Holiday Icicle Lights with Flair

Created by Phillip Burgess

https://learn.adafruit.com/holiday-icicle-lights-with-flair

Last updated on 2023-05-24 02:51:29 PM EDT

Table of Contents

Overview

I adore most anything with LEDs, but found those store-bought icicle or meteor lights a bit lacking. Nature rarely moves in straight lines and perfect intervals! In this project we'll make our own, applying some basic physics for an eye-catching drip effect. PLUS...these have a built-in magic trick, making a "splat" where they hit the ground!

This is a remix of an earlier project: the OOZE MASTER 3000 () from Halloween... mostly it's just larger now and uses different NeoPixel strips, and fits in a window rather than squeezing in a confined space. The code and principles are similar enough that you'd do well to skim that guide. Go ahead, I'll wait here, then explain the differences when you get back...

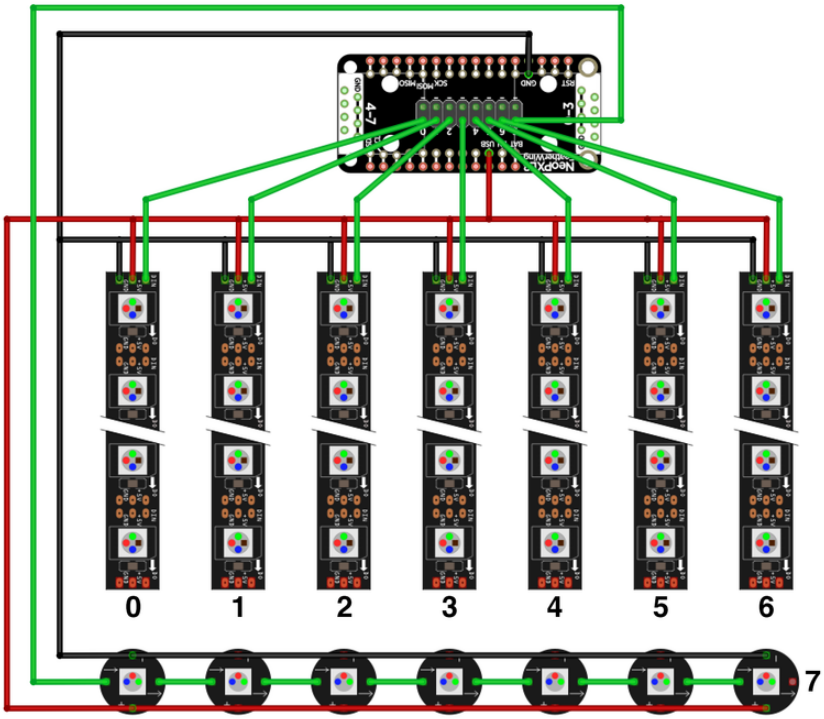
Open OOZE MASTER 3000 guide in a new window

The Halloween Ooze Master 3000 project was inspired by those unconvincing store-bought Christmas meteor lights, stepping it up with a bit of physics math to make the drips look "real." Now we've gone full-circle and it's a Christmas project again!

The same installation can do double-duty...set up for Halloween with a blood or ectoplasm color scheme, then afterward just update the code for a Christmas-like mood. Unless you're a horrible person like me...

Hardware

For posterity, let's take a look at the wiring schematic from the OOZE MASTER 3000 guide (you did at least skim that, yes?), to make better sense of the changes mentioned here...



Seven NeoPixel strips provide our icicles and drips, while a distinct eighth strand of separate NeoPixels makes the “drippings” at the bottom. It’s all powered from the Feather board’s USB port.

The concept is exactly the same as the prior project, just some of the parts are changed...

NeoPixels

With an architectural-scale project like this, we can use regular-width, 60-pixels-per-meter NeoPixel flex strip. Fortunately this is the most common variety around!

I started with a full 4 meter reel (), then divided into 8 half-meter strips. 7 for the icicles, 1 as a replacement spare. I already had some bits of NeoPixel strip from prior projects to cut up for the “splat” pixels, otherwise you might have to use that 8th half-meter for that instead, or some NeoPixel Mini Button PCBs ().

The drip illusion seems most convincing if the strips are in the vicinity of half the window height. I tested on two windows...one a little shorter than a meter, one a bit

taller...in either case, the half-meter strips looked great, so don't stress over getting this just right.

Higher-pixel-density strips are available and will work, but please read the notes later about power requirements.

Driving the Pixels

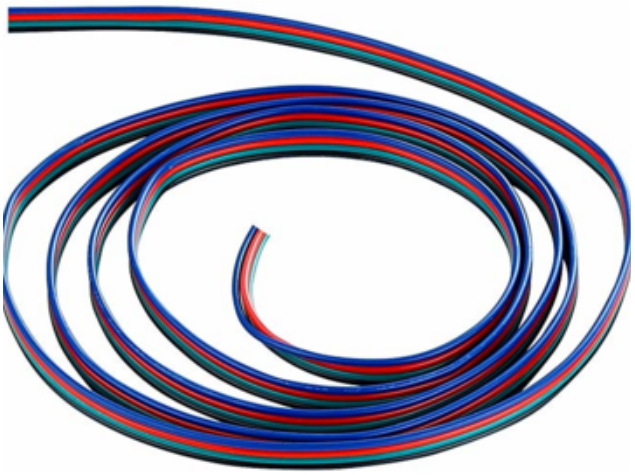
The hardware driving this is identical to the OOZE MASTER 3000 () project...which in turn refers back to the NeoPXL8 guide () for tips about compatibility. A number of Feather boards — M0, M4, RP2040 and ESP32-S3 boards — will work in conjunction with a NeoPXL8 FeatherWing, and a few non-Feathers can work with the NeoPXL8 Friend () (breakout) rather than the FeatherWing. Other microcontrollers like AVR or "classic" ESP will not work. The guide example code accompanying the NeoPXL8 library has comments regarding compatible hardware.

I also used the same Perma-Proto trick as before to make power rails for all the NeoPixel strips. But feel free to improvise with whatever supplies and techniques are available.

OPTIONAL: when possible, I like to build projects to be reconfigured or dismantled and used in other things...so, each of my 7 NeoPixel icicles has a 3-pin JST-SM () connector at the end.

JST-SM connectors are not water resistant. Fine for indoors, or under house eaves if weather allows. If installing your project further out...on a tree, in the snow, etc., there are waterproof connectors () better suited to this (and you'll want good adhesive-lined heat-shrink tubing for waterproof connections). Or forego the connectors and wire everything in permanently.

Wiring



An inordinate amount of wire was involved!

The Feather hardware resides at one corner of a window. The first NeoPixel strip needs only short wires, but each successive strip requires a few inches more than the previous one, then last one spanning all the way across (almost 48"). Finally, one more set of wires down to the “drips” at the bottom.

Each strip requires 5V, ground and data wires. Multi-conductor ribbon cable makes this process more pleasant...found some color-coded 4-conductor wire on Amazon (), cheap enough that I didn’t feel bad peeling off the 4th wire.

Power

The project is powered through the Feather board’s USB port, so you’ll need a higher amperage USB power supply (such as a phone charger) or a capacious USB battery bank if the installation isn’t well suited to running a long USB cable.

Other Parts

As with the OOZE MASTER 3000 project, you will also need:

- Soldering iron and related paraphernalia
- Sundry craft supplies and tools; this will vary with how you decide to implement the project. There will almost certainly be hot glue involved...but you might also need some clear packing tape, heat-shrink tubing, zip ties, hobby wood or edge molding and more. On the next page I’ll show some things I found.

Getting Crafty

Every installation will be different...it depends on where you're putting this and what tools, materials and techniques you have access to...so I won't go into intricate detail on the actual build. But I can show you a couple things along the way that might be insightful...

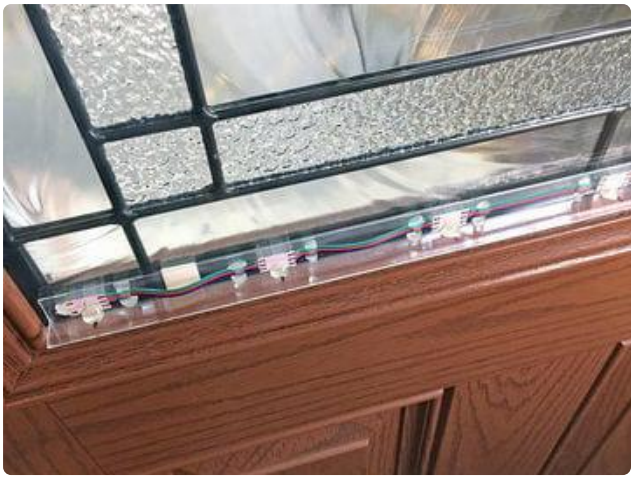
Even though my icicle lights will be inside the window, I wanted to keep the weatherproof covering in case these are re-used in different situations later. This requires sealing the ends of the strips with hot glue and 1/2" clear heat-shrink.

This is already well illustrated in both the NeoPixel Curtain () and Cyber Falls Wig () guides for ideas...please take a look at both!

In the original Ooze Master 3000 project, every strip was carefully "tuned" to a specific length. Here, all the strips are the same length. Half a meter is especially convenient because NeoPixel strips already have solder joints at exactly that interval!

If using this strictly indoors, you could skip the weatherproofing step if you want, but it does make the strips a little more robust for handling.

Either way, take an important lesson from the Ooze Master guide: test every strip at every opportunity. After soldering, after weatherproofing, etc.



My “splat” pixels were strip-end cuttings from prior NeoPixel projects. Originally I was going to glue these to a strip of molding or maybe a dowel, but the local hardware store had this clear “L” corner protector stuff. I cut it to length for the window, hot-glued the pixels in place, and held it to the window with a couple small pieces of double-stick foam tape.

The strips are held to the glass using clear packing tape...both top and bottom, because the strips tend to curl a little otherwise. Originally I thought the JST wires would be bearing the load, but no, tape handles it fine.

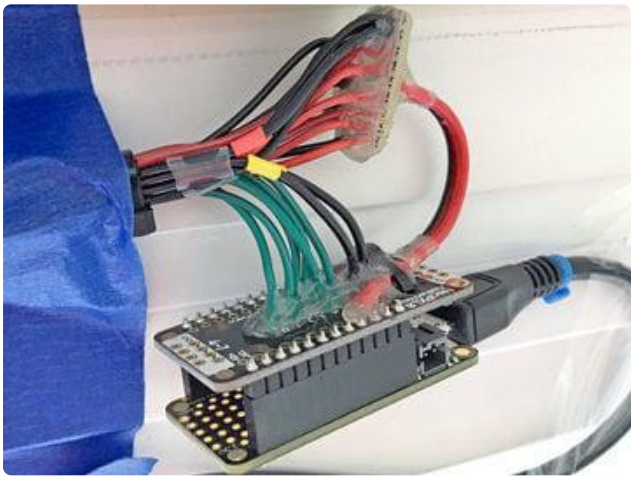
The tape will peel off the window cleanly unless it’s been left up way too long and/or in direct sunlight.

If you do end up with tape residue on your window, it can be loosened up with Goo-Gone®, WD-40® and/or a hair dryer.

Up top, the same plastic “L” channel (with holes drilled at suitable intervals) supports the wires, held with zip-ties. The Feather hardware is out of frame to the right.

Of course, feel free to improvise with whatever you have.

Looks pretty rough from the inside. From outside though, nobody has to know what’s going on, they just see pretty lights.



A view of the Feather, Wing and Perma-Proto power distribution after moving to a different window.

Since it's a temporary install and I'm not expecting company, it was all put up roughly with tape. Those with more patience and higher aesthetic standards can dress it up with an enclosure, maybe something 3D-printed.

As with the Ooze Master project, copious amounts of hot glue were used as strain relief for the wires. After testing that everything works, of course.

There's a whole lot of talk here about windows, but that's really not a requirement! Just like the original Ooze Master project, maybe there's something suitably holiday-themed you can wrap this around...a tree, a wreath, a snowman. Just make sure, if outdoors, that everything's suitably weatherproofed.

The next page is mirrored directly from the Ooze Master 3000 guide, so please excuse any reference to skulls and ectoplasm. That's just to get the code installed and working, then we'll proceed to make some Christmas-y adjustments!

Arduino Code

The software for this project uses the Arduino development environment. If this is your first time using this with Adafruit boards, please see the Feather M0 Setup guide here:

Adafruit Feather M0 Basic Proto: Arduino IDE Setup ()

Aside from installing the Arduino software, you'll need to add the Adafruit boards package as mentioned in the guide above.

Other boards...M4, RP2040, ESP32-S3...will have corresponding starter guides to get these going with the Arduino IDE. If it's your first time using them, use the search field to locate those guides.

Additionally you'll need a few prerequisite libraries, which can be installed with the Arduino Library Manager (Sketch→Include Library→Manage Libraries...). Search for and install:

- Adafruit_NeoPixel
- Adafruit_NeoPXL8
- Adafruit_ZeroDMA

Then fetch the code for this project, downloadable here:

Download OozeMaster3000 source code for Arduino

Select "Feather M0" (or whatever board you're using) from the Tools→Board menu, compile and verify that you can upload to the board. If so, great! Now we need to fine-tune some settings to match your project...

If using a Feather M4 (with M4 version of the NeoPXL8 FeatherWing), look for this line early in the code:

```
int8_t pins[8] = { PIN_SERIAL1_RX, PIN_SERIAL1_TX, MISO, 13, 5, SDA, A4, A3 };
```

Change the list of pins to the following:

```
int8_t pins[8] = { 13, 12, 11, 10, SCK, 5, 9, 6 };
```

The above change is only needed for Feather M4; if using Feather M0, you can leave the defaults.

Other boards will use different pins. If you've worked through the NeoPXL8 guide and puzzled through the strandtest example, this should all be sorted out by now. If using a Feather RP2040 SCORPIO, the sketch takes care of this part.

Now we can configure some visual things...

Near the top of the code is this line:

```
#define PIXEL_PITCH (1.0 / 150.0) // 150 pixels/m
```

This tells the code the "density" of your NeoPixel strip, in LEDs-per-meter. The default, 150.0, corresponds to the density of the ultra-skinny NeoPixel strip. If using different pixels, change this to match...60.0 and 144.0 are both pretty common strip

densities. The code needs to know this figure so the drippings fall at a physically plausible speed.

Q: Can I mix and match strip densities?

A: Electronically there's nothing preventing this, but the code as written works with a single density throughout.

Speaking of physically plausible speed, look for this a few lines further down:

```
#define G_CONST 9.806 // Standard acceleration due to gravity
```

Combined with the strip density figure, this makes the drips fall convincingly. 9.806 is the force of gravity on Earth, in m/s^2. A little math ties this all together to make the drips and the splats play well together and fool your brain.

The value there...9.086...is convincing enough for a thin liquid like water or blood. If you want something really viscous looking...rather than working viscosity calculations into the code, you can be like Q from Star Trek and just change the gravitational constant of the universe. Try values between 2 and 3 for a nice gravy-like feel.

Then, a short way down, look for this section:

```
uint8_t palette[][3] = {
  { 0, 255, 0 }, // Bright green ectoplasm
};
```

This defines the color of the drippings, as an RGB value. The default, {0,255,0}, is a pure green...good for ectoplasm. But you might want red instead, {255,0,0} for blood, or blue {0,0,255} or white {255,255,255} for something more Christmasy. Later we'll explain how to use multiple colors.

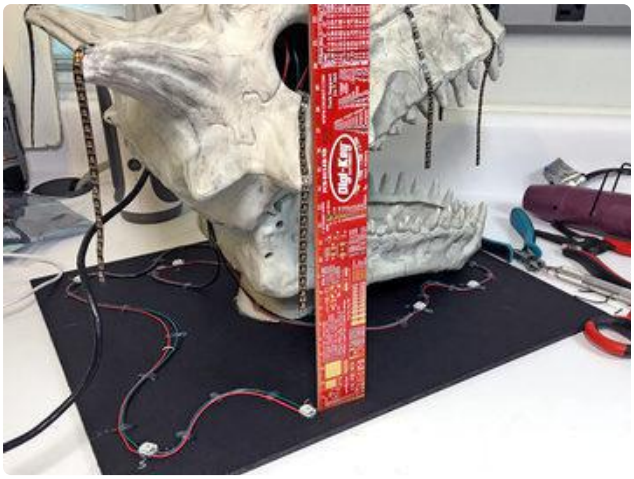
One more piece to this recipe...a little further along in the code, look for this table of numbers:

```
// THIS TABLE CONTAINS INFO FOR UP TO 8 NEOPIXEL DRIPS
  { 16,  7, 0.157, 0, 0 }, // NeoPXL8 output 0: 16 pixels long, drip pauses at index
7, 0.157 meters above ground, use palette colors 0-0
  { 19,  6, 0.174, 0, 0 }, // NeoPXL8 output 1: 19 pixels long, pause at index 6,
0.174 meters up
  { 18,  5, 0.195, 0, 0 }, // NeoPXL8 output 2: etc.
  { 17,  6, 0.16 , 0, 0 }, // NeoPXL8 output 3
  { 16,  1, 0.21 , 0, 0 }, // NeoPXL8 output 4
  { 16,  1, 0.21 , 0, 0 }, // NeoPXL8 output 5
  { 21, 10, 0.143, 0, 0 }, // NeoPXL8 output 6
```

The table holds information about our NeoPixel strips and real-world distances, to make the physics "actual size." Each line, one for each drip (up to 7) contains three numbers:

1. The length of the NeoPixel strip, in pixels.
2. The pixel index (starting at 0, up to length-1) where the dribble pauses before falling. This can be 0 if you want no dribble.
3. The distance from the dribble pixel (the one you just provided the index # for) to the splat pixel below, in meters.
4. A range of color indices to use from the palette[] array. With just one color defined for the default example case, both of these values are 0.

You can have an 8th drip but won't get splats in that case. Given a choice, I think the splats are way cooler.



Here I'm measuring drip-to-splat length for drip #6. It's about 143 millimeters, or 0.143 meters as you can see in the table. Not the topmost pixel, not the end of the drip, but the dribble pixel at the tip of the horn.

If measuring in inches, multiply by 0.0254 to convert to meters.

Get all your numbers and measurements in there, save the code (or a copy) and upload to your Halloween prop. If all goes well, the motion should physically correspond to the thing you've built! If it's close but not quite, tweak your table numbers...move the dribble pixel up or down a bit, change distances (or G_CONST) to get the timing right.

Code Configuration

Once you have the Arduino IDE working with the Feather M0 or M4 board, have the prerequisite libraries installed and can successfully compile the Ooze Master 3000 source code, then we can make some adjustments so it's less oozy and more Wintery ...

But first...if using a Feather M4 (rather than M0), see the note on the prior page about changing the pin numbers. Okay? Okay...

Near the top of the code, change the "dripColor" line like so:

```
uint8_t palette[][3] = {
  { 200, 240, 255 }, // Bluish-white
};
```

This makes a pleasing cool white color for melting snow. Solid blue `{ 0, 0, 255 }` also looks good. Although red and green are traditionally "Christmas colors," in this setting they look like blood and ectoplasm respectively, and are best reserved for Halloween.

Elsewhere, set `PIXEL_PITCH` to match your LED strips:

```
#define PIXEL_PITCH (1.0 / 60.0)         // 60 pixels/m
```

That's assuming the recommended 60/m NeoPixel strip...but if you've got something else like 144/m or 150/m, change the value accordingly.

Next line down sets how bright the "icicles" are:

```
#define ICE_BRIGHTNESS 20                // 20% icycle brightness
```

In the default Oozemaster code, this value is 0 so we just get dribbles and no icicles. 20% looks pretty good for faint always-lit icicles, but feel free to experiment.

Then, scrolling down a bit, there's a table with the NeoPixel strip specifications. Here's what I'm using for the front window:

```
} drip[] = {
  // THIS TABLE CONTAINS INFO FOR UP TO 8 NEOPIXEL DRIPS
  { 30,  6, 0.752, 0, 0 }, // NeoPXL8 output 0: 30 pixels long, drip pauses at index
6, 0.0.752 meters above splat
  { 30, 10, 0.685, 0, 0 }, // NeoPXL8 output 1: 30 pixels long, pause at index 10,
0.685 meters up
  { 30,  9, 0.702, 0, 0 }, // NeoPXL8 output 2: etc.
  { 30, 11, 0.669, 0, 0 }, // NeoPXL8 output 3
  { 30,  7, 0.735, 0, 0 }, // NeoPXL8 output 4
  { 30, 10, 0.685, 0, 0 }, // NeoPXL8 output 5
  { 30,  8, 0.719, 0, 0 }, // NeoPXL8 output 6
```

All the drips are 30 pixels long now, because that's how it was built (7 half-meter lengths, 60/m).

The "dribble" pixel indices are jumbled up a bit to add variety. This also establishes the lengths of the icicles that will be always-lit by the `ICE_BRIGHTNESS` value.

And the last number on each line...that's the height, in meters, of the "dribble" pixel above the "splat" pixel down below. Not the topmost pixel, not the end of the drip, but the dribble pixel.

If measuring in inches, multiply by 0.0254 to convert to meters.

Your numbers will be different because your windows are different, that's okay.

Fix It in Post

I accidentally soldered in my "drip" wires in the wrong order, with the longest one leading to NeoPXL8 pin 0 and the shortest to pin 6. It should have been the other way around. But I'd already buried those wires under hot glue, and the wire leading to the "splat" pixels wasn't long enough to just physically flip that piece around.

No worries, if you make the same mistake, we can easily compensate in software.

Down toward the end of the `loop()` function, look for a block of code starting like this:

```
if(N_DRIPS &lt; 8) { // Do splats unless there's an 8th drip defined
```

Then, a few lines down, look for the `set()` function call. The following change numbers the splats from 6 to 0 instead of 0 to 6:

```
set(7, N_DRIPS - 1 - i, x);
```

Alternately, this could be done by reversing the first six elements of the pins[] array.

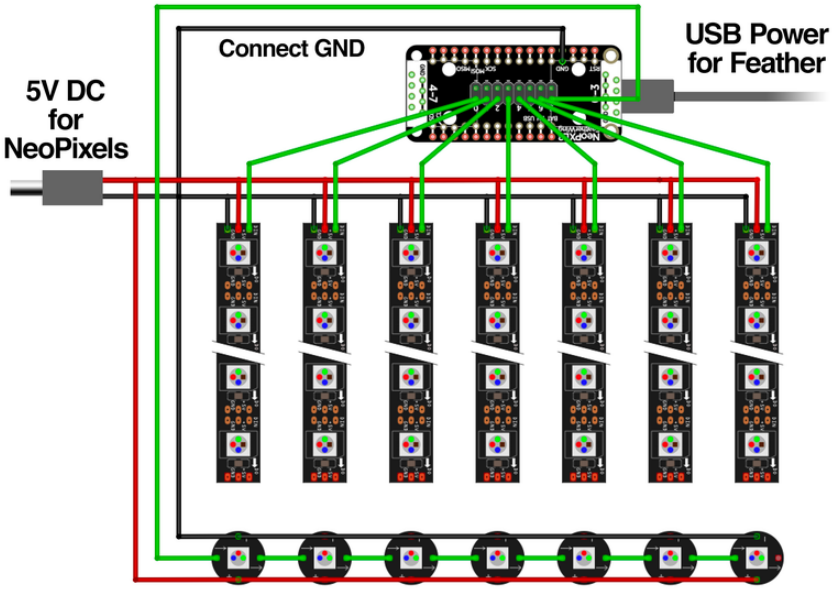
Notes for Big Installs

The schematic as shown on the "Hardware" page (and in the Ooze Master 3000) guide presume that everything will be powered through the Feather board's USB port.

With the number of NeoPixels we're using, and running the code as it's written, this is all fine and good.

However, quite often a thing occurs with NeoPixels. One moment you're animating a few pixels with a carefully planned formula...the next, you want way more pixels, all simultaneously lit in different colors and synchronized to Christmas tracks by Mannheim Steamroller. NeoPXL8 can handle all that data, but a USB power supply can't feed that much current.

I can't help with the Mannheim Steamroller part, but the power issue is pretty straightforward...you just need to power the Feather board (with NeoPXL8 wing) separate from the NeoPixels, with a ground wire linking the two:



Use a regulated 5V DC power supply for the NeoPixels, whatever capacity meets your needs...we have many to choose from, up to 10 Amps (), and these handy screw terminal block adapters () as well.

You don't need to do this if building the project as described. This is only if you find yourself taking things to the next level!

Deeper insight into NeoPixel power consumption can be found in the Adafruit NeoPixel Überguide (), NeoPixel Curtain () project and Sipping Power with NeoPixels ().