NeoPixel Basketball Hoop
Created by Justin Cooper

https://learn.adafruit.com/neopixel-mini-basketball-hoop

Last updated on 2023-08-29 02:30:04 PM EDT
# Table of Contents

**Overview**

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

**Parts Needed**

- Power choices!
- Parts for Option #1
- Parts for Option #2
- Tools Needed

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

**Always On Circuit Diagram**

- Step One:
- Step Two:
- Step Three:
- Step Four:
- Step Five:
- Step Six:
- Step Seven:
- Step Eight:
- Step Nine:
- Step Ten:
- Step Eleven:
- Step Twelve:
- Finished!

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

**Point Sensor Circuit Diagram**

- Step One:
- Step Two:
- Step Three:

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

**Arduino Code**

- Installation
- Strand Test Code for NeoPixels
- Point Sensor Code
- Upload

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
</tr>
</tbody>
</table>

**CircuitPython Code**

- Required Libraries
- Install Circuit Python Libraries
- Strand Test Code for NeoPixels
- Point Sensor Code

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
</tr>
</tbody>
</table>

©Adafruit Industries
Basketball hoop lights usually aren't nearly glam enough! They are usually one not-very-bright color. Not anymore! With NeoPixels and Gemma you can program these basketball hoop lights to be a certain color or change colors. Besides, NeoPixels are much brighter than a few LEDs. In this guide I will show you how to install a strip of weatherproof NeoPixels onto a mini basketball hoop. This guide has two code examples: one is to have the NeoPixels shine all the time, the other code example lights the NeoPixels up only when you score a point.
Parts Needed

Gemma M0 (or Gemma v2) ([http://adafru.it/1222](http://adafru.it/1222)) (We recommend using the Gemma M0 as it is easier to program and more compatible with modern computers)

A 1M NeoPixel Strip - We are using 30 LED-per pixel, you can have either white or black backing color. Both work the same ([http://adafru.it/1376](http://adafru.it/1376))

A mini or full sized basketball hoop. Check your local sports shop or hobby store!
Red (http://adafruit.it/288), Yellow (http://adafruit.it/289), and Black (http://adafruit.it/290) Hookup Wire

The IR sensor and IR LED are only needed if you going to have the lights shine when you score a point.

5mm IR LED
You will also need some zip ties & velcro

You can choose how you want to power the project. There's two options - one for beginners and one for more advanced users

**Power choices!**

Beginners may want to go with option #1 - a 3 x AAA battery pack. This is really easy to use, just pop in 3 AAA alkaline or rechargeable batteries. Its less expensive than a LiPo battery, but its also heavier and doesn't have to coolness of built in rechargeability.

Advanced users can go with option #2 - a lithium polymer battery pack. This is a thin
rechargeable battery. It's a little more expensive but its much lighter and thinner. You have to do some more soldering to attach a switch, and an external charger is required.

Parts for Option #1

3 x AAA battery holder with on/off switch and JST cable (http://adafruit.it/727)

3 AAA batteries (http://adafruit.it/617)
Parts for Option #2

SPDT Switch (http://adafruit.it/805)

Lipo Battery (http://adafruit.it/258) (Make sure this is charged! Use the USB charger to charge it up.)

USB Lipo Battery Charger (http://adafruit.it/1304)

Tools Needed

• A Soldering Iron (http://adafruit.it/180)
Always On Circuit Diagram

The instructions on this page are for an "always on" hoop. See the next page "Point Sensor Circuit Diagram" for instructions on adding an IR Sensor and IR LED to have the hoop illuminate when a point is scored.

This diagram uses the Gemma v2 but you can also use the Gemma M0 with the exact same wiring!

Step One:
Cut and strip a black, yellow, and red wire each about 6" long and with 1/2" stripped off the ends
Step Two:
Solder the black wire to the ground pin on the NeoPixels.

Step Three:
Solder the yellow wire to the data pin on the NeoPixels.
Step Four:
Solder the red wire to the 5V pin on the NeoPixels.

Step Five:
Solder the red wire that is connected to the 5V pin on the NeoPixels to the Voltage Out pin on the Gemma. This pin connects to the battery pack so its the best way to light up a bunch of pixels
Step Six:
Solder the yellow wire that connects to the data pin on the NeoPixels to the D1 pin on the Gemma.

Step Seven:
Solder the black wire that connects to the ground pin on the NeoPixel Strip to the ground pad on the Gemma.
Step Eight:
Trim the excess wire on the Gemma on the pads you have just soldered.

Step Nine:
This is for people using the Lipoly battery power. You can skip this step if you’re using 3xAAA batteries.

Cut the positive lead on the Lipo battery in half. Strip a small bit of insulation off both ends. Solder one of the wires to the middle lead on the switch. Solder the other wire to the lead to the left of the center lead.
Step Ten:
This is for people using the Lipoly battery power. You can skip this step if you’re using 3xAAA batteries

Apply two small pieces of heat shrink tubing to the wires we have just soldered. After that is done, connect the JST connector of the battery to the Gemma. Whenever you switch the switch on, it will power the Gemma and NeoPixels.

Step Eleven:
Attach the NeoPixels to the hoop using zip ties. In my case I attached it to the rim.
Step Twelve:
Using Velcro we will secure the Gemma to the battery pack and the battery pack to the mini hoop.

Finished!
Whenever the switch is flipped, it will light up the hoop as soon as the code is uploaded. Now all we have left is to upload the code to the Gemma. Make sure the battery is turned off when uploading code.
So far our hoop will just light up all the time. Using an IR sensor and IR LED we can make the NeoPixels light up whenever we score.

The IR LED pulses out IR light (940nm wavelength) at 38KHz, same as an IR remote. The IR receiver will pick up the signal as long as it can 'see' the LED. When the basketball goes through the hoop, the sensor will say it can't see the LED any more because its blocked - thats how we know it's time to light up the LEDs.
Step One:
Solder a red wire to the 5V pin on the IR sensor. Solder a black wire to the Ground pin on the temperature sensor. Solder a yellow wire to the Data pin on the temperature sensor. Finally cover these connections with heat shrink tubing.

Step Two:
Solder a black wire to the Ground lead on the LED. The ground lead is the shorter one. Next solder a yellow wire to the longer lead on the LED. Put a small bit of heat shrink tubing on both of the connections. Also put some heat shrink tubing over the LED to narrow its beam.
Step Three:

Solder the black Ground wires from both the IR Sensor and IR LED to the Ground pad on the Gemma. Solder the yellow data wire that leads from the LED to D0 on the Gemma. Solder the red 5V pin on the IR sensor to the Vout pin on the Gemma. Finally solder the yellow data cable from the IR sensor to D2 on the Gemma.

Arduino Code

You will need to install the Gemma and NeoPixel libraries. You can install both of these manually, but there is a better way. You can now download the Arduino IDE with support for NeoPixels and Gemma. Visit the Gemma tutorial for tons more information on getting started with Gemma!
Installation

Before you install the software for this project you'll need to make sure you're running the latest Arduino IDE version. In addition you'll need to install the Adafruit NeoPixel library using either the library manager (recommended) or a manual installation.

Once you have the Arduino IDE and NeoPixel library setup, click the button below to download the Arduino sketches for this project from their home on GitHub:

You can download two Arduino sketches:

- NeoPixel_Basketball_Hoop - The hoop is always illuminated.
- NeoPixel_Basketball_Hoop-Point_Sensor - The hoop illuminates when a point is scored.

Open the appropriate sketch in the Arduino IDE.

Strand Test Code for NeoPixels

Once that is installed, you can upload NeoPixel code to the Gemma. The code below is probably what you want to start with. It is based off of the Strand Test code for NeoPixels. This can easily be modified to any animation you want to program.

```
// SPDX-FileCopyrightText: 2017 Mikey Sklar for Adafruit Industries
// SPDX-License-Identifier: MIT

#include <Adafruit_NeoPixel.h>
define PIN 1
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
  rainbow(20);
  rainbowCycle(20);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
```
Point Sensor Code

So far our hoop will just light up all the time. Using an IR sensor and IR LED we can make the NeoPixels light up whenever we score.

The IR LED pulses out IR light (940nm wavelength) at 38KHz, same as an IR remote. The IR receiver will pick up the signal as long as it can 'see' the LED. When the basketball goes through the hoop, the sensor will say it can't see the LED any more because its blocked - thats how we know it's time to light up the LEDs.

#include <Adafruit_NeoPixel.h>
```c
#define PIN 1
#define IR_LED 0 // Digital pin that is hooked up to the IR LED.
#define IR_SENSOR 2
#define BASKET_CHECK_SECONDS 0.1 // How often it checks to see if there is a ball.

Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);

// Setup function
// Called once at start up.
void setup(void){
    // Set up the input and output pins.
    pinMode(IR_LED, OUTPUT);
    pinMode(IR_SENSOR, INPUT);
    strip.begin();
    strip.show(); // Initialize all pixels to 'off'
}

// Loop Function
// Called continuously after setup function.
void loop(void) {
    if (isBallInHoop()) {
        timedRainbowCycle(2000, 10);
    }
    // Delay for 100 milliseconds so the ball in hoop check happens 10 times a second.
    delay(100);
}

// isBallInHoop function
// Returns true if a ball is blocking the sensor.
boolean isBallInHoop() {
    // Pulse the IR LED at 38khz for 1 millisecond
    pulseIR(1000);
    // Check if the IR sensor picked up the pulse (i.e. output wire went to ground).
    if (digitalRead(IR_SENSOR) == LOW) {
        return false; // Sensor can see LED, return false.
    }
    return true; // Sensor can't see LED, return true.
}

// pulseIR function
// Pulses the IR LED at 38khz for the specified number of microseconds.
void pulseIR(long microsecs) {
    // 38khz IR pulse function from Adafruit tutorial: http://learn.adafruit.com/ir-sensor/overview
    // we'll count down from the number of microseconds we are told to wait
    cli(); // this turns off any background interrupts
    while (microsecs > 0) {
        // do nothing
    }
}
```

©Adafruit Industries
// 38 kHz is about 13 microseconds high and 13 microseconds low
digitalWrite(IR_LED, HIGH);  // this takes about 3 microseconds to happen
delayMicroseconds(10);       // hang out for 10 microseconds, you can also
change this to 9 if its not working
digitalWrite(IR_LED, LOW);    // this also takes about 3 microseconds
delayMicroseconds(10);       // hang out for 10 microseconds, you can also
change this to 9 if its not working

// so 26 microseconds altogether
microsecs -= 26;

sei(); // this turns them back on

void timedRainbowCycle(uint32_t milliseconds, uint8_t wait) {
// Get the starting time in milliseconds.
uint32_t start = millis();
// Use a counter to increment the current color position.
uint32_t j = 0;
// Loop until it's time to stop (desired number of milliseconds have elapsed).
while (millis() - start < milliseconds) {
// Change all the light colors.
for (int i = 0; i < strip.numPixels(); ++i) {
    strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
}strip.show();
// Wait the deisred number of milliseconds.
delay(wait);
// Increment counter so next iteration changes.
    j += 1;
} // Turn all the pixels off after the animation is done.
for (int i = 0; i < strip.numPixels(); ++i) {
    strip.setPixelColor(i, 0);
}strip.show();
}

uint32_t Wheel(byte WheelPos) {
    if(WheelPos < 85) {
        return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
    } else if(WheelPos < 170) {
        WheelPos -= 85;
        return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
    } else {
        WheelPos -= 170;
        return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
    }
}

Upload

Once you’ve configured the sketch for your hardware you’re ready to upload it to the Gemma board. Before you upload make sure you’ve setup the Arduino IDE to program Gemma. In the Tools -> Board menu select the Arduino Gemma board (you can use this option even if you’re using the Adafruit Gemma boards). Also in the Tools -> Programmer menu select the USBtinyISP option.
Make sure the Gemma's USB micro/mini connector is connected to your computer. If your Gemma board has an on/off switch slide it into the on position. Press the reset button on the Gemma and you should see its red light start pulsing as the bootloader waits for a sketch to be uploaded. In the Arduino IDE press the upload button or use the Sketch -> Upload command. After a few moments you should see the sketch uploaded to the hardware (on Linux systems you might see broken pipe errors that can be ignored).

If you receive an error be sure to [read the Gemma guide](https://adafruit.com) and confirm you can upload a basic LED blinking sketch.

### 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](https://adafruit.com).

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 “main.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

©Adafruit Industries
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.

**Required Libraries**

Additional libraries will be necessary to run both of the CircuitPython examples on this page.

- neopixel.mpy
- adafruit_irremote.mpy

These libraries are available for download here:

[Adafruit CircuitPython Library Bundle](#)

**Install Circuit Python Libraries**

Now that we have all of the libraries and know which ones this project needs, we'll need to copy them onto the USB drive (which will be named CIRCUITPY after flashing the firmware). In the CIRCUITPY drive, create a new folder and name it "lib". Then, copy the libraries to that "lib" folder. The lib folder should contain neopixel.mpy and a dafruit_irremote.mpy

There are two different CircuitPython sketches to choose from:

1. [NeoPixel_Basketball_Hoop()](#) - Hoop Lights are always on. No point sensor hardware.
2. [NeoPixel_Basketball_Hoop-Point_Sensor()](#) - Uses IR sensor and IR LED to illuminate only when a point is scored.

**Strand Test Code for NeoPixels**

The code below is probably what you want to start with. It is based off of the Strand Test code for NeoPixels. This can easily be modified to any animation you want to program.
import time
import board
from rainbowio import colorwheel
import neopixel

pixpin = board.D1
numpix = 60
strip = neopixel.NeoPixel(pixpin, numpix, brightness=1, auto_write=True)

# Fill the dots one after the other with a color
def colorWipe(color, wait):
    for j in range(len(strip)):
        strip[j] = (color)
    time.sleep(wait)

def rainbow_cycle(wait):
    for j in range(255):
        for i in range(len(strip)):
            idx = int((i * 256 / len(strip)) + j)
            strip[i] = colorwheel(idx & 255)
        time.sleep(wait)

def rainbow(wait):
    for j in range(255):
        for i in range(len(strip)):
            idx = int(i + j)
            strip[i] = colorwheel(idx & 255)
        time.sleep(wait)

while True:
    colorWipe((255, 0, 0), .05)  # red and delay
    colorWipe((0, 255, 0), .05)  # green and delay
    colorWipe((0, 0, 255), .05)  # blue and delay
    rainbow(0.02)
    rainbow_cycle(0.02)

Point Sensor Code

So far our hoop will just light up all the time. Using an IR sensor and IR LED we can make the NeoPixels light up whenever we score.

The IR LED pulses out IR light (940nm wavelength) at 38KHz, same as an IR remote. The IR receiver will pick up the signal as long as it can 'see' the LED. When the basketball goes through the hoop, the sensor will say it can't see the LED any more because its blocked - that's how we know it's time to light up the LEDs.
import time
import board
from rainbowio import colorwheel
import neopixel
import adafruit_irremote
import pwmio
import pulseio

pixpin = board.D1
ir_led = board.D0
ir_sensor = board.D2
basket_check_seconds = 0.1
numpix = 60

# NeoPixel LED
strip = neopixel.NeoPixel(pixpin, numpix, brightness=1, auto_write=False)

# IR LED output for 38kHz PWM
pwm = pwmio.PWMOut(ir_led, frequency=38000)

# IR Sensor input to detect basketball
pulses = pulseio.PulseIn(ir_sensor, maxlen=200, idle_state=True)
dercoder = adafruit_irremote.GenericDecode()

def timed_rainbow_cycle(seconds, wait):
    # Get the starting time in seconds.
    start = time.monotonic()
    # Use a counter to increment the current color position.
    j = 0

    # Loop until it's time to stop (desired number of milliseconds have elapsed).
    while (time.monotonic() - start) < seconds:
        for i in range(len(strip)):
            idx = int((i * 256 / len(strip)) + j)
            strip[i] = colorwheel(idx & 255)
        strip.show()
        # Wait the desired number of milliseconds.
        time.sleep(wait)
        j += 1

    # Turn all the pixels off after the animation is done.
    for i in range(len(strip)):
        strip[i] = (0,0,0)
    strip.show()

def pulse_ir():
    # enable IR LED
    pwm.duty_cycle = (2 ** 16) - 1

def is_ball_in_hoop():
    # Check if the IR sensor picked up the pulse
    pulse = decoder.read_pulses(pulses)
    ir_detect = (len(pulse))

    if ir_detect == 0:
        return False    # Sensor can see LED, return false.
    return True         # Sensor can't see LED, return true.

while True:
    pulse_ir()
    is_ball_in_hoop()
    if is_ball_in_hoop():
        timed_rainbow_cycle(2, 0.01)
    time.sleep(basket_check_seconds)