adafruit learning system

Random Spooky LED Eyes

Created by Bill Earl

Last updated on 2018-08-22 03:38:00 PM UTC

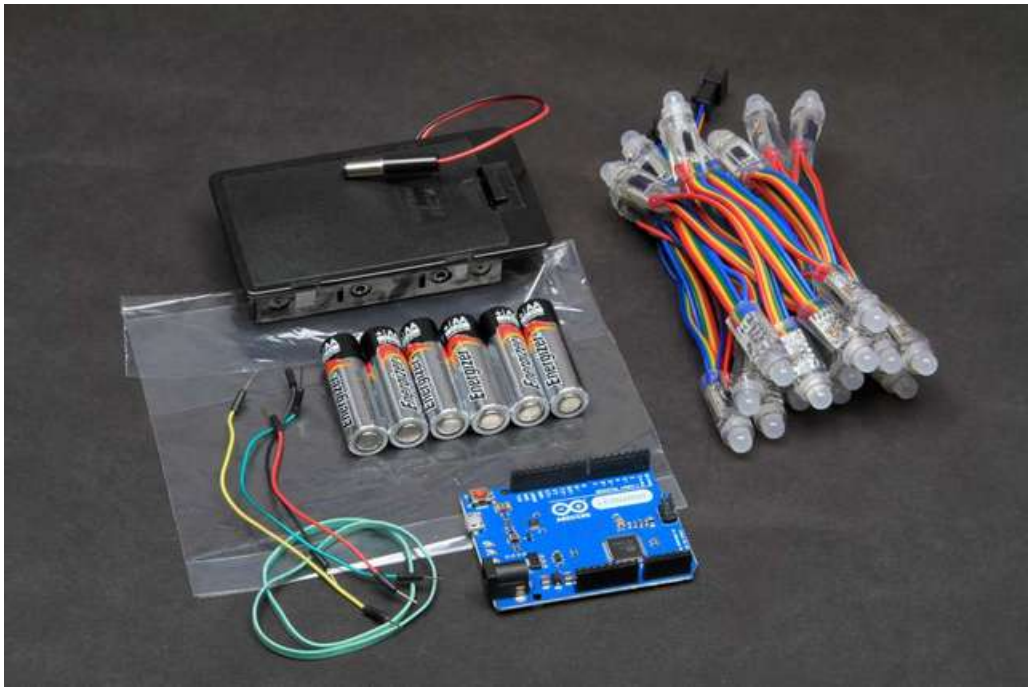
Guide Contents

Overview and Materials

Overview:

This is an easy project that requires no soldering and only takes a few minutes to put together. WIth a strand of digital led pixels and an Arduino, you can create 12 pairs of creepy eyes, randomly blinking from behind a dark window or in the bushes.

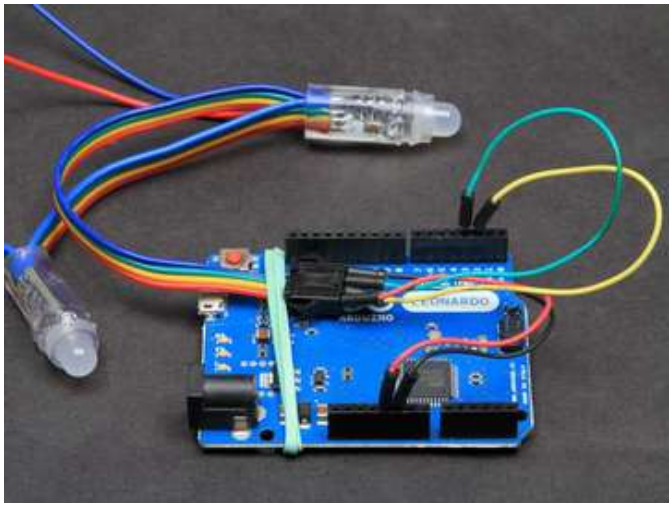
Materials:

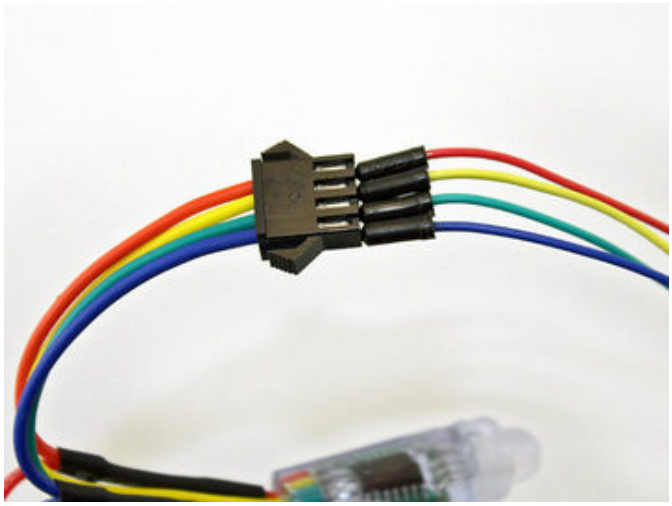


- 1 Arduino (http://adafru.it/849)- Most any type will work (Uno, Leonardo, Mega, Due, Flora etc). We used a Leonardo for the protype.
- 1 12mm Pixel Strand (http://adafru.it/322)
- 4 Jumper wires (http://adafru.it/153)
- 1 6xAA battery box (http://adafru.it/248)
- 6 AA batteries
- 1 small plastic bag (for outdoor use)
- Rubber bands
- Tape (for window mounting)
- Spring clamps, clothespins or twist-ties (for attaching to shrubs)

Assembly

Assembly requires just 4 connections to the pixel strip. These are easily done with breadboard jumper wires, although you could use short pieces of solid core wire as well:



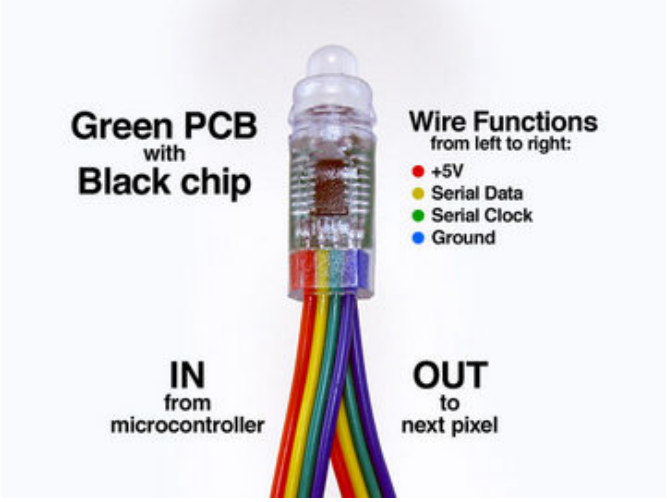


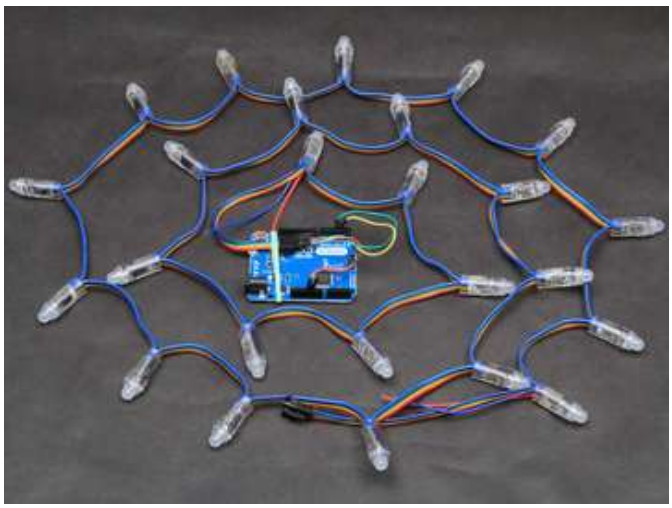
Connect the Pixels:

Connect the Arduino to the pixel strip as shown: Plug one end of the jumper into the Arduino header and the other end into the female connector on the input end of the strip. Tape or rubber-band the connector to the board so the wires don't pull out.

- **+5v - Red wire**
- **GND - Blue (black) wire**
- **Pin 2 (Data) - Yellow wire**
- **Pin 3 (Clock) - Green Wire**

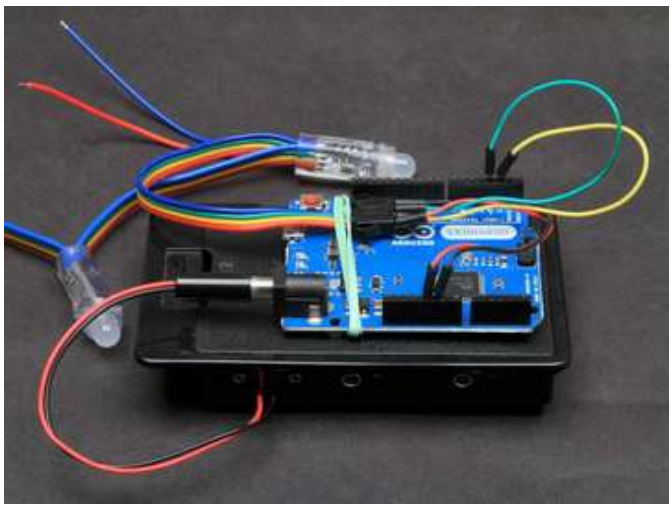
If your pixels do not look like the ones in the photos, check this page for details on wiring other styles of pixels. http://learn.adafruit.com/12mm-led-pixels/wiring (https://adafru.it/cO4)





Load and Test:

At this point, you should load the code (see the next page) and test to make sure everything is connected properly. If all is well, you should see random pairs of leds 'blinking'.



Attach the Battery Pack:

Place the Arduino on top of the battery pack and tape or rubber-band in place.

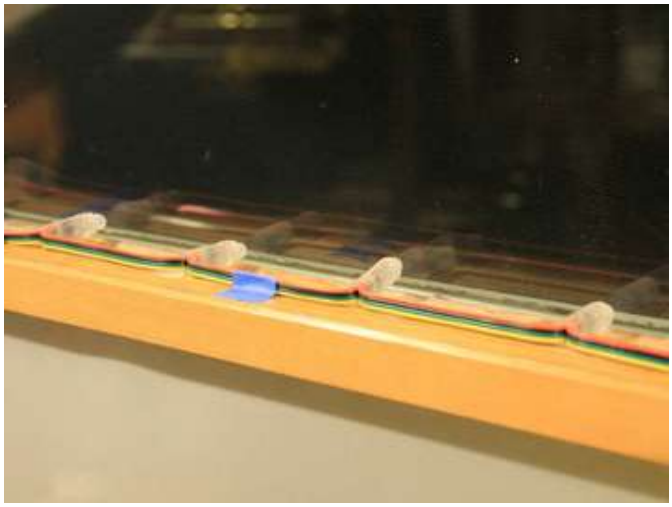
Is it safe to run a pixel strand from the Arduino 5v pin?

Normally, no. But the code for this project only lights up a few pixels (no more than 6) at a time, and never more than 1/3 intensity. If you modify this code to use more pixels or higher power levels, you should consider a separate 5v supply for the strand. http://learn.adafruit.com/battery-power-for-led-pixels-and-strips

Can I add another strand for more 'eyes'?

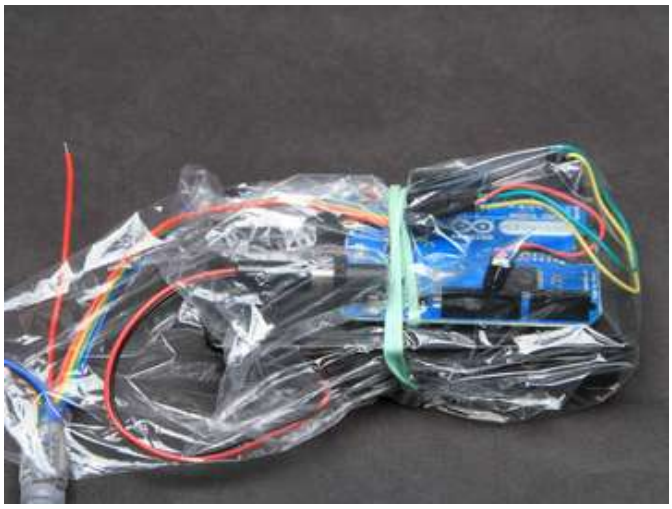
Yes. You can chain the strands together. You will also need to increase "numPixels" in the code. However, if you decide to increase "maxEyes", you risk overloading the Arduino voltage regulator and should use a separate 5v power supply for the pixels. See this tutorial for details: http://learn.adafruit.com/battery-power-for-led-pixels-and-strips

Window Mounting:



For window mounting, simply tape the LED strand to the window sill and/or muntins. Be sure to keep the room lights off for best effect!

Outside Mounting:



Bag it:

For outside use, wrap the whole assembly in a plastic bag and secure with rubber bands. Suspend the assembly with the open end of the bag facing down so the bag does not fill with water if it rains.

Find some dense shrubbery and attach the led pixels to the branches with clothspins, spring clips or twist ties. Try to keep them arranged in roughly horizontal pairs.

Start counting pairs from the input end of the strand. Since there are 25 pixels per strand, you will have one odd pixel left over at the end. Just tuck that one back behind some leaves.

The Code

Libraries:

To run this code you will first need to install the Adafruit WS2801 Pixel library:

https://adafru.it/cO6

https://adafru.it/cO6

See this guide for instructions on installing the library: Arduino Libraries | All About Arduino Libraries | Adafruit Learning System (https://adafru.it/aYM)

About the code:

The code below will run on most models of Arduino. It has been tested on the Uno, Mega and Leonardo. To get up and running quickly, just compile and upload the code. But when you get a chance, take a closer look to see how it is put together.

This code makes use of a couple of very useful programming techniques:

- **Object Oriented Programming -** Each pair of eyes is modeled using the "blinker" class to simplify management of multiple independent pairs. To add another pair of eyes, simply change maxEyes.
- **State Machines -** The "blinker" class is implemented as a state machine. It does not use delays for timing. You just need to call the 'step()' member on a regular basis to update its state. This means that the phase and timing of one pair of eyes is completely independent of the other pairs, and your code can do other useful work in-between calls to 'step()'.

```
/*************************************************************************
Random Eyes sketch for WS2801 pixels
W. Earl 10/16/11
For Adafruit Industries

Creates randomized pairs of WS2801 led pixels that look like eyes peering
from the darkness.

Blinking is implemented as an array of state machines so that multiple pairs
of eyes can be active concurrently, but in different phases of a blink.
*************************************************************************/
#include "SPI.h"
#include "Adafruit_WS2801.h"


int dataPin = 2;
int clockPin = 3;

const int numPixels = 24;  // Change this if using more than one strand

const int maxEyes = 3; // maximum number of concurrently active blinkers

// dead-time between lighting of a range of pixels
const int deadTimeMin = 50;
const int deadTimeMax = 500;

// interval between blink starts - independent of position
const int intervalMin = 10;
const int intervalMax = 300;
```

```
const int intervalMax = 500;

const int stepInterval = 10;
long lastStep = 0;

Adafruit_WS2801 strip = Adafruit_WS2801(numPixels, dataPin, clockPin);

/*****************************************************************************
Blinker Class

Implements a state machine which generates a blink of random duration and color.
The blink uses two adjacent pixels and ramps the intensity up, then down, with
a random repeat now and again.
*****************************************************************************/

class blinker
{
  public:

  boolean m_active;  // blinker is in use.
  int m_deadTime;  // don't re-use this pair immediately

  int m_pos;  // position of the 'left' eye.  the 'right' eye is m_pos + 1

  int m_red;  // RGB components of the color
  int m_green;
  int m_blue;

  int m_increment;  // ramp increment - determines blink speed
  int m_repeats;  // not used
  int m_intensity;  // current ramp intensity

  public:
  // Constructor - start as inactive
  blinker()
  {
    m_active = false;
  }


  // 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;
  }

  // Initiate a blink at the specified pixel position
  // All other blink parameters are randomly generated
  void StartBlink(int pos)
  {
    m_pos = pos;

    // Pick a random color - skew toward red/orange/yellow part of the spectrum for extra creepyness
    m_red = random(150, 255);
```

```
  m_blue = 0;
  m_green = random(100);

  m_repeats += random(1, 3);

  // set blink speed and deadtime between blinks
  m_increment = random(1, 6);
  m_deadTime = random(deadTimeMin, deadTimeMax);

  // Mark as active and start at intensity zero
  m_active = true;
  m_intensity = 0;
}

// Step the state machine:
void step()
{
  if (!m_active)
  {
    // count down the dead-time when the blink is done
    if (m_deadTime > 0)
    {
      m_deadTime--;
    }
    return;
  }

  // Increment the intensity
  m_intensity += m_increment;
  if (m_intensity >= 75)  // max out at 75 - then start counting down
  {
    m_increment = -m_increment;
    m_intensity += m_increment;
  }
  if (m_intensity <= 0)
  {
      // make sure pixels all are off
    strip.setPixelColor(m_pos, Color(0,0,0));
    strip.setPixelColor(m_pos+1, Color(0,0,0));

    if (--m_repeats <= 0)      // Are we done?
    {
       m_active = false;
    }
    else // no - start to ramp up again
    {
        m_increment = random(1, 5);
    }
    return;
  }

  // Generate the color at the current intensity level
  int r =  map(m_red, 0, 255, 0, m_intensity);
  int g =  map(m_green, 0, 255, 0, m_intensity);
  int b =  map(m_blue, 0, 255, 0, m_intensity);
  uint32_t color = Color(r, g, b);

  // Write to both 'eyes'
  strip.setPixelColor(m_pos, color);
  strip.setPixelColor(m_pos+1, color);
```

```
    strip.setPixelColor(m_pos +1, color);
  }
};

// An array of blinkers - this is the maximum number of concurrently active blinks
blinker blinkers[maxEyes];

// A delay between starting new blinks
int countdown;

void setup()
{
  Serial.begin(9600);
  // initialize the strip
  strip.begin();
  strip.show();

  countdown = 0;
}


void loop()
{
  if (millis() - lastStep > stepInterval)
  {
    lastStep = millis();
    --countdown;
    for(int i = 0; i < maxEyes; i++)
    {
      // Only start a blink if the countdown is expired and there is an available blinker
      if ((countdown <= 0) && (blinkers[i].m_active == false))
      {
        int newPos = random(0, numPixels/2) * 2;

        for(int j = 0; j < maxEyes; j++)
        {
          // avoid active or recently active pixels
          if ((blinkers[j].m_deadTime > 0) && (abs(newPos - blinkers[j].m_pos) < 4))
          {
            Serial.print("-");
            Serial.print(newPos);
            newPos = -1;  // collision - do not start
            break;
          }
        }

        if (newPos >= 0)  // if we have a valid pixel to start with...
        {
         Serial.print(i);
         Serial.print(" Activate - ");
         Serial.println(newPos);
         blinkers[i].StartBlink(newPos);
         countdown = random(intervalMin, intervalMax);  // random delay to next start
        }
      }
      // step all the state machines
       blinkers[i].step();
    }
    // update the strip
    strip.show();
```

```
  }
}
```