Flora Brake Light Backpack

Created by Becky Stern

https://learn.adafruit.com/flora-brakelight-backpack

Last updated on 2023-08-29 02:15:46 PM EDT
Overview

My friend asked if I could help make him more visible on his motorcycle, so Tyler Cooper and I designed a brake light backpack that uses a Flora accelerometer to auto-detect braking behavior on a motorcycle or scooter. It also has turn signals triggered by a simple wireless module, and can still leaves plenty of room for your stuff in the pack!

This is a complex project with many components! Here are some recommended prerequisite guides:

- [Getting Started with FLORA ()](http://adafru.it/)
- [Flora Accelerometer ()](http://adafru.it/)

Tools & Supplies

Bill of materials:

- two strands of [12mm flat weather-resistant pixels](http://adafru.it/738)
- [Flora main board](http://adafru.it/659)
- [Flora accelerometer](http://adafru.it/1247)
- [small weather-resistant enclosure](http://adafru.it/903)
- [3xAA battery pack](http://adafru.it/771) (not pictured)
- [Tactile on/off switch with leads](http://adafru.it/1092)
- [Momentary simple RF receiver - 315MHz](http://adafru.it/1096)
- [Keyfob RF remote control - 315MHz](http://adafru.it/1095)
- [4-pin JST SM Plug + Receptacle Cable Set](http://adafru.it/578) (not pictured)
- [JST 2-pin cable](http://adafru.it/261) (not pictured)
• four 4.7K ohm resistors
• Sugru (http://adafruit.it/437)
• Backpack - we're using the Black Diamond Bullet ()
• Zipties
• Gaffers tape
• Velcro tape

Multimeter

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

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

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

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

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

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 Flora (see here [1]).

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! [2]

Solder

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)

Click here to buy a spool of lead-free solder. (http://adafru.it/734)
Solder wick will help correct any mistakes.

Click here to buy some. (http://adafru.it/149)

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)

Heat shrink tubing is the special sauce to keep your circuit out of trouble if you get caught in the rain.

Click here to buy some. (http://adafru.it/344)
Sharp scissors are a must! You'll also need a large piece of cardboard, a marker and ruler, and some tailor's chalk.

Don't forget your wire strippers (http://adafru.it/527), pliers (http://adafru.it/146), and flush snips (http://adafru.it/152) and some hookup wire (http://adafru.it/290)!
You'll use a [needle](http://adafruit.it/615) and thread to stitch the pixels to the bag.
An on/off switch with long leads connects the battery holder to a 2-pin JST connector, allowing the rider to turn the bag on and off while wearing the bag.

The Flora accelerometer connects to Flora's 3.3v, SDA, SCL, and GND pins, all in a row, and will detect when the rider is braking.

The pixel strand connects to VBATT (red wire), GND (blue wire), D9 (green) and D10 (yellow).

VBATT also connects to +5V on the RF module. GND -> GND and each signal pin is connected to a Flora digital input via a voltage divider comprised of two 4.7K ohm
resistors to reduce the signal voltage to ~3V.

The remote buttons trigger the RF modules signal pins to go HIGH. Two buttons are shown for use as turn signals, but you can connect up the remaining buttons in the same way.

**Control Circuit**

To wire up the battery power supply for the circuit, tin the long leads of your on/off switch with solder.

You may also wish to extend the switch leads even longer with pieces of wire.

Tin the leads of your 2-pin JST cable. Solder one switch lead to the JST cable's red wire, and the other switch lead to the battery pack's red wire. If your battery pack doesn't have different colored wires, use your multimeter to determine which is the positive lead.

Also solder the JST cable's black wire to the battery pack's black wire.
Use a 3xAA battery pack for this project. If you don't have one handy, you can bridge across one of the compartments on a 4xAA battery pack-- just solder a wire to each end.
Accelerometer

Attach the Flora accelerometer according to the circuit diagram. Notice that all four pins necessary for this i2c sensor are in a row, making wiring very convenient!

Affix the Flora board and accelerometer to the battery pack with Velcro tape. Likewise between the battery pack and plastic enclosure.

RF Module

These simple wireless receiver/remote pairs are great for adding a few wireless inputs to your project, but they operate at 5V, which doesn't match Flora's 3V i/o. So build a simple voltage divider with two 4.7K ohm resistors!
To attach to Flora, twist the resistor leads together and solder to a digital input pin. Repeat with more pairs of resistors for each button on your remote or as many as you’d like to use.

One resistor in each pair will be soldered to ground, and the other will connect to the RF module’s output.

Female jumper wires (http://adafruit.it/825) can come in handy for attaching to the RF module’s header pins. The pin marked +5V gets connected to VBATT on Flora, and GND goes go GND.

Pixel connector

Plug an extender onto the input side of a strand of weather-resistant LED pixels.

Run the connector cable through a small hole bored in the plastic weather-resistant enclosure.
Use colorful heatshrink tubing to mark the pixel connectors wires, corresponding to the colored wires in the pixel strand. You could also use pieces of taped marked with a letter-- this makes it a lot easier to attach the wires to the Flora correctly.

Connect the four pixel connector wires as per this diagram - red to VBATT, blue to GND, yellow to D10 and green to D9.
Switch

These tactile on/off switches are perfect for toggling power on our backpack, but the switch isn't inherently waterproof. Using a bit of plastic to prevent the button from being clogged, wrap the switch in Sugru and let it cure for 24 hours. Wear gloves to prevent fingerprint marks.
Use a zip tie for strain relief on any wires exiting the enclosure.

For maximum range between the RF remote/receiver pair, uncoil the receiver's antenna wire so it runs along the outer edge of the enclosure.
More Sugrue helps waterproof the hole in the enclosure.

**LED Pixels**

Mark out a double diamond design with tailor's chalk on your backpack.

Transfer that sketch to a piece of cardboard using a ruler and marker. Mark where each LED pixel will go.
Punch holes in the cardboard with an awl.

Install your pixels into the cardboard jig.

Since the design is longer than 1 strand (25 pixels-- our design is 36), solder on or plug in another strand of pixels.
Use zipties to cinch up the wires between pixels so the design has some structure.

Follow the tailor's chalk markings on your backpack and install the LED pixels by cutting a hole and heat-sealing the nylon edge.

Attach the pixels by stitching the wires to the front surface of the bag. Knot and cut the thread at each junction so that if one stitch comes undone, the rest of the design isn't compromised by an unraveling thread.

You can also use tape to bridge across the backs of the pixels. Remove the pixels from the jig.
My install didn't go exactly as planned, and my pixel design had to be compromised. I adjusted my diagram so I'd know which pixels were which when programming animations. It's challenging to install so many pixels inside such a small space, but it was comforting knowing I could fix any layout problems later, in the code.
Decide where you'd like your power toggle switch to go, make a slice in the bag for the switch to come through, heat seal the nylon, then stitch the switch in place!

Prop up the bag and connect the USB cable (keeping the battery power off) for programming.
The Code

This code and project is a proof of concept. The code works reasonably well, but needs a fair amount of work to make it a reliable.

When we first starting working on this project, I thought that the code would be the easy part. As I started working on the code, I quickly realized there are a lot of external factors that can give you erratic results from the accelerometer. Factors such as quick acceleration, bumpy roads, or slow braking gave me results that I wasn't prepared for.

Overall, though, this was a lot of fun to code, and it is a project that I will continue to tinker with until I am happy with the results. Here is the code that I came up with, and how it works.

You'll need to download and install the Adafruit_LSM303 library - we have a tutorial on installing arduino libraries here

Select Download Project Zip to get the Flora_Brakelight_Backpack.ino file.

```cpp
// SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Wire.h>
#include <Adafruit_LSM303.h>
#include <SPI.h>
#include <Adafruit_WS2801.h>

Adafruit_LSM303 lsm;
#define BRAKETHRESHOLD 350
#define BRAKETIMETHRESHOLD 200

int dataPin  = 2;    // Yellow wire on Adafruit Pixels
int clockPin = 3;    // Green wire on Adafruit Pixels
const int cPin = 11;
const int dPin = 6;

// Set the first variable to the NUMBER of pixels. 32 = 32 pixels in a row
// The LED strips are 32 LEDs per meter but you can extend/cut the strip
Adafruit_WS2801 strip = Adafruit_WS2801(36, dataPin, clockPin);

int start = 0;
int prevX = 0;
int currentX = 0;
int cState = 0;
int dState = 0;
long brakeTime = 0;

void setup()
{
}
Serial.begin(9600);

// Start up the LED strip
strip.begin();
// Update the strip, to start they are all 'off'
strip.show();
// Try to initialise and warn if we couldn't detect the chip
if (!lsm.begin())
{
  Serial.println("Oops ... unable to initialize the LSM303. Check your wiring!");
  while (1);
}
pinMode(cPin, INPUT);
pinMode(dPin, INPUT);

void loop()
{
  check_switches();    // when we check the switches we'll get the current state
  lsm.read();
  currentX = abs(lsm.accelData.x);
  if (start == 0){
    prevX = currentX;
    start = 1;
  }
  int i = currentX - prevX;
  if (abs(i) > BRAKETHRESHOLD) {
    brakeTime = millis();
    int strikes = 0;
    while ((abs(i) > BRAKETHRESHOLD) && (strikes < 3)) {
      if (abs(i) < BRAKETHRESHOLD) {
        strikes = strikes + 1;
      }
      lsm.read();
      currentX = abs(lsm.accelData.x);
      i = currentX - prevX;
      if ((millis() - brakeTime) > BRAKETIMETHRESHOLD) {
        brakeLights(Color(255,0,0),250);
        while (abs(i) > BRAKETHRESHOLD) {
          lsm.read();
          currentX = abs(lsm.accelData.x);
          i = currentX - prevX;
          Serial.println(i);
          delay(100);
        }
        hideAll();
        brakeTime = millis();
        i = 0;
        lsm.read();
        currentX = abs(lsm.accelData.x);
      }
    }
  }
  prevX = currentX;
  delay(200);
}

void check_switches()
{
  cState = digitalRead(cPin);
  dState = digitalRead(dPin);
  if (cState == HIGH) {

// left blinker
Serial.println("left blink on");
hideAll();
leftTurn(Color(255,63,0),250);
delay(300);
Serial.println("left blink off");
hideAll();
delay(300);
}

if (dState == HIGH) {
    // right blinker
    Serial.println("right blink on");
hideAll();
rightTurn(Color(255,63,0),250);
delay(300);
Serial.println("right blink off");
hideAll();
delay(300);
}

void leftTurn(uint32_t c,uint8_t wait){
    innerLeftBottom(c);
    innerLeftTop(c);
    strip.show();
delay(wait);
hideAll();
    outerLeftTop(c);
    outerLeftBottom(c);
    strip.show();
delay(wait);
hideAll();
}

void rightTurn(uint32_t c,uint8_t wait){
    innerRightBottom(c);
    innerRightTop(c);
    strip.show();
delay(wait);
hideAll();
    outerRightTop(c);
    outerRightBottom(c);
    strip.show();
delay(wait);
hideAll();
}

void brakeLights(uint32_t c, uint8_t wait){
    innerRightBottom(c);
    innerRightTop(c);
    innerLeftBottom(c);
    innerLeftTop(c);
    strip.show();
delay(wait);
hideAll();
    outerLeftTop(c);
    outerLeftBottom(c);
    outerRightTop(c);
    outerRightBottom(c);
    strip.show();
delay(wait);
hideAll();
}

/* Helper functions */

//Input a value 0 to 384 to get a color value.
//The colours are a transition r - g - b - back to r

void outerRightBottom(uint32_t c){
  for (int i=0; i < 5; i++) {
    strip.setPixelColor(i, c);
  }
}
void outerRightTop(uint32_t c){
  for (int i=5; i < 10; i++) {
    strip.setPixelColor(i, c);
  }
}
void innerRightTop(uint32_t c){
  for (int i=10; i < 14; i++) {
    strip.setPixelColor(i, c);
  }
}
void innerRightBottom(uint32_t c){
  for (int i=14; i < 18; i++) {
    strip.setPixelColor(i, c);
  }
}
void innerLeftBottom(uint32_t c){
  for (int i=18; i < 22; i++) {
    strip.setPixelColor(i, c);
    strip.show();
  }
}
void innerLeftTop(uint32_t c){
  for (int i=22; i < 26; i++) {
    strip.setPixelColor(i, c);
  }
}
void outerLeftTop(uint32_t c){
  for (int i=26; i < 31; i++) {
    strip.setPixelColor(i, c);
  }
}
void outerLeftBottom(uint32_t c){
  for (int i=31; i < 36; i++) {
    strip.setPixelColor(i, c);
  }
}
void hideAll(){
  for(int i = 0; i > strip.numPixels();i++){
    strip.setPixelColor(i,Color(0,0,0));
  }
  strip.show();
}

// Create a 24 bit color value from R,G,B
uint32_t Color(byte r, byte g, byte b)
{
  uint32_t c;
  c = r;
  c <<= 8;
  c |= g;
  c <<= 8;
  c |= b;
  return c;
}
# Code Walkthrough

```cpp
#include &lt;Wire.h&gt;
#include &lt;Adafruit_LSM303.h&gt;
#include &lt;SPI.h&gt;
#include &lt;Adafruit_WS2801.h&gt;

Adafruit_LSM303 lsm;
#define BRAKETHRESHOLD 350
#define BRAKETIMETHRESHOLD 200
```

The most important part of the code is the LSM303 library, the WS2801 library, and the `BRAKETHRESHOLD` and `BRAKETIMETHRESHOLD` values. You will need to install the LSM303 library, which can be found here, and the WS2801 library, which can be found here.

The `BRAKETHRESHOLD` value will need to be tweaked to your liking. This value looks at the amount of force the accelerometer measures before it considers it worth investigating further. Similarly, the `BRAKETIMETHRESHOLD` value is the amount of time that the force exists before it will trigger the brake lights. This is an important part of the code, as it allows us to ignore things like big bumps in the road.

From there, the code is quite simple. If you press and hold the C button on the remote, the left blinker will activate on the backpack. If you press and hold the D button on the remote, the right blinker will activate on the backpack. Release either button to stop blinking. You could easily modify the sketch to press the button to turn on, and press again to turn off....or simply use the toggle type RF receiver which you can buy here (http://adafruit.it/1097).
Wear it!