Festive Feather Holiday Lights

Created by Tony DiCola

https://learn.adafruit.com/festive-feather-holiday-lights

Last updated on 2024-06-03 01:50:43 PM EDT

Table of Contents

Overview 5

Color Sensing Lights 6

- Parts
- Wiring
- Sketch

Bluetooth Low Energy Lights 12

- Parts
- Wiring
- Sketch

Overview

Spice up your holiday decorations using Adafruit's Feather boards (https://adafru.it/keK) to add interactivity in a tiny package. Feathers are little Arduino-compatible microcontrollers that give you all the functionality of a full-size Arduino in a much smaller and battery-friendly package.

You can put a Feather into almost any project, like these festive holiday lights that animate and change color at your command. The 32u4 Feather (http://adafru.it/2771) combined with a color (http://adafru.it/1334) and proximity (http://adafru.it/466) sensor lets you change the color of holiday lights by holding up a brightly colored object to them. The Bluefruit LE Feather (http://adafru.it/2829) lets you change the color of lights using a smartphone or tablet and the Bluefruit LE app. Your holiday decorations will be full of joy with Adafruit's Feathers!

To build these projects you'll want to be familiar with the basics of Arduino. Check out these guides for some background information:

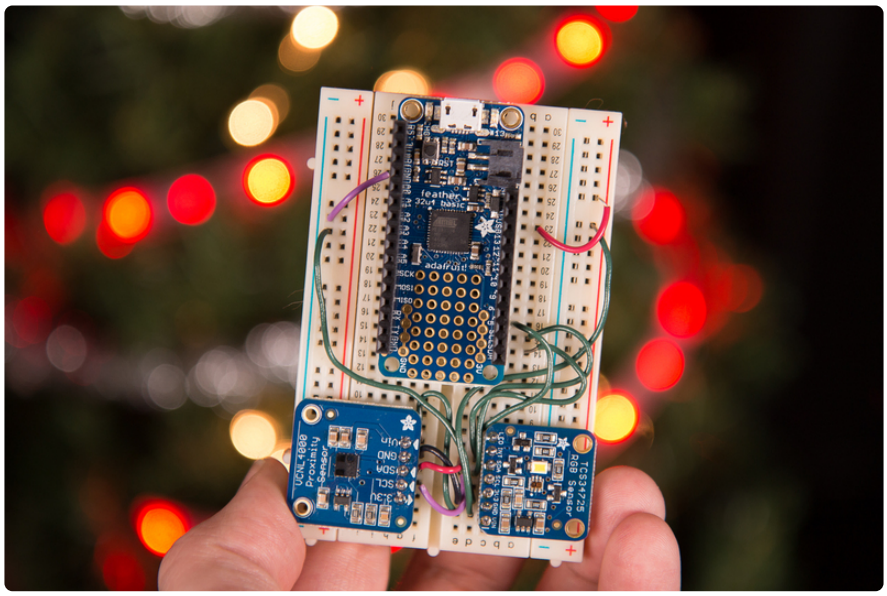
- Learn Arduino Series (https://adafru.it/ja3)
- Adafruit Arduino IDE Setup (https://adafru.it/jAc)

You'll also need to do a little bit of soldering to assemble Feather boards. If you're new to soldering be sure to see the guide to excellent soldering (https://adafru.it/dxy).

Finally for Bluetooth Low Energy it will help to skim these guides for some background information:

- Introduction to Bluetooth Low Energy (https://adafru.it/iCS)
- Bluefruit LE Connect for iOS (https://adafru.it/iCm)

Color Sensing Lights



These holiday lights change their color when you hold an object up to them. A proximity sensor detects when an object is near, and a color sensor detects the object's color so the lights can change to match it--like a chameleon!

Parts

To build these lights you'll need the following parts:

- 32u4 Feather **(http://adafru.it/2771)**
- VCNL4010 Proximity Sensor **(http://adafru.it/466)**
- TCS34725 Color Sensor **(http://adafru.it/1334)**
- NeoPixels **(https://adafru.it/dYn)**
    - You can use a strip, ring, matrix, or even bare pixels. Just make sure your power supply can handle all the pixels you intend to use!
- **5V Power Supply**
    - I recommend the 5 volt 10 amp supply (http://adafru.it/658) as it can handle lighting up to about 150 NeoPixels. Remember each pixel can take up to

60mA alone! Also a DC power jack terminal adapter (http://adafru.it/368) makes it easy to connect a power supply to the circuit.

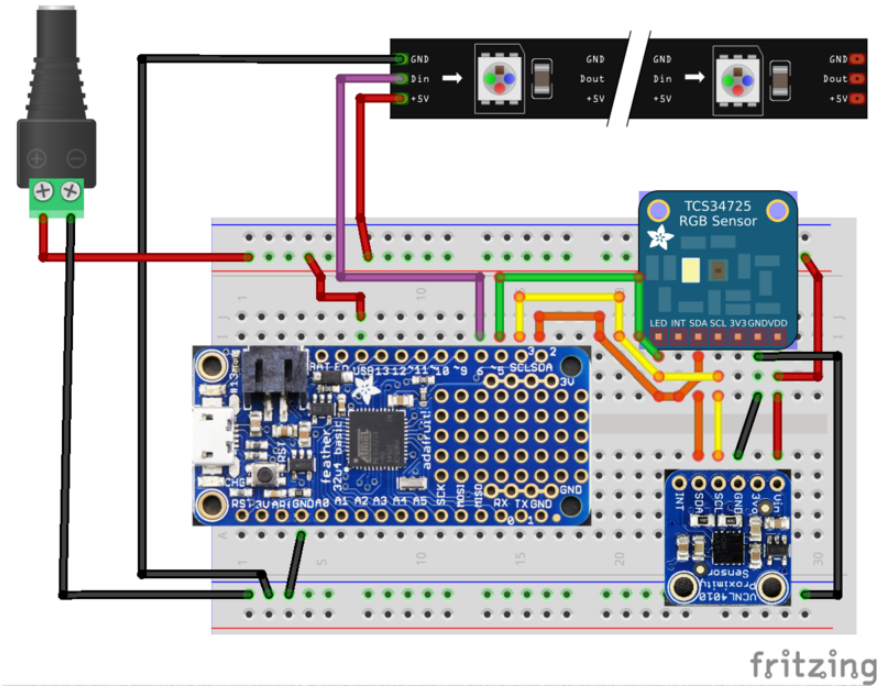
- Breadboard **(http://adafru.it/64)**, jumper wires **(http://adafru.it/153)**, and soldering tools **(http://adafru.it/136)**.

Wiring

First follow the guide for each component to assemble and test it:

- Adafruit 32u4 Feather Guide (https://adafru.it/keQ)
- Adafruit TCS34725 Color Sensor Guide (https://adafru.it/keR)
- Adafruit VCNL4010 Product Page (http://adafru.it/466)

Then wire up the components as shown below:



- Connect **power supply ground** to **32u4 Feather ground**, **TCS34725 ground**, **VCNL4010 ground**, and **NeoPixel ground**.
- Connect **power supply 5V** to **32u4 Feather USB pin**, **TCS34725 VDD pin**, **VCNL4010 VIN pin**, and **NeoPixel +5V**.
- Connect **32u4 Feather SCL** to **TCS34725 SCL**, **VCNL4010 SCL**.
- Connect **32u4 Feather SDA** to **TCS34725 SDA**, **VCNL4010 SDA**.
- Connect **32u4 Feather pin 5** to **TCS34725 LED pin**.
- Connect **32u4 Feather pin 6** to **NeoPIxel signal input (Din)**.

Sketch

Before loading the sketch make sure you have the following libraries installed. Remember you can use the Arduino library manager (https://adafru.it/fCN) to easily search for and install them:

- Adafruit TCS34725 Library (https://adafru.it/cb1)
- Adafruit NeoPixel Library (https://adafru.it/aZU)

Next download the sketch from its home on GitHub (https://adafru.it/EHV) which contains five files under the **Feather_32u4_Lights** folder:

- Adafruit_VCNL4000.cpp (https://adafru.it/EHW)
- Adafruit_VCNL4000.h (https://adafru.it/EHX)
- Adafruit_VCNL4010.cpp (https://adafru.it/EHY)
- Adafruit_VCNL4010.h (https://adafru.it/EHZ)
- Feather_32u4_Lights.ino (https://adafru.it/EH-)

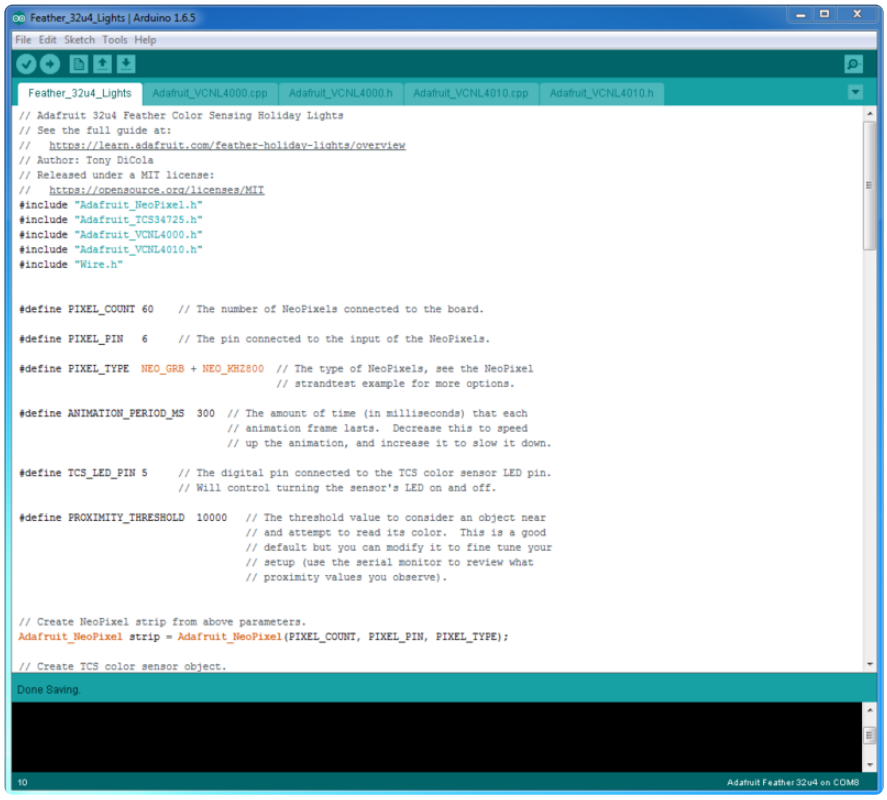
Click **Download: Project Zip** in the main code window below.

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

// Adafruit 32u4 Feather Color Sensing Holiday Lights
// See the full guide at:
//   https://learn.adafruit.com/feather-holiday-lights/overview
// Author: Tony DiCola
// Released under a MIT license:
//   https://opensource.org/licenses/MIT
#include "Adafruit_NeoPixel.h"
#include "Adafruit_TCS34725.h"
#include "Adafruit_VCNL4000.h"
#include "Adafruit_VCNL4010.h"
#include "Wire.h"


#define PIXEL_COUNT 60    // The number of NeoPixels connected to the board.

#define PIXEL_PIN   6     // The pin connected to the input of the NeoPixels.

#define PIXEL_TYPE  NEO_GRB + NEO_KHZ800  // The type of NeoPixels, see the NeoPixel
                                          // strandtest example for more options.

#define ANIMATION_PERIOD_MS  300  // The amount of time (in milliseconds) that each
                                  // animation frame lasts.  Decrease this to speed
                                  // up the animation, and increase it to slow it
down.

#define TCS_LED_PIN 5     // The digital pin connected to the TCS color sensor LED
pin.
                          // Will control turning the sensor's LED on and off.
```

```
#define PROXIMITY_THRESHOLD  10000   // The threshold value to consider an object
near
                                     // and attempt to read its color.  This is a
good
                                     // default but you can modify it to fine tune
your
                                     // setup (use the serial monitor to review what
                                     // proximity values you observe).


// Create NeoPixel strip from above parameters.
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, PIXEL_TYPE);

// Create TCS color sensor object.
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);

// Create VCNL sensor object (defined in this sketch).
// If you're using a VCNL4010 sensor use this line:
Adafruit_VCNL4010 vcnl = Adafruit_VCNL4010();
// However if you're using a VCNL4000 sensor comment the above line and uncomment
this one:
//Adafruit_VCNL4000 vcnl = Adafruit_VCNL4000();

// Build a gamma correction table for better color accuracy.
// Borrowed from TCS library examples.
uint8_t gammatable[256];

// Global variable to hold the current pixel color.  Starts out red but will be
// changed by color sensor.
int r = 255;
int g = 0;
int b = 0;


void setup() {
  Serial.begin(115200);
  Serial.println(F("Adafruit 32u4 Feather Color Sensing Holiday Lights"));

  // Setup TCS sensor LED pin as an output and turn off the LED.
  pinMode(TCS_LED_PIN, OUTPUT);
  digitalWrite(TCS_LED_PIN, LOW);

  // Initialize NeoPixels.
  strip.begin();
  strip.show();

  // Initialize TCS sensor library.
  if (tcs.begin()) {
    Serial.println("Found TCS sensor");
  }
  else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1);
  }

  // Initialize VCNL sensor library.
  if (vcnl.begin()) {
    Serial.println("Found VNCL sensor");
  }
  else {
    Serial.println("No VNCL found ... check your connections");
    while (1);
  }

  // Generate gamma correction table.
  // Taken from TCS color sensor examples.
  for (int i=0; i<256; i++) {
```

```
    float x = i;
    x /= 255;
    x = pow(x, 2.5);
    x *= 255;
    gammatable[i] = x;
  }
}

void loop() {
  // Animate pixels.
  animatePixels(strip, r, g, b, 300);

  // Grab VCNL proximity measurement.
  uint16_t proximity = vcnl.readProximity();
  Serial.print("\t\tProximity = ");
  Serial.println(proximity);

  // Take a TCS color sensor reading if an object is near (proximity measurement is
larger than threshold).
  if (proximity > PROXIMITY_THRESHOLD) {
    // First turn on the LED and wait a bit for a good reading.
    digitalWrite(TCS_LED_PIN, HIGH);
    delay(500);
    // Grab TCS color sensor reading.
    uint16_t raw_r, raw_g, raw_b, raw_c;
    tcs.getRawData(&raw_r, &raw_g, &raw_b, &raw_c);
    // Convert raw values to a value within 0...255, then run it through gamma
correction curve.
    r = gammatable[(int)(((float)raw_r / (float)raw_c)*255.0)];
    g = gammatable[(int)(((float)raw_g / (float)raw_c)*255.0)];
    b = gammatable[(int)(((float)raw_b / (float)raw_c)*255.0)];
    // Print color value.
    Serial.print("R = ");
    Serial.print(r);
    Serial.print(" G = ");
    Serial.print(g);
    Serial.print(" B = ");
    Serial.println(b);
    // Turn off the LED.
    digitalWrite(TCS_LED_PIN, LOW);
    // Pause a bit to prevent constantly reading the color.
    delay(1000);
  }

  // Small delay before looping.
  delay(50);
}

void animatePixels(Adafruit_NeoPixel& strip, uint8_t r, uint8_t g, uint8_t b, int
periodMS) {
  // Animate the NeoPixels with a simple theater chase/marquee animation.
  // Must provide a NeoPixel object, a color, and a period (in milliseconds) that
controls how
  // long an animation frame will last.
  // First determine if it's an odd or even period.
  int mode = millis()/periodMS % 2;
  // Now light all the pixels and set odd and even pixels to different colors.
  // By alternating the odd/even pixel colors they'll appear to march along the
strip.
  for (int i = 0; i < strip.numPixels(); ++i) {
    if (i%2 == mode) {
      strip.setPixelColor(i, r, g, b);  // Full bright color.
    }
    else {
      strip.setPixelColor(i, r/4, g/4, b/4);  // Quarter intensity color.
    }
  }
  strip.show();
}
```

Load the **Feather_32u4_Lights** sketch in the Arduino IDE.



Near the top of the sketch are **#define** values that you can adjust for your light configuration:

- **PIXEL_COUNT** - Set this to the number of NeoPixels you're using.
- **PIXEL_PIN** - Set this to the pin connected to the NeoPixel input (pin 6 if you followed the wiring).
- **PIXEL_TYPE** - Set this to the type of NeoPixel. Use the default unless you know you're using a different type of NeoPixel (see the strandtest example in the NeoPixel library for more information on types of NeoPixels).
- **ANIMATION_PERIOD_MS** - This is the amount of time a single animation frame takes (in milliseconds). Adjust this to slow down and speed up the animation.
- **TCS_LED_PIN** - Set this to the digitial pin connected to the TCS color sensor LED pin (pin 5 if you followed the wiring).
- **PROXIMITY_THRESHOLD** - This is the proximity value that's used to consider an object close enough to read its color with the TCS sensor. You can tune this by looking at the serial monitor output to see the stream of proximity values and adjusting the threshold to be near the values you see when an object is near the sensor.

You can also adjust a part of the code depending on if you're using a VCNL4010 or older VCNL4000 proximity sensor:

```
// Create VCNL sensor object (defined in this sketch).
// If you're using a VCNL4010 sensor use this line:
Adafruit_VCNL4010 vcnl = Adafruit_VCNL4010();
// However if you're using a VCNL4000 sensor comment the above line and uncomment
this one:
//Adafruit_VCNL4000 vcnl = Adafruit_VCNL4000();
```

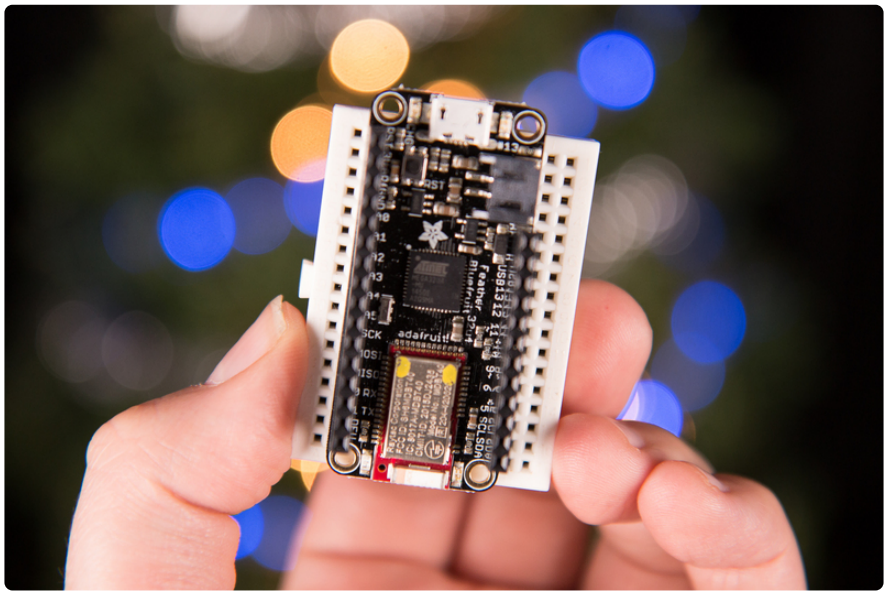
Make sure only one of the lines above is uncommented, depending on what sensor you're using.

Now load the sketch on your hardware, then open the serial monitor at 115200 baud. You should see proximity measurement values printed continuously to the console.

Move a colored object right above the VCNL and TCS color sensor. If the proximity value crosses the threshold the TCS sensor's LED will turn on and read the color of the object. The sensor will wait about a second for you to remove the object, then the lights will start to animate using the color of the object.

Woo hoo, that's all there is to the 32u4 Feather Color Sensing Holiday Lights project! This project is great for coloring lights on the tree using brightly wrapped gifts.

Bluetooth Low Energy Lights



These holiday lights can be controlled right from your smartphone or tablet. Just open the Bluefruit LE app and use its color picker to choose the best color for any celebration.

Parts

To build these lights you'll need the following parts:

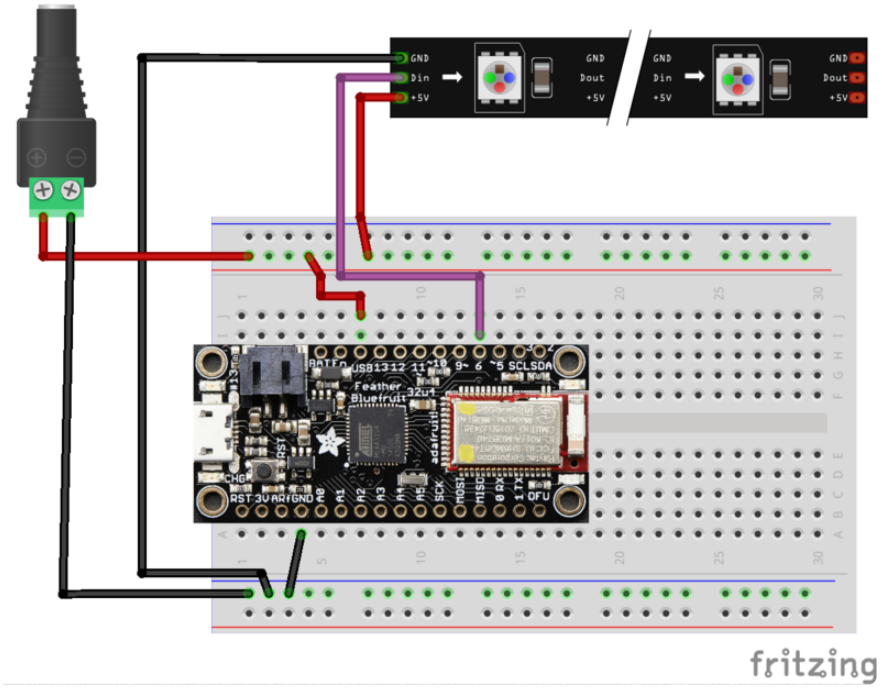
- Bluefruit LE Feather **(http://adafru.it/2829)**
- NeoPixels **(https://adafru.it/dYn)**
    - You can use a strip, ring, matrix, or even bare pixels. Just make sure your power supply can handle all the pixels you intend to use!
- **5V Power Supply**
    - I recommend the 5 volt 10 amp supply (http://adafru.it/658) as it can handle lighting up to about 150 NeoPixels. Remember each pixel can take up to 60mA alone! Also a DC power jack terminal adapter (http://adafru.it/368) makes it easy to connect a power supply to the circuit.
- Breadboard **(http://adafru.it/64)**, jumper wires **(http://adafru.it/153)**, and soldering tools **(http://adafru.it/136)**.

Wiring

First follow the guide for each component to assemble and test it:

- Adafruit Bluefruit LE Feather Guide (https://adafru.it/kcc)

Then wire up the components as shown below:



- Connect **power supply ground** to **Bluefruit LE Feather ground** and **NeoPixel ground**.
- Connect **power supply 5V** to **Bluefruit LE Feather USB pin** and **NeoPixel +5V**.
- Connect **Bluefruit LE Feather pin 6** to **NeoPIxel signal input (Din)**.

Sketch

Before loading the sketch make sure you have the following libraries installed. Remember you can use the Arduino library manager (https://adafru.it/fCN) to easily search for and install them:

- Adafruit Bluefruit nRF51 (https://adafru.it/f4V)
- Adafruit NeoPixel Library (https://adafru.it/aZU)

Next download the sketch from its home on GitHub (https://adafru.it/EHV) which contains **three** files under the **Feather_BluefruitLE_Lights** folder:

- BluefruitConfig.h (https://adafru.it/EI0)
- Feather_BluefruitLE_Lights.ino (https://adafru.it/EI1)
- packerParser.cpp (https://adafru.it/EI2)

Click **Download: Project Zip** in the main code window below.

```
// SPDX-FileCopyrightText: 2019 Tony DiCola for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// Adafruit Bluefruit LE Feather Holiday Lights
// See the full guide at:
//   https://learn.adafruit.com/feather-holiday-lights/overview
// Author: Tony DiCola
// Based on the neopixel_picker example from the Adafruit Bluefruit nRF51 library.
// Released under a MIT license:
//   https://opensource.org/licenses/MIT
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_NeoPixel.h"
#include "BluefruitConfig.h"
#include "SoftwareSerial.h"
#include "SPI.h"


#define PIXEL_COUNT 60    // The number of NeoPixels connected to the board.

#define PIXEL_PIN   6     // The pin connected to the input of the NeoPixels.

#define PIXEL_TYPE  NEO_GRB + NEO_KHZ800  // The type of NeoPixels, see the NeoPixel
                                          // strandtest example for more options.

#define ANIMATION_PERIOD_MS  300  // The amount of time (in milliseconds) that each
                                  // animation frame lasts.  Decrease this to speed
                                  // up the animation, and increase it to slow it
down.


// Create NeoPixel strip from above parameters.
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, PIXEL_TYPE);

// Create the Bluefruit object using hardware SPI (for Bluefruit LE feather).
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ,
BLUEFRUIT_SPI_RST);

// Global variable to hold the current pixel color.  Starts out red but will be
// changed by the BLE color picker.
int r = 255;
int g = 0;
int b = 0;

// Function prototypes and data over in packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);
extern uint8_t packetbuffer[];


void setup(void)
{
  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit LE Holiday Lights"));

  // Initialize NeoPixels.
  strip.begin();
  strip.show();

  // Initialize BLE library.
  if (!ble.begin(false))
  {
    Serial.println(F("Couldn't find Bluefruit LE module!"));
    while (1);
  }
  ble.echo(false);
```

```
  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in Controller
mode"));
  Serial.println(F("Then activate/use the color picker to change color."));
  Serial.println();

  // Wait for connection.
  while (!ble.isConnected()) {
    // Make sure to animate the pixels while waiting!
    animatePixels(strip, r, g, b, ANIMATION_PERIOD_MS);
    delay(50);
  }
  ble.setMode(BLUEFRUIT_MODE_DATA);
}

void loop(void)
{
  // Animate the pixels.
  animatePixels(strip, r, g, b, ANIMATION_PERIOD_MS);

  // Grab a BLE controller packet if available.
  uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
  if (len == 0) return;

  // Parse a color packet.
  if (packetbuffer[1] == 'C') {
    // Grab the RGB values from the packet and change the light color.
    r = packetbuffer[2];
    g = packetbuffer[3];
    b = packetbuffer[4];
    // Print out the color that was received too:
    Serial.print ("RGB #");
    if (r < 0x10) Serial.print("0");
    Serial.print(r, HEX);
    if (g < 0x10) Serial.print("0");
    Serial.print(g, HEX);
    if (b < 0x10) Serial.print("0");
    Serial.println(b, HEX);
  }
}

void animatePixels(Adafruit_NeoPixel& strip, uint8_t r, uint8_t g, uint8_t b, int
periodMS) {
  // Animate the NeoPixels with a simple theater chase/marquee animation.
  // Must provide a NeoPixel object, a color, and a period (in milliseconds) that
controls how
  // long an animation frame will last.
  // First determine if it's an odd or even period.
  int mode = millis()/periodMS % 2;
  // Now light all the pixels and set odd and even pixels to different colors.
  // By alternating the odd/even pixel colors they'll appear to march along the
strip.
  for (int i = 0; i < strip.numPixels(); ++i) {
    if (i%2 == mode) {
      strip.setPixelColor(i, r, g, b);  // Full bright color.
    }
    else {
      strip.setPixelColor(i, r/4, g/4, b/4);  // Quarter intensity color.
    }
  }
  strip.show();
}
```

Unzip the archive and load the **Feather_BluefruitLE_Lights** sketch in the Arduino IDE.

Near the top of the sketch are **#define** values that you can adjust to suite your light configuration:

- **PIXEL_COUNT** - Set this to the number of NeoPixels you're using.
- **PIXEL_PIN** - Set this to the pin connected to the NeoPixel input (pin 6 if you followed the wiring).
- **PIXEL_TYPE** - Set this to the type of NeoPixel. Use the default unless you know you're using a different type of NeoPixel (see the strandtest example in the NeoPixel library for more information on types of NeoPixels).
- **ANIMATION_PERIOD_MS** - This is the amount of time a single animation frame takes (in milliseconds). Adjust this to slow down and speed up the animation.

Upload the sketch to your hardware and you should see the lights turn on and animate with a red color. Now use the Bluefruit LE app on an iOS (https://adafru.it/iCm) or Android (https://adafru.it/f4G) device and connect to the Bluefruit Feather in **Controller** mode. Use the **Color Picker** to pick a color and **send** it to the Bluefruit--woo hoo, the lights will change color!

Check out the Bluetooth NeoPixel Goggles project (https://adafru.it/keU) for a little more explanation about using the color picker.

That's all there is to the Bluefruit LE Feather holiday light project! Use the color picker on your phone/table to control your holiday lights with ease!