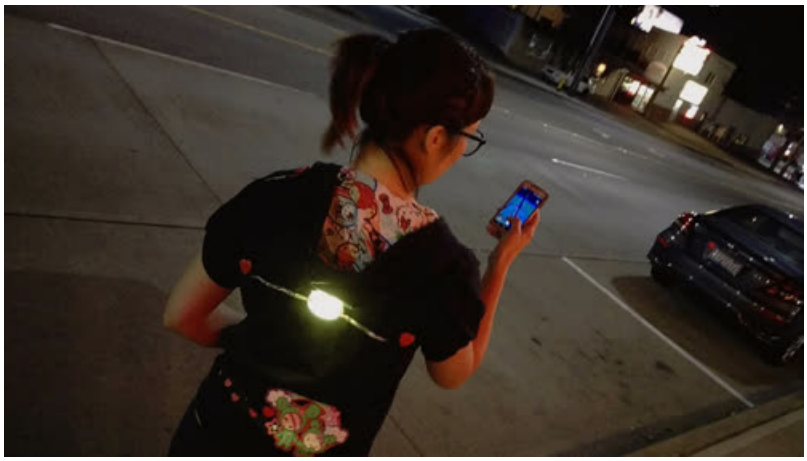
adafruit learning system

Glowing LED Team Badge for Pokemon Go

Created by Richard Albritton

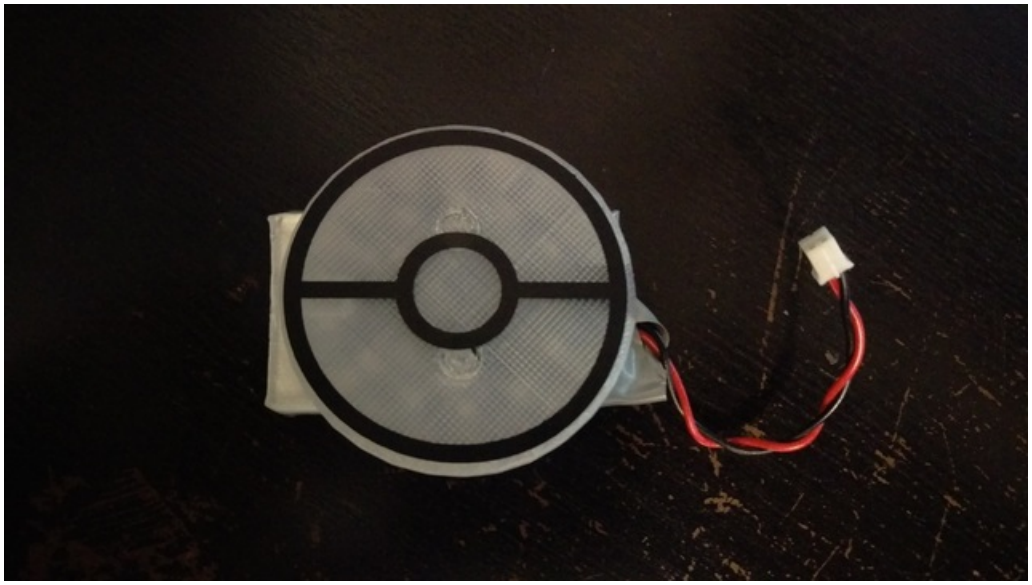


Last updated on 2018-08-22 03:55:08 PM UTC

Guide Contents

Project Overview

Pokemon go has shown us that people can really go crazy over running around our cities bent on catching imaginary animals. It has also shown us that people are going crazy over chasing down said animals at all hours of the night and not always paying attention to what is going on around them. This is amplified when we are talking about children that are having a blast literally trying to catch them all.



We set out to think of a way to increase the visibility of these players while adding a fun new way to add something to the user's game play. What we came up with was a device that let's you display what team you are on and flash noticeably when the player is walking. Adafruit's Circuit Playground provided the perfect platform to start with.

You'll need:

- Circuit Playground (https://adafru.it/ncE)
- Magnetic pin back (https://adafru.it/rd8)
- 3 x AAA battery pack (https://adafru.it/dYF) + AAA batteries (https://adafru.it/e3G)
- OR a Lipoly battery (https://adafru.it/dyW) and a micro lipo charger (https://adafru.it/ewv)
- 3D Printer + filament (or customize your own design!)

Best of all, no soldering is required!

If you have never used the Circuit Playground before, here is a wonderful guide to get you started (https://adafru.it/naC).

Upload the Code

Once the Arduino IDE is set up we will want to upload the following code to the board.

https://adafru.it/p0e

https://adafru.it/p0e

Aside from the color LED animation, pulsing the NeoPixels from light to dark and back, the code is looking for a button press and readings from the Accelerometer.

First we look to see if the Right side button is pressed. If so, we add 1 to the ledState variable. This is then passed through a Switch statement that changes the R,G, and B values for the NeoPixel animation and captures the current ColorState. Since there are only 3 colors, when we get to the 4th switch option, ledState is just set to 0 again so that the colors cycle in a loop for button presses.

```
switch (ledState) {
  case 0:    // Red Team
    R = 255;
    G = 0;
    B = 0;
    ColorState=0;
    break;
  case 1:    // Yellow Team
    R = 255;
    G = 255;
    B = 0;
    ColorState=1;
    break;
  case 2:    // Blue Team
    R = 0;
    G = 0;
    B = 255;
    ColorState=2;
    break;
  case 3:    // Return to Red Team
    ledState = 0;
    break;
```

Next we look for movement using the Accelerometer. There will always be an acceleration reading on the accelerometer because gravity is an acceleration force that will constantly act on the sensor at 9.807 m/s2 in the direction of the ground. So standing still, we should have a net reading from the censor of about 9 to 10. Anything above that would indicate movement and that is what we want to look for. After a bit of testing, we found that a reading of 19 m/s2 was a good indication that someone is walking or running.

The Accelerometer measures acceleration along the X, Y, and Z axis. We need to get a reading on all of the axes at once to look at the net acceleration force exerted on the sensor. We do that by simply adding the X, Y, and Z sensor readings together.

```
Sensor = abs(CircuitPlayground.motionX())+abs(CircuitPlayground.motionY())+abs(CircuitPlayground.motionZ());
```

In theory, no matter what orientation we hold the Circuit Playground in, we will get a net value of ~10 m/s2. There is a bit of variation in the actual reading we get, but it is close.

In conclusion, if the net value of the Accelerometer reading is more that 19 m/s2, we assume that the person is

walking. Then we set the ledState variable to 4. In the switch statement the number 4 option tells the NeoPixels to alternate blinking bright white and a dimer version of the team color.

```
case 4: // Walking detected
ColorFill(255, 255, 255);
delay(300);
ColorFill(R/20, G/20, B/20);
delay(300);
ledState=ColorState;
break;
```

The rest of the code controls the LED animations. Since we want to read the sensors as often as possible, the animations are done a bit more complex than most would be used to. There is a really good article that goes into detail about multitasking NeoPixel animations (https://adafru.it/pcO).

```
#include <Adafruit_CircuitPlayground.h>
// Variables will change:
int ledState = 0;         // 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
int ColorState;   // the previous Color set
int R = 0;
int G = 255;
int B = 0;
int Roffset;
int Goffset;
int Boffset;
int FadeDir = 1;
int FadeTurn =0;
int FadeRate = 50;

// 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
int Sensor;

void setup() {
  CircuitPlayground.begin();
  CircuitPlayground.setBrightness(155);
}

void loop() {
  //CircuitPlayground.lightSensor()
  // read the state of the button into a local variable:
  int reading = CircuitPlayground.rightButton();

  // 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
    if (buttonState == HIGH) {
            ++ledState;
            FadeTurn = 0;
    }
  }
}
Sensor = abs(CircuitPlayground.motionX())+abs(CircuitPlayground.motionY())+abs(CircuitPlayground.motion
if(Sensor>19)ledState=4;

// set the LED:
// do something different depending on the
// range value:
switch (ledState) {
  case 0:    // Red Team
    R = 255;
    G = 0;
    B = 0;
    ColorState=0;
    break;
  case 1:    // Yellow Team
    R = 255;
    G = 255;
    B = 0;
    ColorState=1;
    break;
  case 2:    // Blue Team
    R = 0;
    G = 0;
    B = 255;
    ColorState=2;
    break;
  case 3:    // Return to Red Team
    ledState = 0;
    break;
  case 4:    // Walking detected
    ColorFill(255, 255, 255);
    delay(300);
    ColorFill(R/20, G/20, B/20);
    delay(300);
    ledState=ColorState;
    break;
}
delay(1);        // delay in between reads for stability
Roffset=R/FadeRate;
Goffset=G/FadeRate;
Boffset=B/FadeRate;
if (FadeDir){
  FadeTurn++;
  ColorFill(R-(Roffset*FadeTurn), G-(Goffset*FadeTurn), B-(Boffset*FadeTurn));
  if(FadeTurn>FadeRate){
```

```
      FadeDir = 0;
      //FadeTurn = 0;
    }
  } else{
    FadeTurn--;
    ColorFill(R-(Roffset*FadeTurn), G-(Goffset*FadeTurn), B-(Boffset*FadeTurn));
    if(FadeTurn<1){
      FadeDir = 1;
      //FadeTurn = 0;
    }
  }

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

void ColorFill(uint16_t r,uint16_t g,uint16_t b){
  for(uint16_t i=0; i<10; i++) {
    CircuitPlayground.setPixelColor(i, r, g, b);
  }
}
```

3D Printed Enclosure

The enclosure will protect the Circuit Playground and act as a defuser for the onbourd NeoPixels.

You will want to download the STL file for the print from one of the following links:

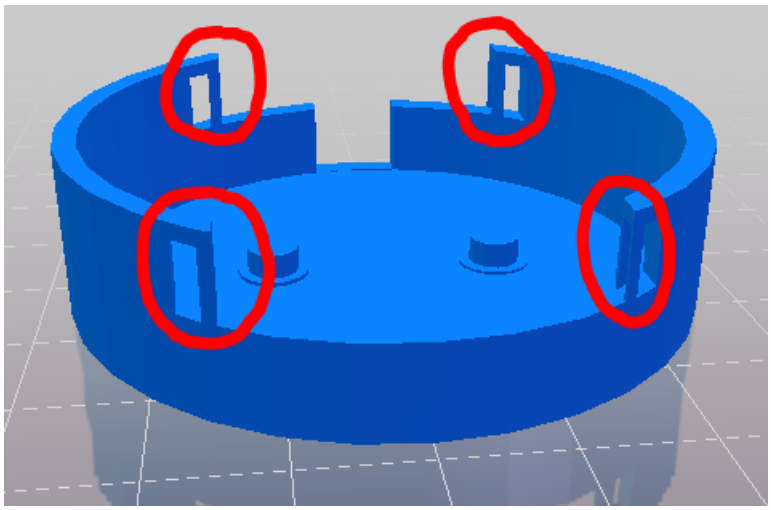
https://adafru.it/p0f

https://adafru.it/p0f

https://adafru.it/p0A

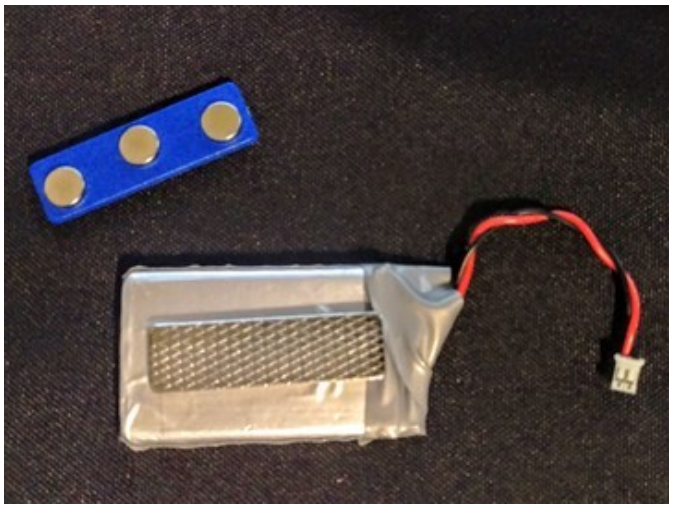
https://adafru.it/p0A

Once the print is done, you will want to remove the support meterial from the battery holder.



Final Assembly

Now we will put all of the poces together.

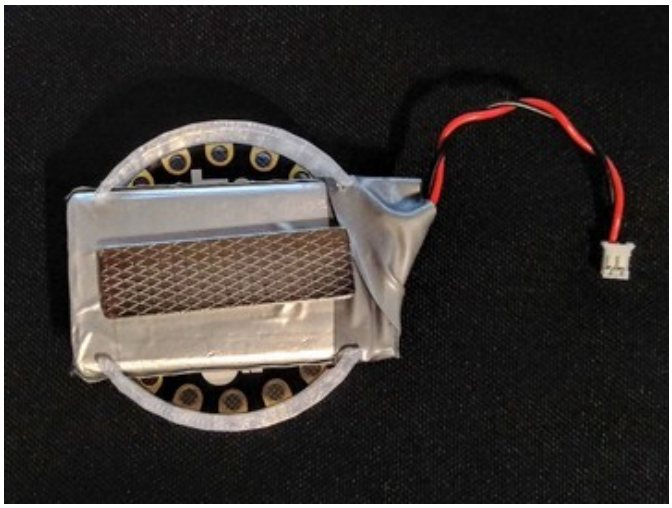


First, attach the Magnetic Pin Back to the Battery.

You can also use a AAA battery pack (http://adafru.it/727), its a bit larger but can use any AAA alkaline or rechargeables

Next place the Circuit Playground into the 3D printed enclosure.

Be sure that the battery connector is lined up with the opening on the side.



Slide the battery into the back bracket so that it holds the Circuit Playground into the enclosure.

Place a bit of electrical tape at one or both ends of the battery to help it stay in better.

When you are ready to use the badge, connect the battery.

All finished!

Using the Badge

Turn it On

Connect the battery to the battery connector on the Circuit Playground.

Attach the Badge

Use the Magnetic Pin Back to attatch the badge to your shirt or jacket.

Changing the color

Pressing the center of the badge will change the color to Red, Yellow, or Blue. You should feel a 'click' when this happens.

Note: The code we uploaded looks for a right side button was press to let you cycle throgh team colors. Pressing the center of the badge just presses both buttons. Since the right side button is the only one used, pressing the left side button as well still changes the color of the badge.

Safty flash

You may notice that moving or tapping the badge makes it flash a bright white light. The badge is set up to detect a certain amount of movement before it starts to flash white. So it will only flash while you are walking or running. Otherwise the light will pulse with your team color.

Recharging the Battery

Unplug the battery and connect it to the Micro Lipo charger.