Neopixel Cosmic Turtle Necklace

Created by Erin St Blaine

https://learn.adafruit.com/neopixel-led-magnetic- pendant-necklace

Last updated on 2023-08-29 02:43:26 PM EDT
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Materials</td>
<td>3</td>
</tr>
<tr>
<td>Wiring</td>
<td>4</td>
</tr>
<tr>
<td>The Code</td>
<td>5</td>
</tr>
<tr>
<td>• If you encounter trouble...</td>
<td></td>
</tr>
<tr>
<td>Necklace Build</td>
<td>10</td>
</tr>
<tr>
<td>LED Build</td>
<td>11</td>
</tr>
<tr>
<td>Clasp &amp; Finishing</td>
<td>15</td>
</tr>
</tbody>
</table>
Overview

Cosmic Turtle wakes
Spins the tides with pulsing light
Ocean in your heart

Turn him on by putting him on — the magnetic necklace clasp doubles as an on/off switch. Recharge him by plugging him handily into your computer with a USB cable. And when his battery dies, he glows in the dark!

Wear him proudly around your neck or give him to someone shiny.

This project is easy to customize! If you already have a Cosmic Turtle, or if for some unfathomable reason you don’t like Turtles, you can still create a blinking, twinkling, pulsing, slick LED necklace that embraces your inner Spirit Animal.

Materials

Important Bits:

- Pro Trinket 3V ()
- Pro Trinket Charging Backpack ()
- Neopixel Jewel (http://adafruit.it/2226)
• **150mAh LiPo battery** ([http://adafru.it/1317](http://adafru.it/1317))

• Hollow necklace pendant (at least 1.5” x 2”)

• Magnetic necklace clasp ([I like this one](http://adafru.it/1317))

**Tools & Other Stuff**

• Soldering iron, solder & accessories

• Light-colored polymer clay ([I used glow-in-the-dark Sculpey](http://adafru.it/1311))

• Black 26AWG Silicone stranded wire

• **Solid core wire in various colors** ([http://adafru.it/1311](http://adafru.it/1311)) (you just need a tiny bit but it's good to have on hand)

• Electrical tape

• Hot glue gun

• Oven

**Wiring**

Using solid core wire:

• Neopixel PWR -> Trinket JST + pad (on the back)
• Neopixel GND -> Trinket GND
• Neopixel IN -> Trinket 9

NOTE: do not use the 3V pin on the far end of the Pro Trinket (the one next to your GND connection). This pin is for input from an FTDI cable, not output, and your lights won’t come on.

Using Backpack Charger’s included header pins:

• Backpack BAT -> Trinket BAT
• Backpack G -> Trinket G
• Backpack 5v -> Trinket BUS

Using 26g silicone stranded wire:

• Two Backpack switch pins to magnetic clasp

The Code

Software Setup

If this is your first time using Pro Trinket, take a look at Introducing Pro Trinket () to get a guided tour.

Once you’ve got your Pro Trinket up and running with Arduino (), you’ll need to install the FastLED library.

FastLED Library

You will also need to install the FastLED library in Arduino (Sketch > Include Library > Manage Libraries...)
Libraries? Why? What's a Library?

In a nutshell, Arduino libraries have a lot of pre-written functions that make your neopixels easy to command. You can do fancy stuff without being a code guru. Yay Libraries!

FastLED is a fast, efficient, easy-to-use Arduino library for programming addressable LED strips and pixels. It has a lot of features to get your animations up and running fast -- and it has a lot of code samples available if you’re just learning to code.

All about Arduino Libraries () will tell you everything you ever wanted to know about libraries, including more detailed installation instructions.

Once your curiosity is satiated and your library is installed, copy and paste the code into your Arduino window.

Go to your Tools menu and select "Pro Trinket 3V USB" from the list of boards. Plug your Pro Trinket into your computer via the onboard USB port. Press the "reset" button on your Pro Trinket and wait for the blinky red light, then click the upload button in Arduino.

This Twinkling Lights code was written by Mark Kriegsman. It slowly twinkles and fades your Neopixels on and off and gives you a variety of color palettes to choose from.

```cpp
#include "FastLED.h"

#define LED_PIN     9
#define LED_TYPE    WS2812B
#define COLOR_ORDER GRB
#define NUM_LEDS    5

CRGB leds[NUM_LEDS];
CRGBPalette16 currentPalette;
TBlendType    currentBlending;

//  Twinkling 'holiday' lights that fade up and down in brightness.
//  Colors are chosen from a palette; a few palettes are provided.
//  The basic operation is that all pixels stay black until they
//  are 'seeded' with a relatively dim color. The dim colors
//  are repeatedly brightened until they reach full brightness, then
//  are darkened repeatedly until they are fully black again.
//  A set of 'directionFlags' is used to track whether a given
//  pixel is presently brightening up or darkening down.
//  For illustration purposes, two implementations of directionFlags
//  are provided: a simple one-byte-per-pixel flag, and a more
//  complicated, more compact one-BIT-per-pixel flag.
//  Darkening colors accurately is relatively easy: scale down the
```
// existing color channel values. Brightening colors is a bit more
// error prone, as there's some loss of precision. If your colors
// aren't coming our 'right' at full brightness, try increasing the
// STARTING_BRIGHTNESS value.

// -Mark Kriegsman, December 2014

#define MASTER_BRIGHTNESS   18   // change this to change overall brightness
#define STARTING_BRIGHTNESS 128  // change this to change maximum brightness
#define FADE_IN_SPEED       5    // lower number fades in slower
#define FADE_OUT_SPEED      10   // higher number hangs around longer
#define DENSITY             255

void setup() {
  delay(3000);
  FastLED.addLeds<LED_TYPE,LED_PIN,COLOR_ORDER>(leds,
      NUM_LEDS).setCorrection(TypicalLEDStrip);
  FastLED.setBrightness(MASTER_BRIGHTNESS);
  currentBlending = LINEARBLEND;
}

CRGBPalette16 gPalette;

void loop() {
  //Un-comment the line with the colors you like best
gPalette = OceanColors_p;
  //gPalette = RainbowColors_p;
  //gPalette = HeatColors_p;
  //gPalette = PartyColors_p;
  //gPalette = CloudColors_p;
  //gPalette = RainbowStripeColors_p;
colortwinkles();
  FastLED.show();
  FastLED.delay(70);       // change this to change overall speed
}

enum { GETTING_DARKER = 0, GETTING_BRIGHTER = 1 };

void colortwinkles() {
  // Make each pixel brighter or darker, depending on
  // its 'direction' flag.
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);

  // Now consider adding a new random twinkle
  if( random8() &lt; DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( gPalette, random8(), STARTING_BRIGHTNESS,
          NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void brightenOrDarkenEachPixel( fract8 fadeUpAmount, fract8 fadeDownAmount) {
  for( uint16_t i = 0; i &lt; NUM_LEDS; i++) {
    if( getPixelDirection(i) == GETTING_DARKER) {
      // This pixel is getting darker
      leds[i] = makeDarker( leds[i], fadeDownAmount);
    } else {
      // This pixel is getting brighter
      leds[i] = makeBrighter( leds[i], fadeUpAmount);
      // now check to see if we've maxxed out the brightness
      if( leds[i].r == 255 || leds[i].g == 255 || leds[i].b == 255) {
        // if so, turn around and start getting darker
      }
    }
  }
}
```cpp
setPixelDirection(i, GETTING_DARKER);
}
}

CRGB makeBrighter( const CRGB& color, fract8 howMuchBrighter)
{
    CRGB incrementalColor = color;
    incrementalColor.nscale8( howMuchBrighter);
    return color + incrementalColor;
}

CRGB makeDarker( const CRGB& color, fract8 howMuchDarker)
{
    CRGB newcolor = color;
    newcolor.nscale8( 255 - howMuchDarker);
    return newcolor;
}

// For illustration purposes, there are two separate implementations
// provided here for the array of `directionFlags`:
// - a simple one, which uses one byte (8 bits) of RAM for each pixel, and
// - a compact one, which uses just one BIT of RAM for each pixel.

// Set this to 1 or 8 to select which implementation
// of directionFlags is used.  1=more compact, 8=simpler.
#define BITS_PER_DIRECTION_FLAG 1

#if BITS_PER_DIRECTION_FLAG == 8
// Simple implementation of the directionFlags array,
// which takes up one byte (eight bits) per pixel.
uint8_t directionFlags[NUM_LEDS];

bool getPixelDirection( uint16_t i) {
    return directionFlags[i];
}

void setPixelDirection( uint16_t i, bool dir) {
    directionFlags[i] = dir;
}
#endif

#if BITS_PER_DIRECTION_FLAG == 1
// Compact (but more complicated) implementation of
// the directionFlags array, using just one BIT of RAM
// per pixel.  This requires a bunch of bit wrangling,
// but conserves precious RAM.  The cost is a few
// cycles and about 100 bytes of flash program memory.
uint8_t  directionFlags[ (NUM_LEDS+7) / 8];

bool getPixelDirection( uint16_t i) {
    uint16_t index = i / 8;
    uint8_t  bitNum = i &amp; 0x07;
    // using Arduino 'bitRead' function; expanded code below
    return bitRead( directionFlags[index], bitNum);
    // uint8_t  andMask = 1 << bitNum;
    // return (directionFlags[index] &amp; andMask) != 0;
}

void setPixelDirection( uint16_t i, bool dir) {
    uint16_t index = i / 8;
    uint8_t  bitNum = i &amp; 0x07;
    // using Arduino 'bitWrite' function; expanded code below
    bitWrite( directionFlags[index], bitNum, dir);
    //  uint8_t  orMask = 1 << bitNum;
```

© Adafruit Industries
There are a number of things you can change in this code to get exactly the effect you're looking for with your necklace.

- Brightness: play with changing the MASTER_BRIGHTNESS and STARTING_BRIGHTNESS numbers
- Speed: play with the FADE_IN_SPEED and FADE_OUT_SPEED numbers
- Color Palette: Un-comment the palette featuring the colors you want, or check out the FastLED site for instructions on how to create your own palette

I used a moderate brightness so as not to blind people with Cosmic Turtle's shiny shell, and used an ocean palette since, well, Turtle.

If you encounter trouble...

Any time you hit a roadblock with a neopixel project, we'll usually ask that you start with the "strandtest" example from our own Adafruit_NeoPixel library. This helps us narrow down whether it's a hardware or software issue. The library is installed similarly to FastLED or any other in Arduino (Sketch > Include Library > Manage Libraries...)

You'll find the strandtest example under File→Sketchbook→Libraries→Adafruit_NeoPixel→strandtest

If strandtest fails to run, this suggests a hardware issue...for example, connecting to the wrong Gemma pin.

If you’re new to Arduino programming and LEDs, we usually suggest starting with the Adafruit_NeoPixel library...it’s pretty basic, the strip declaration is more conventional, and we can stay on top of keeping it compatible with our own products and the most mainstream Arduino boards.

As FastLED is a more "bleeding edge" third-party library, we can’t always guarantee compatibility across versions or with specific boards. You can find help through their community on Google+. This is potent stuff, written by people with a deep appreciation for LED art, and we wanted to showcase it.
Necklace Build

Selecting Your Pendant

Find a pendant that speaks to you. You want one that's around 1.5 x 2 inches in size, with holes in it so the lights can get through. A bigger pendant is better! If it's too small, the electronics will show around the edges.

It's best to find one that has a concave open area inside. We'll be making the electronics as small and flat as possible, but a scooped out pendant like this turtle will give you a little more space to work with.

Or, you can download & print (link) or order one of these 3D-printed pendants (link) which are just the right size and shape. Order in metal to be super fancy! Or order him in white or colored plastic and paint him with shiny spray paint or enamel.

Light Diffusion

Knead a little ball of polymer clay until it's soft, then press it into the back of the pendant. You want a very thin layer -- as thin as you can get it without compromising stability. Work it gently into place, then hold it up to the light to be sure it will work as a light diffuser, and not as a light blocker.
For this Sculpey glow-in-the-dark clay, I baked my little turtle at 275 degrees for 10 minutes. Read the directions, every kind of clay is a little different.

If you're using a 3D Printed plastic pendant, this will still work. The polymer clay wants a pretty low temperature. Don't worry.. you won't melt him!

---

**LED Build**

**Charging System**

In order to make this necklace as user-friendly as possible, we'll add a battery charger right into the design. That way you can just plug your necklace in to your computer with a USB Micro cable whenever he gets dim. (This also makes him into a wonderful gift for people who are not as tech-savvy as you are).
Solder three header pins into the charging backpack.

Take a utility knife and break the trace between the two connected pins next to the header. This will enable you to add an on/off switch to the necklace.

Solder a long (1.5 - 2 foot) black silicone 26awg wire to each of these two pins. These will later become your necklace cords so make them long enough to reach around your neck when the board is hanging at pendant height.
Solder the 3 header pins into the Trinket Pro using the BUS, G, and BAT+ pins next to the USB port. Trim off any extra header sticking out on the back side.

LED Soldering

Solder a power, ground and data wire to the Neopixel Jewel as shown. It works best to have the wires come in from the LED side and solder on the back side.

Solid core wires work best here! You want them to hold their shape when you bend them and stranded wires are tougher to work with in this case.

Putting it All Together

Make a little sandwich of all your electronics with the LEDs on top, the battery next, then the Trinket Pro and the charging backpack on the bottom. The battery pack itself
then acts as a non-conductive insulator between the metal pendant and the Pro Trinket board.

Bend the Neopixel Jewel wires around and line them up with their corresponding pins on the Trinket Pro. Cut them to length while you’re holding the battery in place.

Pull the battery out of the way and solder the Neopixel Jewel to the board, being sure to leave space to put the battery back when you’re done.

Jewel PWR -> Trinket JST +
Jewel G -> Trinket G
Jewel IN -> Trinket 9

Wrap the battery in electrical tape to avoid any shorts between components and help with strain relief.

Plug in your JST connector and slip all the components into place in their neat little stack. Wind the wire from the battery around so it's hidden, or you can shorten the wires and splice them back together if you prefer. Set them in your pendant and admire your handiwork.
Clasp & Finishing

Adding the Clasp

Thread the two long black wires through the top of your pendant. I wound a piece of jewelry wire around the neck of my turtle to hold the wires in place and help with strain relief.
Hold your necklace in place around your neck and decide how long you'd like it to be. Add an extra inch or so to the wires and snip.

Strip the wires that full inch -- you want a lot of bare wire here -- and slide a short piece of heat shrink tubing over the ends.

Slip the bare strands through the hole on each side of your magnetic clasp, just about all the way up to the wire shielding. Wind the bare ends around and around and around until the necklace clasp feels secure. If you can, thread the wires through the clasp's hole a second time. You want a lot of metal-on-metal connection here.

DO NOT SOLDER these in place! A weird property of magnets is that if you heat them up past around 500 degrees, they will lose their magnetism. Sad Christmas! So wind the wires around really really well, then add a dab of hot glue and slide the heat shrink over while the hot glue is wet, then shrink it on down. Pull gently on the wire and the magnet to make sure they're firmly in contact while the glue sets up so you have a good strong connection.
Finishing

Add a little hot glue around the inside of the pendant and press your lights in place. Try it on! How does it feel? (If the JST connector is a little too prickly and cornery for you, sand down the edges a bit)

Advanced Option:

If that JST connector is just too bulky (and it's amazing how bulky a JST connector can be), advanced users can desolder it from the charging backpack and solder the wires from the battery directly to the JST pads. Be sure not to mix up positive and negative battery wires or the whole thing might just catch fire. Cover up the connections with some hot glue to secure them in place.

Care & Feeding

Enjoy your necklace! To charge, simply plug a micro USB cable into the port on the Trinket for a few hours. The charger has an indicator light to tell you when it's done. Depending on how you decide to set the brightness in the code, this necklace will glow all night long -- and when the battery dies, the glow in the dark polymer clay will take over!