NeoPixel LED Necklace Insert with USB Charging

Created by Erin St Blaine

https://learn.adafruit.com/neopixel-led-necklace-insert-with-usb-charging

Last updated on 2023-08-29 04:30:35 PM EDT
Overview

Sometimes you want to add animated lights to your costume or project and you're really limited on space. Necklaces are a great example of this - a glowing pendant can be so striking, but if it's too big and bulky it can cross the line to "gaudy" pretty quickly.

This tutorial aims to make the smallest possible NeoPixel module or insert for your project. We've worked to make it super small, at around 1/2" thick and about 2" across.

This tiny package still contains all the niceties:

- 7-12 addressable NeoPixel lights
- A compact on/off switch
- A rechargeable battery with 4+ hours of battery life
- Onboard USB charging capability

We're using an ItsyBitsy M0 microcontroller, and we've included an easy-to-customize code sample as well.
Difficulty

Electronics Build

The physical build is the tricky party with this project.

This is an intermediate project that requires some tight soldering and careful measuring and cutting. You'll need to work small, and working small can be a challenge. But it's all through-hole soldering -- nothing fancy -- so it's definitely doable with a good soldering iron and a little patience.

Software

We've provided CircuitPython code for this project. The code is pretty straightforward and easy to customize with a few keystrokes.

Price Point

We've tried to keep this project fairly inexpensive so it's easy to fit into your cosplay budget. With a NeoPixel jewel, the cost of this project is around $25-$30 including everything you need.
Parts Needed

**Adafruit ItsyBitsy M0 Express - for CircuitPython & Arduino IDE**
What's smaller than a Feather but larger than a Trinket? It's an Adafruit ItsyBitsy M0 Express! Small, powerful, with a rockin' ATSAMD21 Cortex M0...
https://www.adafruit.com/product/3727

**Adafruit Lithium Polymer Battery - 3.7v 100mAh**
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/1570

**Adafruit LiIon/LiPoly Backpack Add-On for Pro Trinket/ItsyBitsy**
If you have an ItsyBitsy or Pro Trinket you probably know it's the perfect little size for a portable project. This LiPoly backpack makes it really easy to do! Instead of wiring 2...
https://www.adafruit.com/product/2124
Breadboard-friendly SPDT Slide Switch
These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They’re easy to switch...
https://www.adafruit.com/product/805

NeoPixel Choices
For my necklace, I’m using a NeoPixel Jewel with 7 pixels. This project would also work well with a 12 or 16 pixel NeoPixel ring if you want more lights.

And in case you’re wanting to pack in as many lights as possible, here’s a tip: the NeoPixel jewel fits perfectly inside the 16 pixel ring! So you can pack in 22 dense little lights within the 2 inch radius, if that’s what your project calls for.

NeoPixel Jewel - 7 x 5050 RGB LED with Integrated Drivers
Be the belle of the ball with the NeoPixel Jewel! We fit seven of our tiny 5050 (5mm x 5mm) smart RGB LEDs onto a beautiful, round PCB with mounting holes and a...
https://www.adafruit.com/product/2226
NeoPixel Ring - 12 x 5050 RGB LED with Integrated Drivers
Round and round and round they go! 12 ultra bright smart LED NeoPixels are arranged in a circle with 1.5" (37mm) outer diameter. The rings are 'chainable' - connect the...
https://www.adafruit.com/product/1643

NeoPixel Ring - 16 x 5050 RGB LED with Integrated Drivers
Round and round and round they go! 16 ultra bright smart LED NeoPixels are arranged in a circle with 1.75" (44.5mm) outer diameter. The rings are 'chainable' - connect the...
https://www.adafruit.com/product/1463
Wiring Diagram

NeoPixel Jewel or ring to Itsy Bitsy:

- PWR --> 5V
- G --> G
- Data IN --> A1

LiPoly Charger to Itsy Bitsy:

- BAT --> BAT
- G --> G
- 5V --> USB

We'll wire the switch to the two switch pins on the LiPoly charger.

Software

I'm using CircuitPython and the FancyLED library by Phil Burgess to make a lovely color palette based animation for my necklace. If you want softly shifting mood light animations, this is the code for you.

I really like the FancyLED library for small color-shifting projects. At the time of writing, it's still a bit slow if you have a large number of LEDs. But for tiny projects like this, it works great and is very easy to customize with your own colors.

The ItsyBitsy also works with Arduino or with Microsoft MakeCode, so if you prefer using one of those platforms, we have lots of tutorials and code samples in the Adafruit Learning System.
We need to do a bit of setup to get the ItsyBitsy working with CircuitPython. Here's what's on this page:

1. Install the latest version of CircuitPython on the board
2. Install the necessary CircuitPython libraries
3. Copy and update the Python code
4. Save the code to your ItsyBitsy

Ready to start? Here we go!

Install CircuitPython

This guide tells you all you need to know about CircuitPython:

https://learn.adafruit.com/welcome-to-circuitpython/installing-circuitpython

For now, I'll just cherry-pick the necessaries, but be sure to read through the guide to get all the nitty gritty details and troubleshooting tips.

Scroll down until you find the ItsyBitsy board you're using and click on it. Make sure you've got the right one - I'm using an M0, but there's also an M4 board option available.

Then click Download for the latest release. A file will download to your computer with a file extension of uf2. Take note of which version you downloaded.

Plug your ItsyBitsy into your computer with a USB cable. You may need to click or double-click the Reset button. The board will appear on your computer as a drive called ITSYBOOT. Drag the file you just downloaded onto this drive to install CircuitPython. The disk drive name ITSYBOOT will magically change to read CIRCUIT PY.

Note: if you don't see ITSYBOOT, but instead see a drive called CIRCUIIPY, that means CircuitPython is already installed. You probably still want to follow the instructions to update CircuitPython to the latest version.
Install Libraries

Now we need to install a few libraries onto our board as well. Here's a guide that tells you all you'll ever want to know about installing libraries:

https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-libraries

I'll just hit the highlights again to get you up and running.

Download Circuit Python Library Bundle

The above button takes you to a page where you can download the latest library release. Click the big purple button to do so.

Now go to your CIRCUITPY drive and create a new folder called lib. Unzip the Library bundle and find:

- adafruit_fancyled
- neopixel.mpy

Drag these two folders/files into your brand new lib folder.

Upload the Code

The last thing we need to add is a file called code.py on the CIRCUITPY drive. This is where the ItsyBitsy will look for actual instructions on what to do. Copy the code below into a text or code editor -- we recommend the Mu editor which can be downloaded here.
```
import board
import neopixel
import adafruit_fancyled.adafruit_fancyled as fancy

NUM_LEDS = 15

# Define your palettes. Add as many colors as you like.
# You can use CRGB, CHSV or Hex format, or any combination therein
# Select which palette you're using below the palette definitions

palette_fire = [fancy.CRGB(0, 0, 0),        # Black
                fancy.CHSV(1.0),            # Red
                fancy.CRGB(1.0, 1.0, 0.0),  # Yellow
                0xFFFFFF,]                  # White

palette_water = [fancy.CRGB(0, 214, 214), # blues and cyans
                 fancy.CRGB(0, 92, 160),
                 fancy.CRGB(0, 123, 255),
                 fancy.CRGB(0, 100, 200),
                 fancy.CRGB(0, 120, 210),
                 fancy.CRGB(0, 123, 255),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 28, 214),
                 fancy.CRGB(0, 68, 200),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 28, 214),
                 fancy.CRGB(0, 68, 200),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 200, 50),
                 fancy.CRGB(0, 200, 80),
                 fancy.CRGB(0, 200, 20),
                 fancy.CRGB(0, 100, 50),
                 fancy.CRGB(0, 150, 50),
]

palette_forest = [0xa6db97,
                  0xc6de50,
                  0x2a7a02,
                  0x5fb862,
                  0x314a32,
                  0xd5e8d6,]

palette_cloud = [fancy.CHSV(0.8, 1.0, 1.0),
                 fancy.CHSV(0.6, 0.8, 0.7),
                 fancy.CHSV(0.7, 1.0, 0.8),]

# Choose your active palette
palette = palette_water

# Declare a NeoPixel object on pin A1 with NUM_LEDS pixels, no auto-write.
# Set brightness to max because we'll be using FancyLED's brightness control.
# pixels = neopixel.NeoPixel(board.A1, NUM_LEDS, brightness=1.0,
#                           auto_write=False)
OFFSET = 0  # Positional offset into color palette to get it to 'spin'

while True:
    for i in range(NUM_LEDS):
        pixels[i] = palette[(i + OFFSET) % len(palette)]
```
# Load each pixel's color from the palette using an offset, run it
# through the gamma function, pack RGB value and assign to pixel.
color = fancy.palette_lookup(palette, OFFSET + i / NUM_LEDS)
color = fancy.gamma_adjust(color, brightness=0.25)
pixels[i] = color.pack()
pixels.show()
OFFSET += 0.005  # Bigger number = faster spin

NUM_LEDS and Animation Flow

Once you have the code in your editor, look near the top and find this line:

```python
num_leds = 15
```

For the moment, change this number to reflect the total number of lights you have in your project.

Once your project is built and you're at the testing stage, come back here and make this number bigger, just to see the effect it has on the lights.

The code will automatically space your color palettes out to fill the total number of LEDs you have. If you have 7 colors in your palette and 7 pixels, you'll get one pixel per color. If you have 7 colors in your palette and 14 pixels, you'll get two pixels per color.

All the colors will be there, they'll just be more spread out. Since the palette is rotating along the strip, all the colors will eventually appear on the first 7 pixels. So, one quick and easy way to make the animation slower and smoother is to tell CircuitPython you have more NeoPixels than you actually have.

My necklace has just 7 pixels, but my final code pretends there are 15 pixels. This is a slightly hack-y way to fine tune the motion and color change, but it's really easy to do and works great! As long as your reported number of pixels is equal to or greater than your actual number of pixels, your lights will look fantastic.

Don't make this number smaller than your actual number of pixels, or some of your pixels won't light up at all.

Customizing Palettes

I've added four different color palettes for the animations accessed from the Control Pad: a "fire" and a "water" palette, plus "forest" and "cloud" colors. You can customize
these fairly easily in the code. The power of the FancyLED library allows you so much control when it comes to choosing custom colors and animating them smoothly.

Find the palette definitions in the code:

```python
# Define your palettes. Add as many colors as you like. 
# You can use CRGB, CHSV or Hex format, or any combination therein
# Select which palette you're using below the palette definitions

palette_fire = [fancy.CRGB(0, 0, 0),        #Black
                fancy.CHSV(1.0),            #Red
                fancy.CRGB(1.0, 1.0, 0.0),  #Yellow
                0xFFFFFF,]                  #White

palette_water = [fancy.CRGB(0, 214, 214), # blues and cyans
                 fancy.CRGB(0, 92, 160),
                 fancy.CRGB(0, 123, 255),
                 fancy.CRGB(0, 100, 200),
                 fancy.CRGB(0, 120, 210),
                 fancy.CRGB(0, 123, 255),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 28, 214),
                 fancy.CRGB(0, 68, 200),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 68, 214),
                 fancy.CRGB(0, 200, 50),
                 fancy.CRGB(0, 200, 80),
                 fancy.CRGB(0, 200, 20),
                 fancy.CRGB(0, 100, 50),
                 fancy.CRGB(0, 150, 50),]

palette_forest = [0xa6db97,
                  0xc6de50,
                  0x2a7a02,
                  0x5fb862,
                  0x314a32,
                  0xd5e8d6,]

palette_cloud = [fancy.CHSV(0.8, 1.0, 1.0),
                 fancy.CHSV(0.6, 0.8, 0.7),
                 fancy.CHSV(0.7, 1.0, 0.8),]
```

There are so many ways to define palettes. You can use CRGB values or CHSV values to choose colors, or use them both at the same time. You can also use HEX codes. There are also multiple ways to declare values and a lot of control over how spread out the gradients can be.

This is explained in detail in the FancyLED guide so take a look to find out all you need to know about creating your own custom color palettes.
My favorite way to pick colors is to use the Google Color Picker. Just drag the sliders to the color you want and you'll get the values to plug into your code.

Remember that lighter colors (with more white) will appear brighter when mapped to LEDs, and darker colors (with more black) will appear dimmer.

Copy the HEX code from the colors. Get rid of the "#" and replace it with "0x" (as I've done in the palette_forest example above). Don't forget to put a comma between each color you choose.

Or, you can use the RGB values it gives you. Look at the palette_water definitions to see these being used.

HSV color values are available in the color picker too, but they don't correspond exactly to the HSV values that CircuitPython is looking for. It's possible to modify them to work, but easier to stick with RGB or HEX if you're a beginner.

Select Your Palette

```python
#choose your active palette
palette = palette_water
```

Just below the palette definitions you'll see a place to choose which palette you want to use. I like to save multiple palettes in the code so I can re-use the code on multiple projects, and just change this one line to get a totally different color gradient.
Save Your Code

Save the code on your CIRCUITPY drive, called code.py.

CircuitPython has a slightly different paradigm than Arduino or MakeCode. It works more like a USB stick, where you save the file to the drive and the code can be opened and edited directly on your microcontroller. You can even save multiple versions of the code on the CIRCUITPY drive if you'd like. The one you call code.py will be the one that runs.

Electronics Assembly

Prep your Wires

I'm using solid core wires for all the connections between the boards and NeoPixels, and silicone stranded wires to reach to the on/off switch. The solid core wires will give us strength and stability for the components that need to stack tightly together, and the stranded wires will give us flexibility in terms of the on/off switch placement.

Any type of wires will work, but I found this method to be the easiest to assemble.

Cut two small pieces of solid core wire in red, white, and black.

Cut two slightly longer pieces of silicone stranded wire for the on/off switch, and two pieces of heat shrink tubing.
On/Off Switch

Connect a stranded wire to the middle leg and to one (it doesn't matter which) of the outside legs of your on/off switch. Solder and cover with heat shrink.

Battery Charger

To enable the on/off switch take a knife and scratch the copper trace between the two pins marked Pwr Switch. Solder the two stranded wires from your switch into these two pins. It doesn't matter which wire goes into which hole.
Solder a solid core red wire into BAT, a black wire into G, and a white wire into 5v.

Jewel Assembly

Solder the other solid core wires into the NeoPixel jewel:
Red into PWR
Black into G
White into DIN
I've found it's easiest to insert the wires into the holes from the front and solder on the back of the jewel. It seems counter-intuitive, but it's very hard to reach between the LEDs on the front of the jewel with your soldering iron without damaging the NeoPixels.
Putting It All Together

The LiPoly backpack is designed to sit on top of the ItsyBitsy, but that would make our components stack too thick. We'll place it right next to the ItsyBitsy for a lower final profile.

Place the charger alongside the ItsyBitsy and trim the wires carefully to reach the appropriate holes:

- BAT to BAT
- G to G
- 5v to USB

The wires will cross each other but it's possible to get them into a nice, neat package. Solder them in place.
Flip the ItsyBitsy and charger over - we'll attach our NeoPixels to the back of the boards. Center your Jewel and solder the white data wire to pin A1.

I'm using a 3V ItsyBitsy, which has no more available ground pins. It does have a power pad and a ground pad available, however. There are two copper pads on the back of the board intended for soldering on a JST connector, in case you want to connect a battery directly to the ItsyBitsy. We can solder our NeoPixel PWR and G wires to these pads.

Tin the pads generously, then trim and strip the red and black wires. Attach the red wire to the right side pad (with the USB port facing away from you) and the black wire to the left side pad.

Plug your battery into the battery charger's JST port and test to be sure everything is working. If you've uploaded your code, the NeoPixels should light up in a beautiful shifting ocean palette.

Advanced Battery Connection Option

If this is still too big and you want to make it even smaller, you can shave a couple more millimeters off by connecting the battery directly to the charger instead of using the JST port. The technique is similar to the one you just finished - we'll solder directly to the copper pads currently holding the JST connector in place on the LiPoly backpack.
This is an advanced option for a reason! Lithium Polymer batteries can be dangerous if shorted or punctured. They can catch fire and fill your lungs with toxic smoke. So be sure you know what you’re doing before attempting this modification. A sloppy solder job could wreck your project.

Flip your power switch to the OFF position.

Use flush cutters to carefully snip through the solder holding the front of the JST connector to the battery charger. You should be able to carefully snip both corners and then gently pull the JST connector off.

Trim off your battery's JST connector by snipping the red wire ONLY, then soldering to the + pad. Then snip the black wire and solder to the - pad.

Don't cut both wires at once as this could short your battery through the metal snips. Cutting one wire at a time, then soldering, is the safest way to avoid a short, which could damage your battery or components.

Make extra sure your soldering job is clean and the two pads aren't bridged before flipping your switch to the ON position.

---

Build the Necklace

I’m using a 2” cabochon made from epoxy resin colored with alcohol inks and mica powder as the face of my necklace. The secret to getting good diffusion is to use white mica powder stirred into the resin. The white mica powder lets the light diffuse through. Swirling it into the tinted resin makes for a beautiful necklace even when your NeoPixels are turned off.
I did a lot of experimenting to find a diffusion formula I liked. I thought the gold mica powders would look fabulous, but they just didn't work as a diffuser - they blocked the light entirely.

The epoxy resin pour is beyond the scope of this guide, but I've made a [blog post](https://example.com) with product links and tips and tricks. Check it out!

For my necklace, I used hot glue to secure everything to the back of the resin cabochon. I started by gluing down the NeoPixel Jewel, then arranged the components as neatly as possible on top.

I want the USB port on the ItsyBitsy to be accessible - this is how we'll charge the necklace. I also want the on/off switch to be easy to reach. I wound the silicone wire leads around beneath the ItsyBitsy and hot glued it into place.
From the side, you can see that all the electronics are barely 1/4" thick - not even as thick as the resin cabochon. This should make a really nice low-profile necklace.

Wire Wrapping

There are lots of wire wrapping tutorials online that go into a lot of detail. I'm just a beginner but I'm still able to get some really beautiful designs. Here are the basics. I've gone into more detail over on my blog.

I cut a piece of leather to the same size of my pendant and hot glued it to the back, over the electronics.

Then I cut 6 pieces of wire to around 14" long. I'm using a combination of square and round wire for this particular pendant. The wire can be anywhere from 16g to 22g, depending on the look you're going for.
Line up your 6 pieces of wire - I found taping them to the table really helped keep them neat. Cut another small piece of wire and bend it around the 6 wires right in the center to hold them all together. This will become the bezel, the main loop that goes around your pendant.

Find something with a slightly smaller diameter than your pendant to use as a mandrel. I found a pill bottle that was the perfect size. Bend your bezel around the mandrel and secure it on the other side with another wire wrap around all 12 pieces of wire.
Place your pendant inside the bezel and tighten it down. Spread out the wires on the front and back edges of the bezel to hold the pendant securely in place. Make sure your on/off switch and your USB port are accessible. This method gives you a lot of freedom in designing your setting so you can cover any battery corners while keeping your components in easy reach.

Make a bail at the top for the necklace chain to go through using 4-5 of the wire ends. Wrap some of the rest around to secure the bail.

Finish up by artfully bending the rest of the wire tails into spirals or rosettes or whatever suits your fancy. I made a Pinterest board of wire wrapped jewelry examples for inspiration. Pinterest knows my secret heart so well.
Finish off with a necklace chain and enjoy your beautiful glowing amulet pendant.