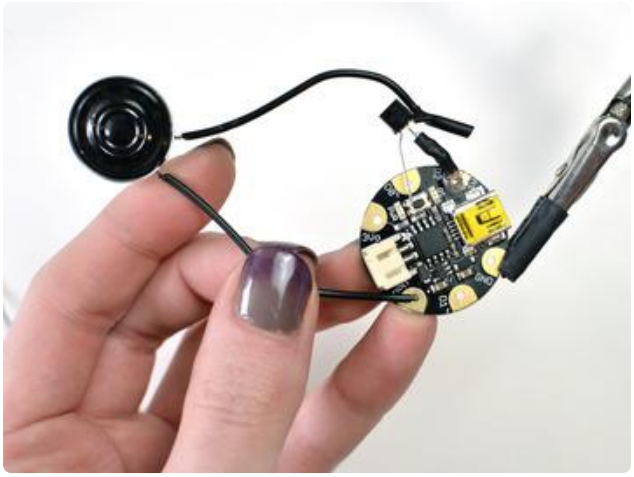
Snip off any excess and slide a piece of heat shrink tubing over the connection.

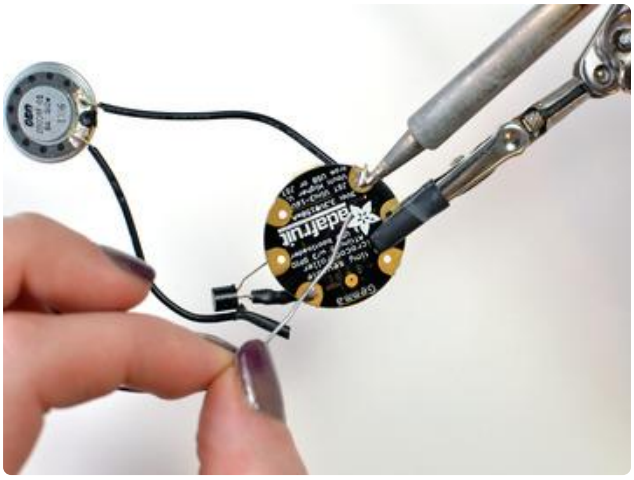
Heat to shrink in place.





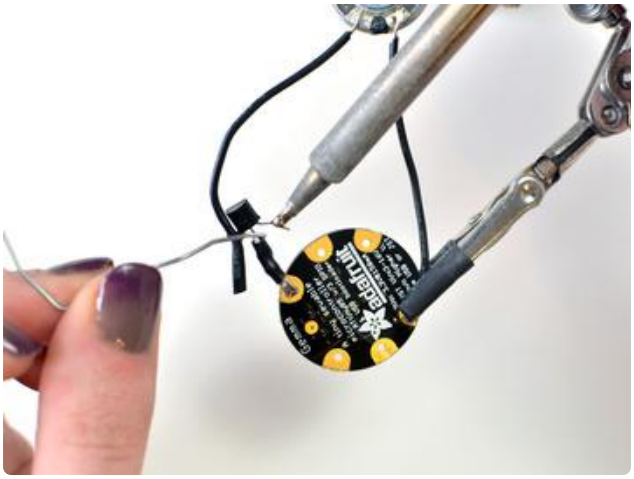


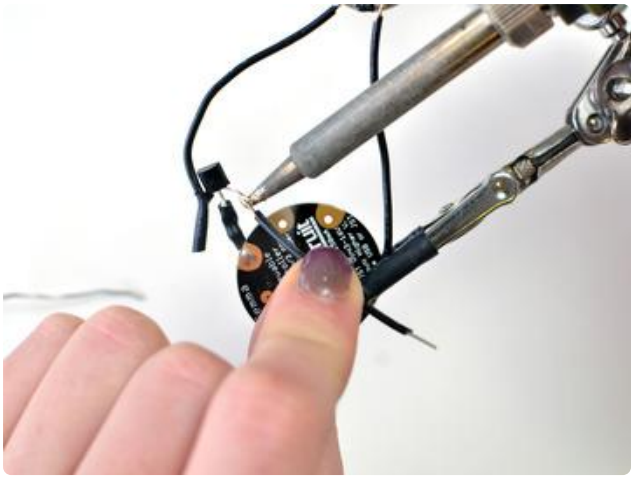




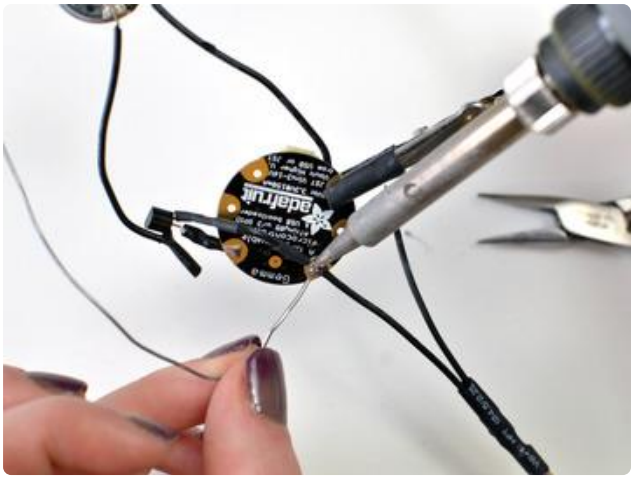
Bend the speaker's other wire to the Vout pin on GEMMA and solder in place.

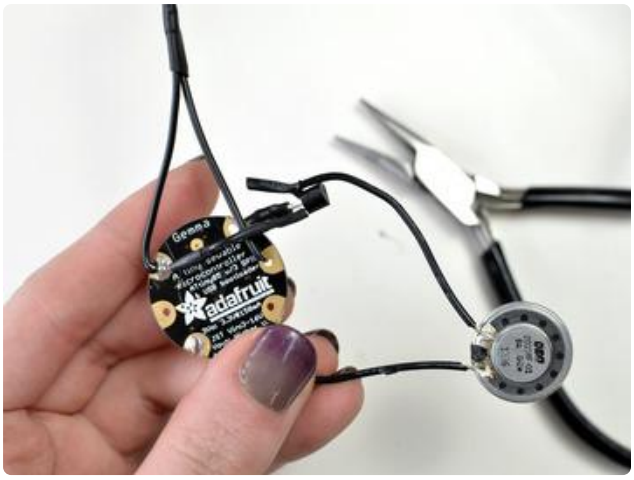
Clip off any excess wire.





Tin the last remaining lead of the transistor and solder to a piece of wire. Slide on some heat shrink and use a heat gun to shrink it.

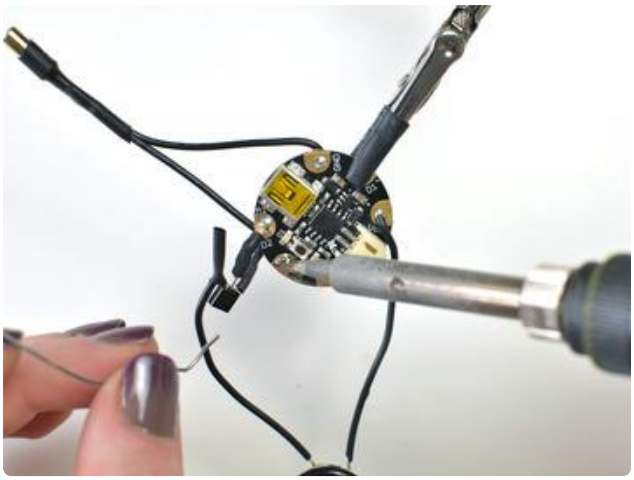


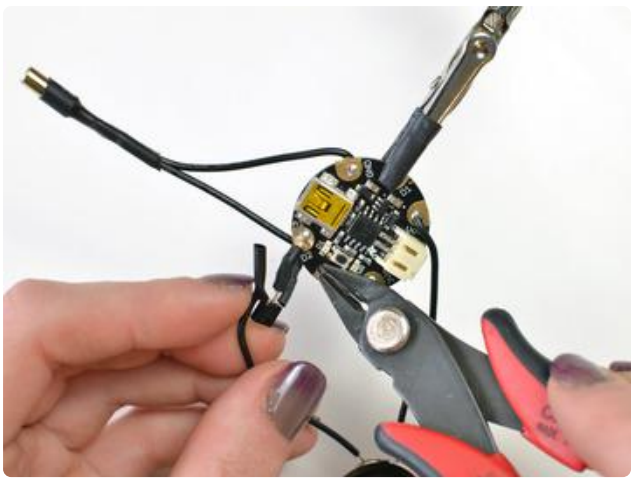


The tilt ball switch has two leads. Solder one lead to D0 on GEMMA and the other lead to GND on GEMMA.

Solder both of these wires to the pad of the circuit board together.

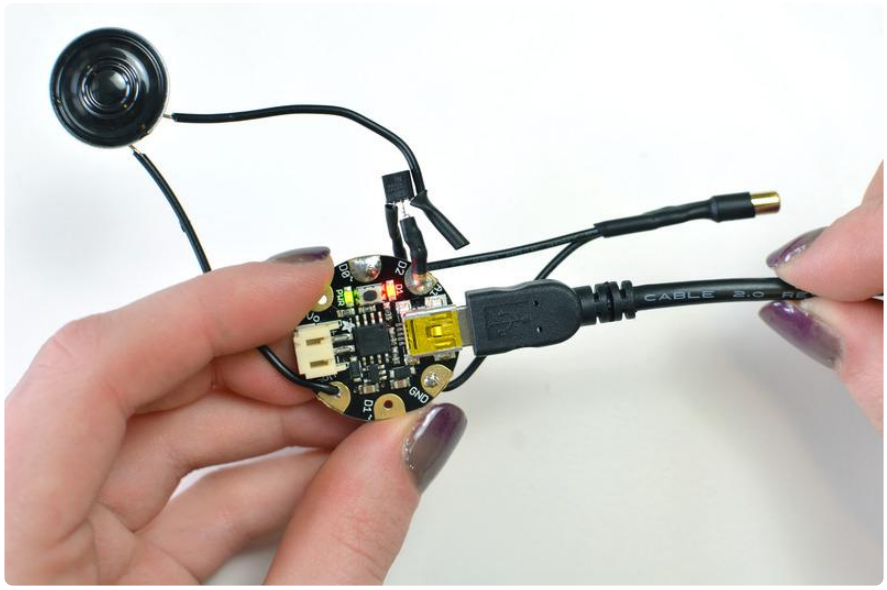
Clip off any excess wire.





Your circuit is complete! Look it over, comparing it to the circuit diagram and double checking your solder joints.

Arduino Code



The Arduino code presented below works equally well on all versions of GEMMA: v1, v2 and M0. But if you have an M0 board, consider using the CircuitPython code on the next page of this guide, no Arduino IDE required!

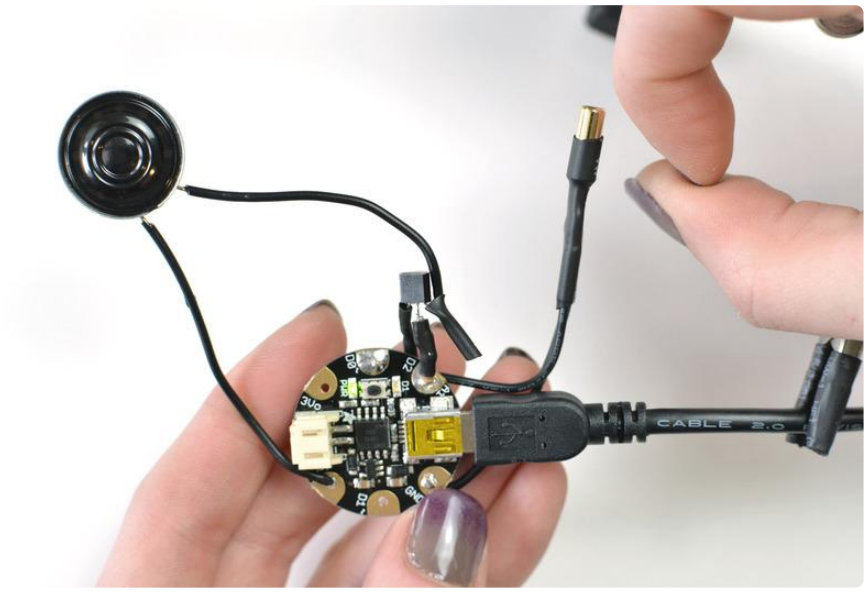
Plug GEMMA into your computer with a USB cable. If you've never programmed GEMMA before, you'll have to download and install the special Adafruit version of the Arduino IDE, which you can find in the Introducing GEMMA guide (). Once you're able to load programs onto GEMMA successfully, load up the following code:

```
// SPDX-FileCopyrightText: 2018 Becky Stern for Adafruit Industries
// SPDX-FileCopyrightText: 2018 T Main for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
Chirp Owl written by Becky Stern and T Main for Adafruit Industries
Tutorial: http://learn.adafruit.com/chirping-plush-owl-toy/

Includes animal sounds by Anne Barela
http://learn.adafruit.com/adafruit-trinket-modded-stuffed-animal/animal-sounds

based in part on Debounce
 created 21 November 2006
 by David A. Mellis
 modified 30 Aug 2011
 by Limor Fried
 modified 28 Dec 2012
 by Mike Walters

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/Debounce
 */

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 0;    // the number of the pushbutton pin
const int speakerPin = 2;      // the number of the LED pin
const int ledPin = 1;

// Variables will change:
int ledState = HIGH;         // the current state of the output pin
int buttonState;             // the current reading from the input pin
int lastButtonState = LOW;   // the previous reading from the input pin

// the following variables are long's because the time, measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
long lastDebounceTime = 0;  // the last time the output pin was toggled
long debounceDelay = 50;    // the debounce time; increase if the output flickers

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(speakerPin, OUTPUT);
  //digitalWrite(speakerPin, HIGH);
  digitalWrite(ledPin, LOW);
  //digitalWrite(buttonPin, HIGH);
  // set initial LED state
  //digitalWrite(speakerPin, ledState);
  //Serial.begin(9600);
}

void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed the button
  // (i.e. the input went from LOW to HIGH),  and you've waited
  // long enough since the last press to ignore any noise:

  // If the switch changed, due to noise or pressing:
```

```
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer
    // than the debounce delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

      // only toggle the LED if the new button state is HIGH
      //Serial.println("chirp");
        chirp(); // change this line to change animal sound
        //meow();
        //meow2();
        ////mew();
        //ruff();
        //arf();
    }
  }

  // set the LED:
  //digitalWrite(speakerPin, ledState);

  // save the reading.  Next time through the loop,
  // it'll be the lastButtonState:
  lastButtonState = reading;
}


// Generate the Bird Chirp sound
void chirp() {
 for(uint8_t i=200; i>180; i--)
   playTone(i,9);
}

// Play a tone for a specific duration.  value is not frequency to save some
//   cpu cycles in avoiding a divide.
void playTone(int16_t tonevalue, int duration) {
 for (long i = 0; i < duration * 1000L; i += tonevalue * 2) {
   digitalWrite(speakerPin, HIGH);
   delayMicroseconds(tonevalue);
   digitalWrite(speakerPin, LOW);
   delayMicroseconds(tonevalue);
 }
}

void meow() {  // cat meow (emphasis ow "me")
  uint16_t i;
  playTone(5100,50);        // "m" (short)
  playTone(394,180);        // "eee" (long)
  for(i=990; i<1022; i+=2)  // vary "ooo" down
     playTone(i,8);
  playTone(5100,40);        // "w" (short)
}

void meow2() {  // cat meow (emphasis on "ow")
  uint16_t i;
  playTone(5100,55);       // "m" (short)
  playTone(394,170);       // "eee" (long)
  delay(30);               // wait a tiny bit
  for(i=330; i<360; i+=2)  // vary "ooo" down
     playTone(i,10);
  playTone(5100,40);       // "w" (short)
}
```

```
void mew() {  // cat mew
  uint16_t i;
  playTone(5100,55);       // "m"   (short)
  playTone(394,130);       // "eee" (long)
  playTone(384,35);        // "eee" (up a tiny bit on end)
  playTone(5100,40);       // "w"   (short)
}

void ruff() {   // dog ruff
  uint16_t i;
  for(i=890; i<910; i+=2)     // "rrr"  (vary down)
     playTone(i,3);
  playTone(1664,150);         // "uuu" (hard to do)
  playTone(12200,70);         // "ff"  (long, hard to do)
}

void arf() {    // dog arf
  uint16_t i;
  playTone(890,25);          // "a"    (short)
  for(i=890; i<910; i+=2)    // "rrr"  (vary down)
     playTone(i,5);
  playTone(4545,80);         // intermediate
  playTone(12200,70);        // "ff"   (shorter, hard to do)
}
```

Now when you flick or shake the tilt ball switch, the circuit should chirp! If it doesn't, check your connections with a multimeter and be sure your software is configured properly for programming GEMMA. Mike Barela has written additional animal sounds ( ), in case you want to meow or bark instead of chirp!



CircuitPython Code

GEMMA M0 boards can run CircuitPython — a different approach to programming compared to Arduino sketches. In fact, CircuitPython comes factory pre-loaded on GEMMA M0. If you've overwritten it with an Arduino sketch, or just want to learn the basics of setting up and using CircuitPython, this is explained in the Adafruit GEMMA M0 guide ().

These directions are specific to the “M0” GEMMA board. The original GEMMA with an 8-bit AVR microcontroller doesn’t run CircuitPython…for those boards, use the Arduino sketch on the “Arduino code” page of this guide.

Below is CircuitPython code that works similarly (though not exactly the same) as the Arduino sketch shown on a prior page. To use this, plug the GEMMA M0 into USB…it should show up on your computer as a small flash drive…then edit the file “code.py” with your text editor of choice. Select and copy the code below and paste it into that file, entirely replacing its contents (don’t mix it in with lingering bits of old code). When you save the file, the code should start running almost immediately (if not, see notes at the bottom of this page).

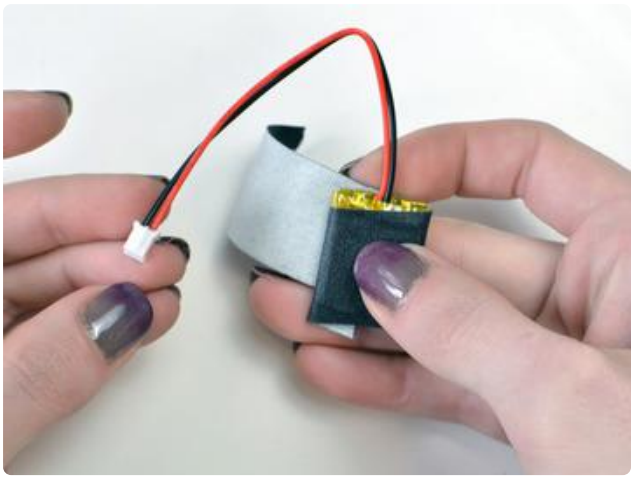
If GEMMA M0 doesn’t show up as a drive, follow the GEMMA M0 guide link above to prepare the board for CircuitPython.

```
# SPDX-FileCopyrightText: 2018 Becky Stern for Adafruit Industries
# SPDX-FileCopyrightText: 2018 T Main for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Chirp Owl written by Becky Stern and T Main for Adafruit Industries
# Tutorial: http://learn.adafruit.com/chirping-plush-owl-toy/

# Includes animal sounds by Anne Barela
# http://learn.adafruit.com/adafruit-trinket-modded-stuffed-animal/animal-sounds
# based in part on Debounce
#  created 21 November 2006
#  by David A. Mellis
#  modified 30 Aug 2011
#  by Limor Fried
#  modified 28 Dec 2012
#  by Mike Walters
#  CircuitPython Port 2018
#  by Mikey Sklar
#  This example code is in the public domain.
#  http://www.arduino.cc/en/Tutorial/Debounce

import time
import board
import digitalio

# setup for vibration sensor
motion = digitalio.DigitalInOut(board.D0)
motion.direction = digitalio.Direction.INPUT
motion.pull = digitalio.Pull.UP

# setup for speaker output
speaker = digitalio.DigitalInOut(board.D2)
speaker.direction = digitalio.Direction.OUTPUT

def chirp():
    for i in range(200,180,-1):
        play_tone(i,9)

# emphasis ow "me"
def meow():
    play_tone(5100,50)           # "m" (short)
    play_tone(394,180)           # "eee" (long)
```

```
    for i in range(990,1022,2): # vary "ooo" down
        play_tone(i,8)
    play_tone(5100,40)           # "w" (short)

# cat meow (emphasis on "ow")
def meow2():
    play_tone(5100,55)          # "m" (short)
    play_tone(394,170)          # "eee" (long)
    time.sleep(0.03)            # wait a tiny bit
    for i in range(990,1022,2): # vary "ooo" down
        play_tone(i,8)
    play_tone(5100,40)          # "w" (short)

# dog ruff
def ruff():
    for i in range(890,910,2):  # "rrr"  (vary down)
        play_tone(i,3)
    play_tone(1664,150)         # "uuu" (hard to do)
    play_tone(12200,70)         # "ff"  (long, hard to do)

# dog arf
def arf():
    play_tone(890,25)             # "a"    (short)
    for i in range(890,910,2):    # "rrr"  (vary down)
        play_tone(i,5)
    play_tone(4545,80)            # intermediate
    play_tone(12200,70)           # "ff"   (shorter, hard to do)

def play_tone(tone_value, duration):
    microseconds = 10 ** 6              # duration divider, convert to microseconds

    for i in range(0, duration):
        i += tone_value * 2
        speaker.value = True
        time.sleep(tone_value / microseconds)
        speaker.value = False
        time.sleep(tone_value / microseconds)

# loop forever...
while True:

    # reverse logic for motion pins
    if not motion.value:
        # bird chirp noise
        # comment out chirp and uncomment a different sound below
        # for more animal noises
        chirp()
        # meow()
        # meow2()
        # ruff()
        # arf()
        time.sleep(.5)                  # leave some time to complete rotation
```

Assemble Owl

Try out your circuit on battery power by plugging it into GEMMA's JST port.

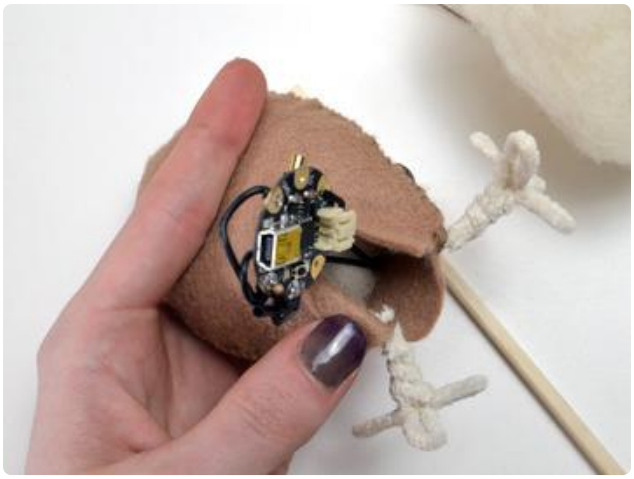


Tape up the battery to protect it and protect the wire connections.

Start making the owl according to the paper instructions included in the box.

Cut a slit up the back body piece to make a path for the speaker wires.

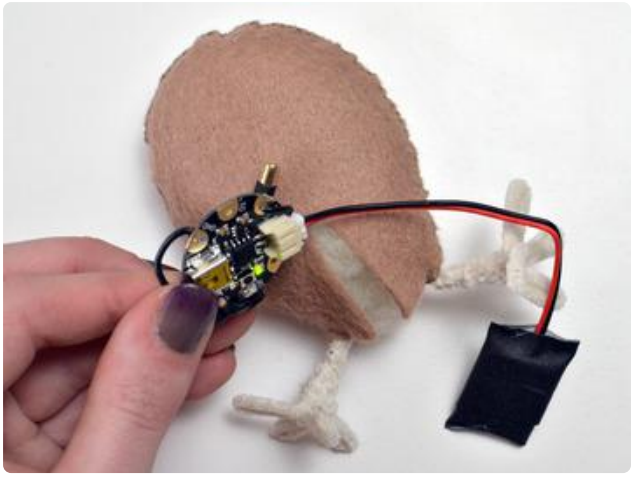
Arrange your circuit as shown.



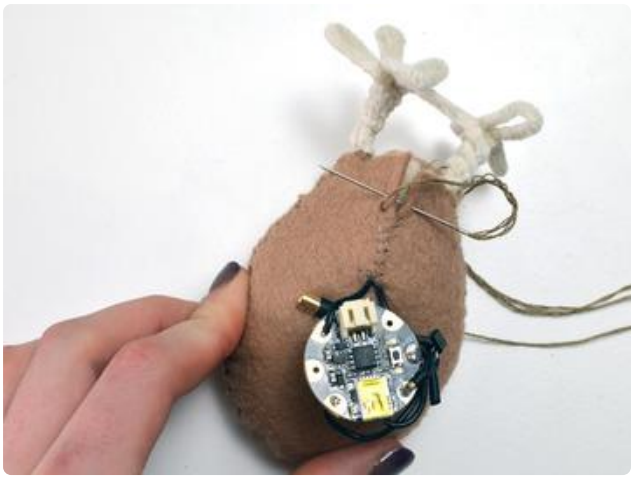
Begin stuffing the body, then insert the speaker just behind the front of the body.

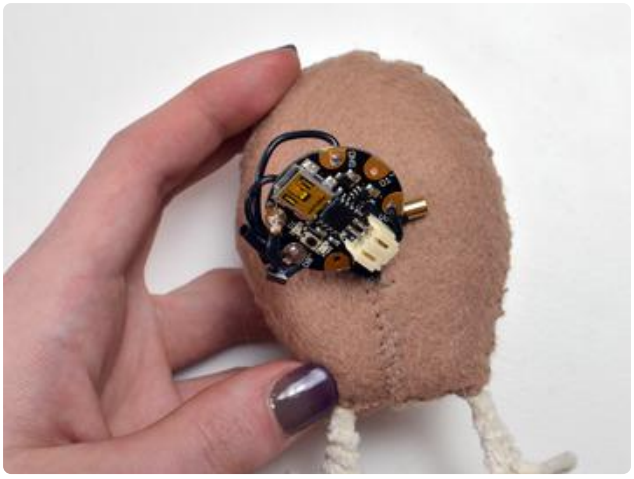
Bend the wires so they route straight out the back of the owl body, at the top of the slit you made in the last step.

Continue and finish stuffing the owl. Plug in the battery to test out the loudness of your circuit. If it's not as loud as you like, make sure the speaker is flush up against the front of the body and not insulated by any stuffing.

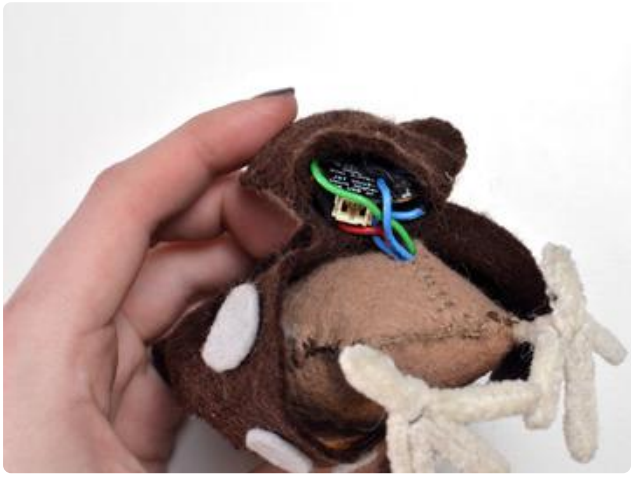


Unplug the battery to finish constructing the owl.





Stitch up the slit and bottom of the owl and continue making the owl as per the paper instructions.



Because the GEMMA sits inside the tail of the owl, we needed to cut an additional tail piece from the brown felt included in the owl kit, then sewed it to the original tail piece to make a pocket.

The battery sits inside the cape of the owl, and GEMMA's JST port conveniently faces the opening for convenient disconnection. When the battery runs down you will need to disconnect and remove the battery, and charge it using the MicroLipo charger. (Lithium batteries are not made to be recharged while inside the owl, as the battery may get hot. So that's why you need to remove to recharge). Also the GEMMA v1, v2 and M0 do not support recharging Lithium cells.

Because the wires enter the tail, this mod has an added advantage of allowing the stiffened tail to stabilize the toy when it stands!