Kitty's Flowers - Bluetooth Wearable Brooches - Art by Physicist

Created by Kitty Yeung

https://learn.adafruit.com/flowers-bluetooth-wearable-brooches-art-by-physicist

Last updated on 2023-08-29 04:48:11 PM EDT
# Table of Contents

**Overview**  
- Specifications

**Applications**  
**Pinouts**  
- Power Connections
- Sensors and output components

**Tutorial**  
10

**Arduino IDE Setup**  
- Arduino IDE Download

**Sample Code for Vibration Motor**  
11

**Sample Code for Touch**  
11

**Sample Code for the RGB LEDs**  
12

**Sample Code for the Accelerometer**  
14

**Sample Code for BLE**  
15
Overview

A pair of brooches that are not only decorative, but also interactive and educational! These flower-shaped PCBs are wearable brooches that each functions like an Arduino UNO with integrated components all on the same board and connects to each other through Bluetooth.

Jointly created by Art by Physicist (Dr. Kitty Yeung) and DFRobot, they contain an integrated power management system, RGB LED lights, an accelerometer, a touch sensor, and a vibrating motor. The brooches support Arduino and Scratch programming and can be used as the core intelligent control unit for wearable products, interactive art installations, and new accessories. The brooches come in matched pairs.
Specifications

- Microcontroller: ATmega328
- BLE chip type: TI CC2540
- Supports Bluetooth HID
- Debug Bluetooth module through AT command
- Serial port communication
- USB BLE firmware update
- Power supply interface: USB / 3.7V lithium battery, JST connector
- External power supply range: 7-12V
- Bootloader: Arduino UNO
- Size: 70mm x 70mm
- Weight: 30g

Kitty's Flowers - Pair of Bluetooth Wearable Brooches
Kitty's Flowers is a pair of beautifully designed Bluetooth-wearable brooches, which can be programmed to interact with one another. The brooches are perfect for...
https://www.adafruit.com/product/5215
Applications

With many components integrated on the boards, you can program them to achieve various functionalities. Whether you are wearing a pair as a parent with your child, or wear them between siblings, friends or lovers, you can add programs through the serial port to the brooches for all kinds of fun interactions with your loved ones.

The video demonstrates the factory setting, which sync the two boards when they are near each other and detect the distance between them. If it goes beyond 2m, the central board vibrates with the LEDs blinking in warning mode. This can be used for parents to keep their children by their side.

You may also get inspired by the ideas in this Hackster.io project(). A group of awesome makers got together to hack on the Flowers Bluetooth brooches during the holidays and proposed many use cases of the flowers, including detecting fall by elderly people, hide-and-seek, non-verbal communication, and even using them as counters to count number of spoons() and practice mindful breathing()!
See more ideas in cartoons captured by Tanya Fish in the Hackster.io project.

Here is a recording of the holiday hack. What will you use these flowers for?

**Pinouts**

Detailed schematics of the PCB can be downloaded here: Kitty Flower Schematic
Power Connections

- **USB C connector** - This is used for power and data. Connect to your computer via a USB C cable to update firmware and edit code.
- **LiPoly Battery connector** - This 2-pin JST PH connector allows you to plug in lipoly batteries to power the Flower. The Flower is also capable of charging batteries plugged into this port via USB.

You will need to get the lipoly batteries separately to power the Flowers. These are some good options that are small and have short connecting cables so you can hide the batteries behind the flowers.

Lithium Ion Polymer Battery - 3.7v 100mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...

https://www.adafruit.com/product/1570
Lithium Ion Polymer Battery Ideal For Feathers - 3.7V 400mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/3898

Lithium Ion Polymer Battery with Short Cable - 3.7V 350mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/4237

Lithium Ion Polymer Battery with Short Cable - 3.7V 420mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/4236
Sensors and output components

On the back side of the PCB, you can find the accelerometer and vibration motor.

- I2C - This is for LIS2DH 3-Axis Accelerometer.
- D5 - This is the pin for programming the vibration motor.

On the front side of the PCB, you can find seven NeoPixel RGB LEDs that decorate the outskirt of the flower petals. The golden flower core/stamen is the touch sensor.

- D6 - This is the ping for programming the touch sensor.
- D9 - This is the pin for programming the NeoPixel RGB LEDs.
Tutorial

In the following subpages, you will find examples of how to program each of the components. Try to mix-and-match the sample code to achieve various applications.

Here's a demonstration video to get you started.

Arduino IDE Setup

A Flower PCB functions essentially as an Arduino UNO. This page covers getting your Arduino IDE set up to include your board.

Arduino IDE Download

The first thing you will need to do is to download the latest release of the Arduino IDE.

Download and install it to your computer.

Once installed, open the Arduino IDE. Choose Arduino UNO as your board after plugging in a Flower PCB.
Sample Code for Vibration Motor

This sample code shows you how to turn on and off the vibration motor.

```c
#define VMPIN 5     //Vibration motor pin

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(VMPIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(VMPIN, HIGH);   // turn the Motor on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(VMPIN, LOW);    // turn the Motor off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Sample Code for Touch

This sample code shows you how to program the touch sensor to sense your touch on the flower core/stamen. Turn on the Serial Monitor to observe the touch sensing.

```c
#define TOUCHPIN 6

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(115200);
  // make the pushbutton's pin an input:
```
Sample Code for the RGB LEDs

To use the NeoPixels, you need to first install its library, if you haven't already. You can simply search for "Adafruit NeoPixel" in the Arduino Library Manager as described here and jump right back to continue this tutorial. For background reading, here's how you can install a library on Arduino in general. If you need the NeoPixel library source code, here's its GitHub page.

The following code shows you how to create different lighting effects with the LEDs. There are lots of tutorials on programming NeoPixels, for example, on Adafruit. Try them and find your favorite effect to put on your Flowers.

```c
#include <Adafruit_NeoPixel.h>

#define PIN 9     // The signal pin connected with Arduino
#define LED_COUNT 7 // the amount of the leds of your strip

// Create an instance of the Adafruit_NeoPixel class called "leds". // That'll be what we refer to from here on...
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_COUNT, PIN, NEO_GRB + NEO_KHZ800);

void setup()
{
  leds.begin();  // Call this to start up the LED strip.
  clearLEDs();   // This function, defined below, turns all LEDs off...
  leds.show();   // ...but the LEDs don't actually update until you call this.
}

void loop()
{
  for (int i=0; i<LED_COUNT; i++)
  {
    rainbow(i);
    delay(10);  // Delay between rainbow slides
  }
}

// Sets all LEDs to off, but DOES NOT update the display; // call leds.show() to actually turn them off after this.
void clearLEDs()
{
  for (int i=0; i<LED_COUNT; i++)
  {
    leds.setPixelColor(i, 0);
  }
}
```

©Adafruit Industries
// Prints a rainbow on the ENTIRE LED strip.
// The rainbow begins at a specified position.
// ROY G BIV!
void rainbow(byte startPosition)
{
    // Need to scale our rainbow. We want a variety of colors, even if there
    // are just 10 or so pixels.
    int rainbowScale = 192 / LED_COUNT;

    // Next we setup each pixel with the right color
    for (int i=0; i<LED_COUNT; i++)
    {
        // There are 192 total colors we can get out of the rainbowOrder function.
        // It'll return a color between red-&gt;orange-&gt;green-&gt;...-&gt;violet
        for 0-191.
        leds.setPixelColor(i, rainbowOrder((rainbowScale * (i + startPosition)) %
192));
    }
    // Finally, actually turn the LEDs on:
    leds.show();
}

// Input a value 0 to 191 to get a color value.
// The colors are a transition red-&gt;yellow-&gt;green-&gt;aqua-&gt;blue-
&gt;fuchsia-&gt;red...
// Adapted from Wheel function in the Adafruit_NeoPixel library example sketch
uint32_t rainbowOrder(byte position)
{
    // 6 total zones of color change:
    if (position &lt; 31)  // Red -&gt; Yellow (Red = FF, blue = 0, green goes 00-
FF)
    {
        return leds.Color(0xFF, position * 8, 0);
    }
    else if (position &lt; 63)  // Yellow -&gt; Green (Green = FF, blue = 0, red
goes FF-&gt;00)
    {
        position -= 31;
        return leds.Color(0xFF - position * 8, 0xFF, 0);
    }
    else if (position &lt; 95)  // Green-&gt;Aqua (Green = FF, red = 0, blue goes
00-&gt;FF)
    {
        position -= 63;
        return leds.Color(0, 0xFF, position * 8);
    }
    else if (position &lt; 127)  // Aqua-&gt;Blue (Blue = FF, red = 0, green goes
FF-&gt;00)
    {
        position -= 95;
        return leds.Color(0, 0xFF - position * 8, 0xFF);
    }
    else if (position &lt; 159)  // Blue-&gt;Fuchsia (Blue = FF, green = 0, red
goes 00-&gt;FF)
    {
        position -= 127;
        return leds.Color(position * 8, 0, 0xFF);
    }
    else  //160 &lt;position&lt; 191  Fuchsia-&gt;Red (Red = FF, green = 0, blue
goes FF-&gt;00)
    {
        position -= 159;
        return leds.Color(0xFF, 0x00, 0xFF - position * 8);
    }
}
Sample Code for the Accelerometer

Similarly to the NeoPixels, you will need a library to use the accelerometer. This time, you will be downloading a package from DFRobot:

![Download DFRobot LIS2DH12 Accelerometer Library](image)

and follow the [instruction here](#) to install the library.

The following code shows you how to get the accelerometer to detect motion. Open the Serial Monitor to see the acceleration reading in the x, y and z directions.

```cpp
#include <Wire.h>
#include <DFRobot_LIS2DH12.h>

DFRobot_LIS2DH12 LIS; //Accelerometer

void setup() {
  Wire.begin();
  Serial.begin(115200);
  while (!Serial);
  delay(100);

  // Set measurement range
  // Ga: LIS2DH12_RANGE_2GA
  // Ga: LIS2DH12_RANGE_4GA
  // Ga: LIS2DH12_RANGE_8GA
  // Ga: LIS2DH12_RANGE_16GA
  while (LIS.init(LIS2DH12_RANGE_16GA) == -1) { //Equipment connection exception or I2C address error
    Serial.println("No I2C devices found");
    delay(1000);
  }
}

void loop() {
  acceleration();
}

/*! 
@brief Print the position result.
*/

void acceleration(void) {
  int16_t x, y, z;
  delay(100);
  LIS.readXYZ(x, y, z);
  LIS.mgScale(x, y, z);
  Serial.print("Acceleration x: "); //print acceleration
  Serial.print(x);
  Serial.print(" mg \ty: ");
  Serial.print(y);
  Serial.print(" mg \tz: ");
  Serial.print(z);
  Serial.println(" mg");
}
Sample Code for BLE

Finally you can connect the two Flowers to have them interact!

You can define which Flower to be the central and which to be the peripheral. You can download the sample code below and upload KittyMother as the central and KittyChild as the peripheral, separately. The two demos complete the pairing and will automatically set the central/peripheral status of the motherboards, and then read the Bluetooth RSSI field strength value to determine the distance between each other.

<table>
<thead>
<tr>
<th>Acceleration x: -62 mg</th>
<th>y: 0 mg</th>
<th>z: 1015 mg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration x: -46 mg</td>
<td>y: 0 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1015 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 15 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1015 mg</td>
</tr>
<tr>
<td>Acceleration x: -46 mg</td>
<td>y: 0 mg</td>
<td>z: 1015 mg</td>
</tr>
<tr>
<td>Acceleration x: -78 mg</td>
<td>y: 0 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 15 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1015 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 15 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 15 mg</td>
<td>z: 1000 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: -15 mg</td>
<td>z: 1031 mg</td>
</tr>
<tr>
<td>Acceleration x: -62 mg</td>
<td>y: 0 mg</td>
<td>z: 1015 mg</td>
</tr>
</tbody>
</table>

Download the BLE Central/Peripheral Demo Code

There you have it - examples of how to program each component. Now you can combine them to achieve different applications. See how others have used the Flowers to detect fall using the accelerometer, detect distance using the Bluetooth RSSI field strength, count the number using the touch sensor, control breathing using the microcontroller timer and LEDs and more. Show the world how you will be using these super integrated beautiful flowers!