Four Seasons Fairy Bottle Lanterns

Created by Erin St Blaine

https://learn.adafruit.com/four-seasons-fairy-bottle-lanterns

Last updated on 2024-03-08 03:41:33 PM EST

Table of Contents

Overview

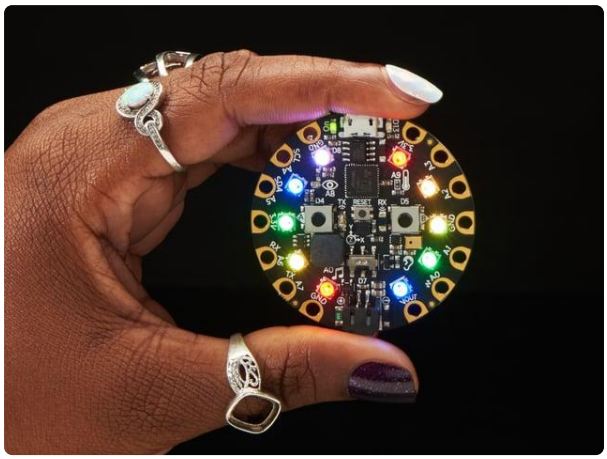
There's something magical about fairy lights in bottles. They evoke dreams of endless summer nights: catching fireflies at the bottom of the garden, telling ghost stories by firelight, or cuddling up with a book while a thunderstorm rages outside.

This project uses a Circuit Playground Express to control nine beautiful, crafty fairy bottles hanging from the branches of an indoor tree sculpture. Capacitive touch metal buttons select color modes based on the four seasons: a relaxing rainbow for summer, flickering candlelight for autumn, raindrops and lighting for winter, and a beautiful energetic and colorful animation for spring.

This is a fairly advanced project with a lot of tricky wiring and soldering. The coding is also fairly involved, creating four different modes and switching between them.

None of the steps are terribly difficult in-and-of themselves, but there are a lot of steps and a lot of things that can go wrong. But the end result is stunning in its beautiful simplicity, so with a bit of patience and plenty of time, you can make something wonderful.

Parts



Circuit Playground Express

Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...

https://www.adafruit.com/product/3333

Adafruit Mini Skinny NeoPixel Digital RGB LED Strip - 60 LED/m

So thin. So mini. So teeeeeeny-tiny. It's the 'skinny' version of our classic NeoPixel strips!These NeoPixel strips have 60 digitally-addressable pixel Mini LEDs per...

https://www.adafruit.com/product/2964

1 x NeoPixel Dots — https://www.adafruit.com/product/3631

4" pitch NeoPixel Dots

---

3 x Splitter — https://www.adafruit.com/product/402

4-1 JST Connector Splitter

---

20 x 2-Pin Connectors — https://www.adafruit.com/product/2880

2-Pin JST Connector Set

---

1 x 4-Pin Connector — https://www.adafruit.com/product/578

4-pin JST Connector

---

3 x Wire — https://www.adafruit.com/product/1970

26 AWG Wire in Various Colors

---

1 x USB Cable — https://www.adafruit.com/product/4148

2m USB Cable

5V 2A Power Supply
1 x Power Supply https://www.adafruit.com/product/276
5V 2A Power Supply

Materials Needed

Bottles

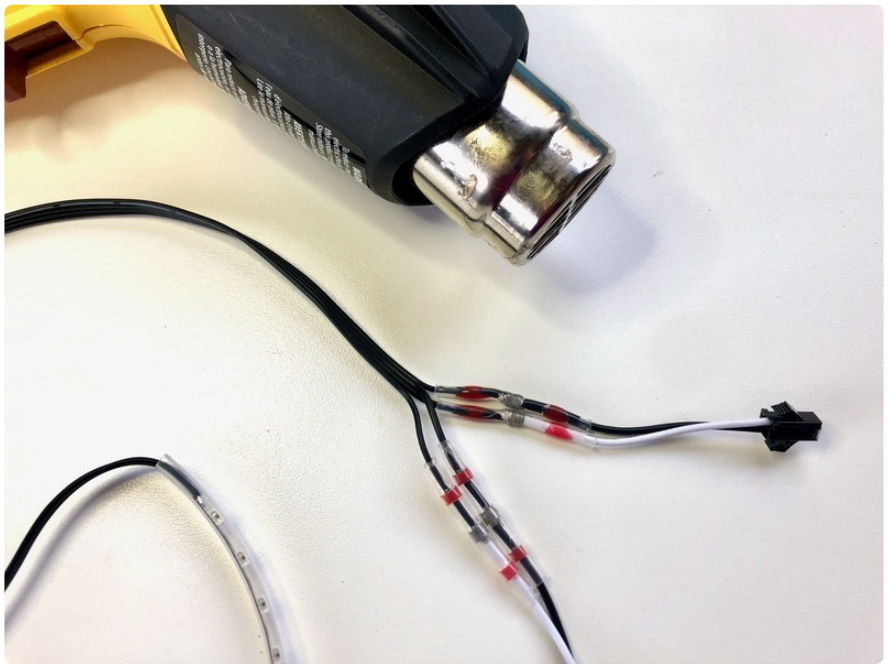
- Small glass bottles of various shapes & sizes
- Corks to fit in the bottles
- Iridescent cellophane or fantasy film
- Leather or rope scraps for hanging

Controller Pad

- Conductive metal charms (buy or make your own)
- Galvanized wire
- E6000 glue

Tools Needed

- Soldering iron & accessories
- Heat gun
- Wire strippers & snips
- Scissors or utility knife



Wire Connectors

There are a LOT of wire connections in this project. For the connections to the bottles, I used the traditional soldering-iron-and-heat-shrink attachment method. For the connections that needed to be made in place up in the tree, I made really good use of a new type of heat shrink connectors.

These connectors come in varying sizes and are really cool! They've simplified the process of connecting wire to one or two steps, and eliminated the need to use a soldering iron for wire-to-wire connections. The colored lines are hot-melt glue, and the silver bit in the middle is low temp solder.

I found them on Amazon here (https://adafru.it/lSD).

Twist your wires together securely, then slide the heat shrink connector over your twisted wire. Heat the whole thing with a heat gun and you've soldered, sealed, and glued your connection in one step. These things are a game-changer for in-place installations where a soldering iron is impractical to use. Magic!

Decorate the Bottles

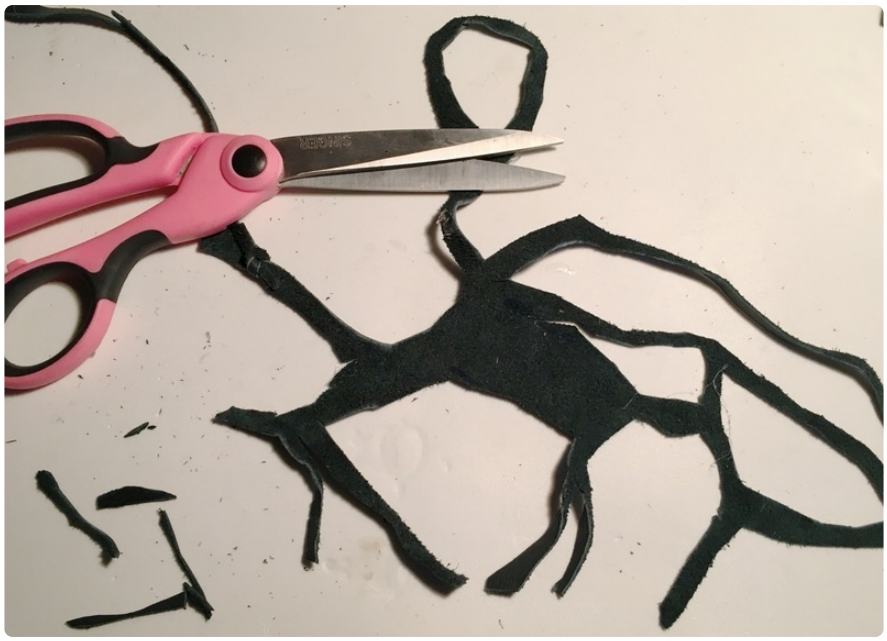
I chose a variety of shapes, sizes and colors for my bottles. I love the look of "theme and variation" this creates.

I ended up using mostly clear bottles, but threw in a few colored ones as well. The colored bottles add another dimension of interest to the final artwork, since they change the look of the LED animations -- for example, the red bottle doesn't let blue light through at all, so it goes "dark" during a mostly blue animation, while any white light animations look red. More variety, less coding!

I covered some of my bottles with Fantasy Film and some with plain craft-store iridescent cellophane. Fantasy film (https://adafru.it/lHc) is a product that looks like cellophane, but it's a little thicker, and bonds to itself seamlessly when heat is applied. It works quite a bit better than the cellophane for wrapping the bottles, but the cellophane is much cheaper and easier to obtain -- they have it at most craft stores in the wrapping paper section.

Get out your heat gun and cut a bunch of strips of fantasy film (https://adafru.it/lHd) or iridescent cellophane. Layer the film over your bottle and gently hit it with the heat. Watch it shrink and bond around your bottle (man this is fun!) Put a bunch of layers on until you're happy with the diffusion of the light shining through.

It's easier to work with smaller pieces of fantasy film, since they stick to each other really well. I had more success using a big piece of cellophane and gently shrinking it to cover the whole bottle.



Cut a bunch of strips into your leather scraps and artfully wrap it around the bottle, tying and gluing as needed. I left a few long strips and a long strand at the top, securely wrapped around the bottle's neck. This will bear the weight of the bottle and tie to a tree branch, so be sure it's secure. It's a much better idea to hang the bottle from the leather than from the wires, which will break if you tug on them too hard.

Code with MakeCode

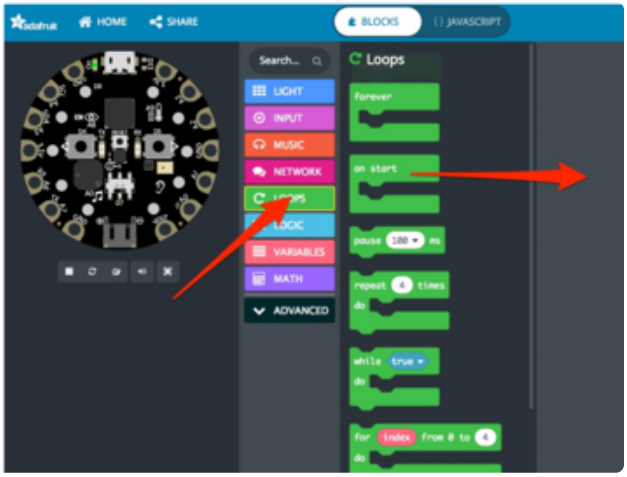
Microsoft's MakeCode editor is a great way to get your Circuit Playground Express to do things without having to muck about in actual code. It's a fun drag-and-drop editor that's easy to use and sneakily teaches you how to write code while you're making pretty things that light up.

The MakeCode project linked below has four different modes, one for each season of the year, plus a mode for "off". Each mode is triggered by a capacitive touch pad on the Circuit Playground Express. You can download the completed code and get up and running right away, or you can follow along and create your own code, adding your own customizations as you go.

Complete Fairy Bottles MakeCode

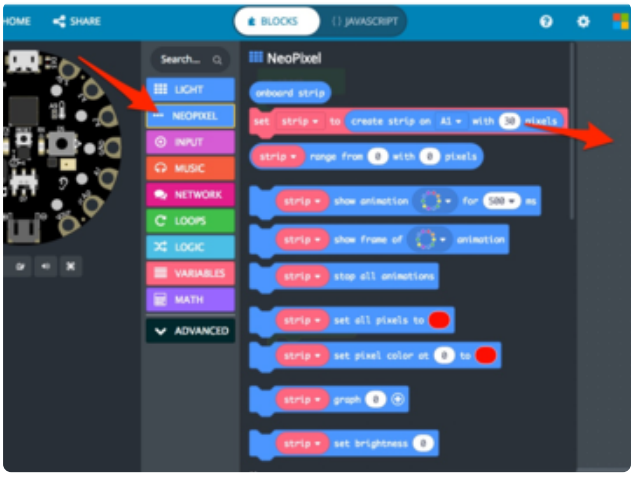
https://adafru.it/IFW

Our code has four different color modes: a high-energy "party" mode for spring, a relaxing rainbow for summer, flickering candle light for autumn, and a rainstorm with lighting for winter. This will make our MakeCode project a bit complex, so if it's your first time using MakeCode, head over to the MakeCode Intro Guide (https://adafru.it/AEp) and play around with a couple of the beginner projects to get a feel for how it all works. Once you're ready to go, head over to makecode.adafruit.com (https://adafru.it/wmd), (https://adafru.it/DGk) roll up your sleeves and it's time to dive in.



Click the Circuit Playground Express and then **New Project**. You'll see your workspace, with a Circuit Playground on the left, some tabs in the middle and your workspace on the right. Click the **LOOPS** tab, and find `on_start`. Drag it into your workspace.

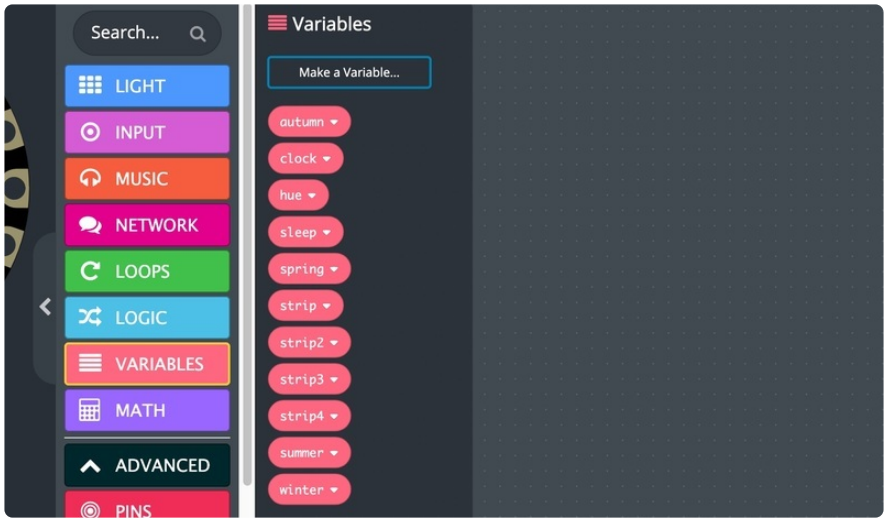
Whatever you put into the `on_start` loop will run once, when you first boot up your Circuit Playground Express. Whatever you put in the `forever` loop will run over and over again, forever.



Let's tell the Circuit Playground Express that we attached some lights to pin **A1**. Click the **LIGHT** tab and another sub-tab will appear labeled **NEOPIXEL**. Anything in the **LIGHT** tab refers to the lights on the face of the Circuit Playground Express itself. Anything in the **NEOPIXEL** tab will refer to lights that you add. Find `set strip to create strip on A1 with 200 pixels` (or whatever your total number of pixels in ALL bottles is, plus a couple extra) and drag it onto your workspace inside the `on_start` loop.

Variables

A variable is a placeholder or container for a number. We use them so we can change numbers willy-nilly, on the fly, or in multiple places at once. We'll set up all our variables now so we can use them later on in our code. Click the **VARIABLES** tab to create your variables.



I've created a variable for each of the four seasons: `autumn`, `spring`, `summer`, and `winter`, as well as one called `sleep` (for "off" mode).

I've got one called `clock`, which we'll use to time our lightning strikes in the winter animation.

There's one called `hue`, which we'll use in the spring and autumn animations to randomize our colors.

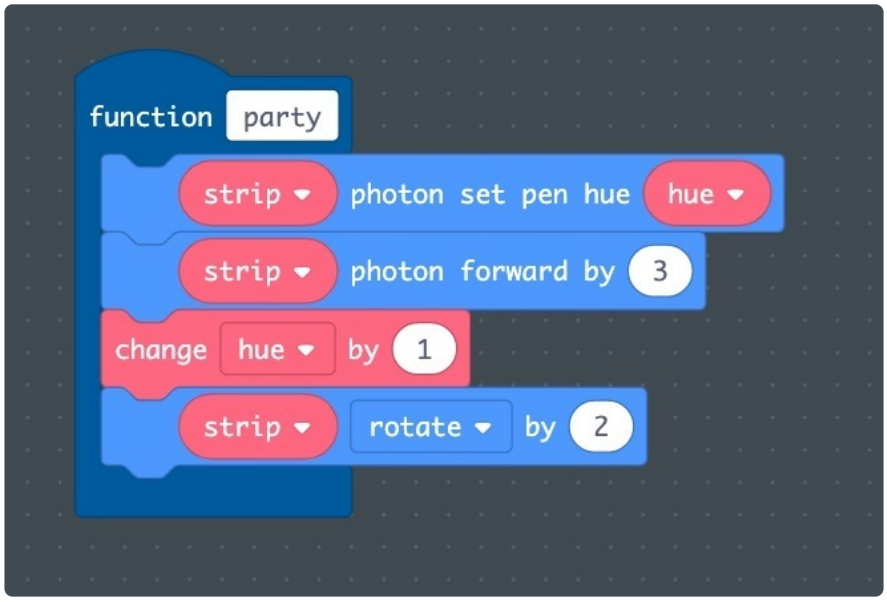
I've also created `strip2`, `strip3`, and `strip4`. These will be used to break up our strip into arrays, or chunks of LEDs which we can call individually, to apply animations to part of our strip at a time. We'll use these in our autumn animation as well.

Functions

I've set up a separate function for each of the seasons and one for "off". Functions are a fancy way of creating a separate box for a bunch of code which we can call when we want it, but it'll stay out of our way the rest of the time. You can create them from the **FUNCTIONS** menu which is found under the **ADVANCED** tab.

Name your functions something other than "spring" or "summer" -- since we already have variables set up using those names, we'll need to call our functions something different.

Spring / "Party" Mode



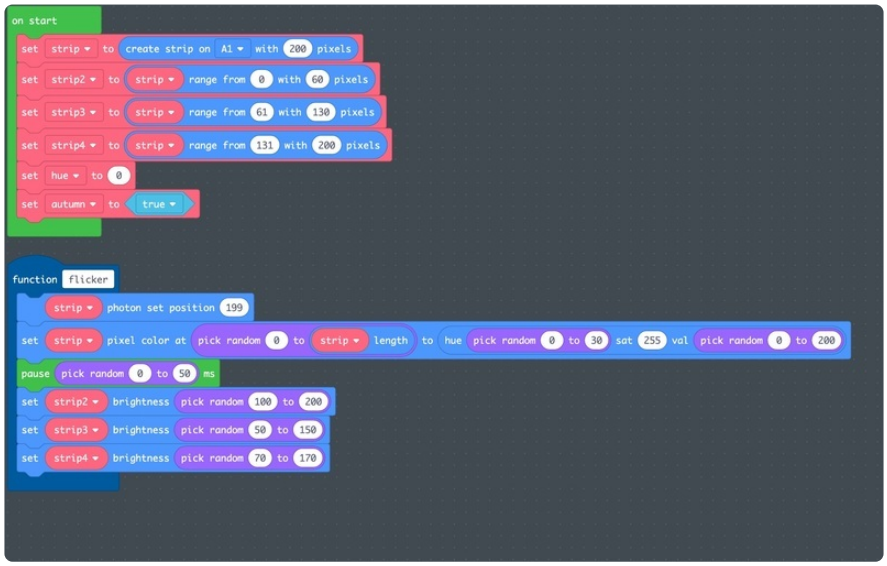
For spring, I want a lively animated mode with lots of color and motion. I'm using the **Photon** feature in this mode. **Photon** lights an LED in white and moves it along your LED strip, trailing color behind it.

By setting the pen color to our `hue` variable and changing our hue each time the photon moves, we're able to have the photon trail a rainbow behind it instead of just a solid color. Adding `strip rotate` gives more interest, as each light will slowly animate through the rainbow while it's waiting for the photon to come by. You can play with the numbers until you get an effect you like.

The **photon** feature has a bit of a bug, in that the white photon light tends to stay stuck "on" when you change to a different color mode. To work around this, I added a line to all the other modes setting the photon to LED #199 (shown below).

In reality I have fewer LEDs than 199 -- my total number is probably somewhere around 195 (I lost count at some point!). So setting the white photon light to #199 means it's moved off my strip entirely and is happily stuck "on" someplace in Imagination Land where it won't interfere with my project.

Autumn / "Flicker" Mode



For autumn, I want a halloween-y candle flame effect in my bottles. This is very similar to the flame effect I wrote for my paper mache dragon project, and I go into more detail about it here (https://adafru.it/IFX). I've added a little more complexity to this animation for this project.

Looking at the function: I'm starting with the `photon` bug fix described above. The next line creates the candle flicker. I'm choosing a random LED (the first `pick random`), then setting its hue to something between 0 and 30, and its intensity (`val`) to something between 0 and 200. This will make all the LEDs color in various shades of red and orange, with randomized brightness. Then, I pause for a random number of milliseconds to make the flicker.

Next, I randomized the brightness of the whole strip to add another dimension of flameyness. But when I tested this on the bottles, it looked fake -- all the bottles seemed to be flickering at the exact same time, which really wouldn't happen if there were actual candles in the bottles. How to make it look more natural? This is where the arrays come in.

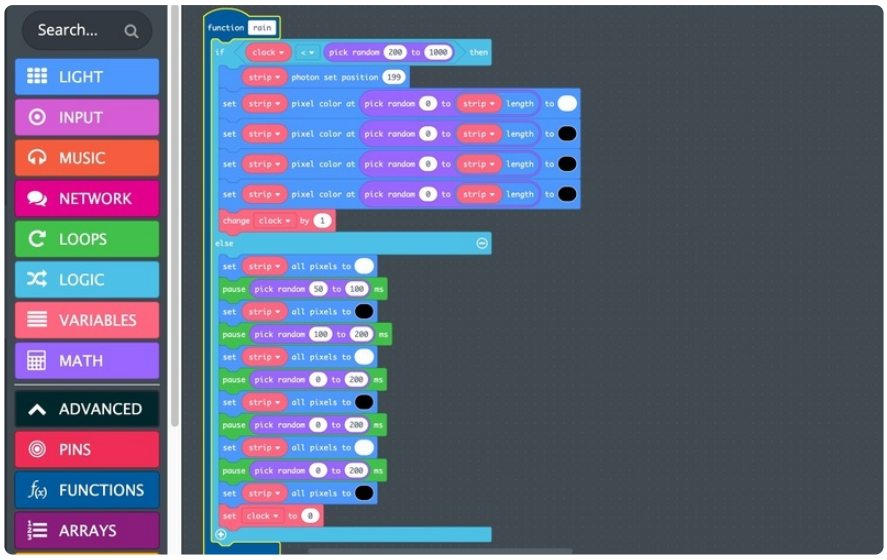
I added three lines to the `on start` loop, setting the variables I made earlier (`strip2, strip3, strip4`) to each represent roughly 1/3 of the total strip. Then I was able to vary the brightness of 1/3 of the strip at a time by randomizing brightness to those arrays. This makes the whole animation look a lot more natural.

Had I wanted to be very thorough, I could have made an array for each of my 9 bottles, using just the number of lights in each bottle, then varied the brightness of each bottle individually. But because of the way the bottles are spaced pretty far

apart and not sitting in a line, having 3 arrays was plenty for an illusion of randomness. If your bottles are all lined up on a shelf, you'll probably want separate arrays for each bottle so they don't flicker in "chunks."

This is my favorite mode, so I want it to be the default when the tree starts up. I added a line to my `on start` code setting my `autumn` variable to `true`. I also set `hue` to `0` to be sure I get a nice red color when the bottles start up.

Winter / "Rain" Mode



For winter time, I wanted the feel of raindrops dripping from the tree, punctuated by the occasional flash of lightning. The top half of the code makes the raindrops, and the bottom half makes the lightning. The `clock` variable is used to count out the time between strikes.

This is based on the raindrop animation I wrote for the FloraBrella project, and I go into more detail about that here (https://adafru.it/IFY).

Using an `if/then` statement (found under **LOGIC** tab) makes this possible. In English, this code says something like:

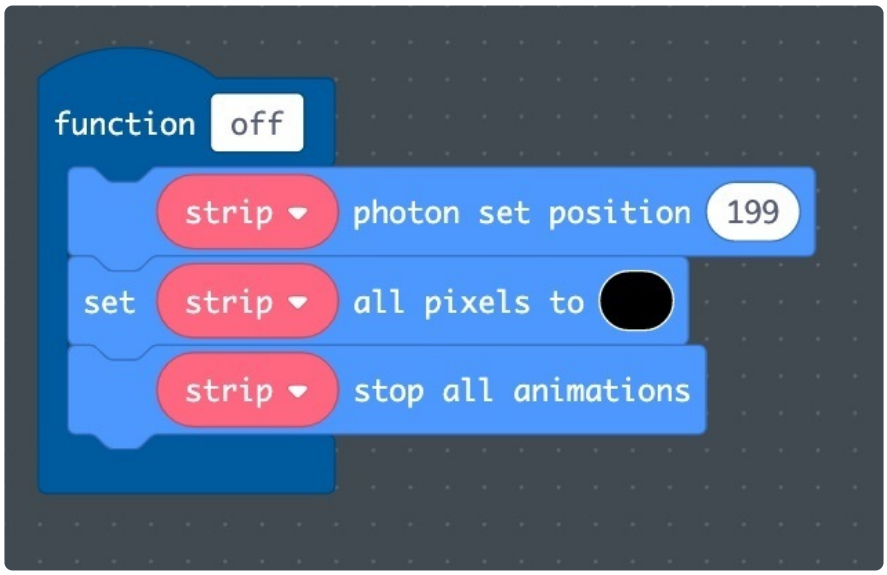
> If we haven't counted to some random number between 200 and 1000 yet, fix the photon bug and then turn on a white raindrop somewhere, while turning off some other raindrops. Then count up.
>
> Once we've counted to our random number, flash all the pixels three times for a short randomized amount of time, then turn them all off and set our counter back to 0.

It's easy to make the lightning more or less frequent by adjusting the values. This code looks really cool, and is also my favorite mode. (Yes, I'm allowed to have two favorites!)

Summer / Rainbow Mode

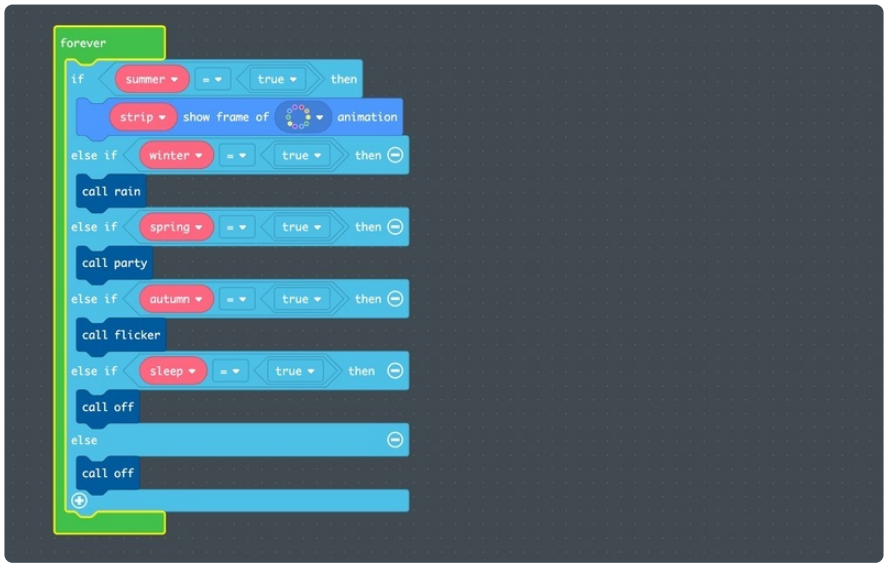
Summertime is lazy, so I kept this one simple and just used one of MakeCode's canned animations. This was so simple that I didn't bother to make a function for it, I just call it in the **FOREVER** loop, which we will get to shortly.

Off / "Sleep" Mode



Finally, our "sleep" mode fixes the photon bug, then turns all the LEDs to black / off, and stops any animations.

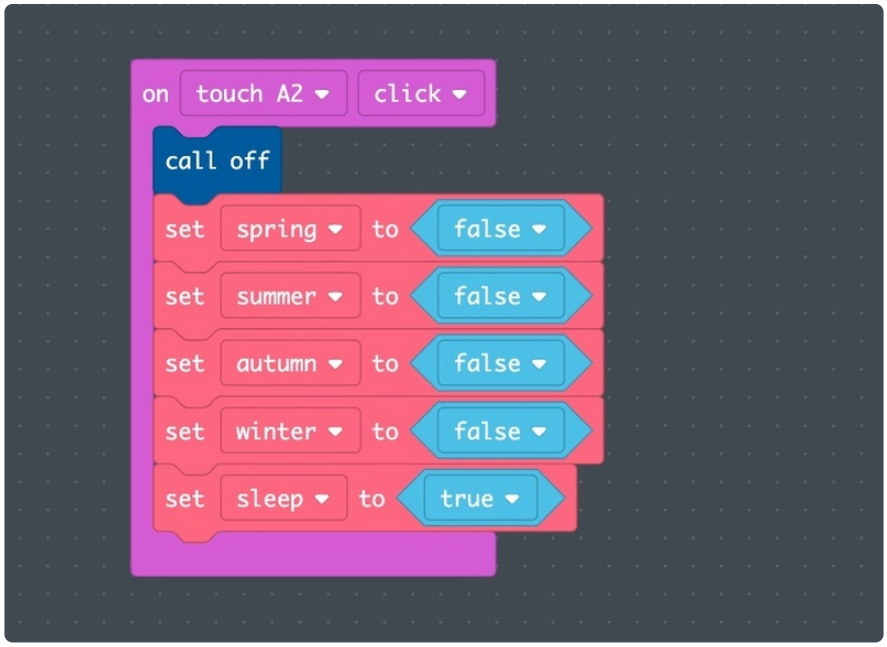
Calling the Functions



To make the animations play, we'll use an `if/then` statement inside our **FOREVER** loop. For summer, I used the canned rainbow animation (using `show frame of animation` gives you a bit more control over speed and seems to curtail bugs), and for each of the other modes I called the function written for that season. Using the functions makes the code so simple!

Inputs

Lastly, we need to hook up the Circuit Playground Express' capacitive touch pads to trigger the different modes.



I'm using **A2** for "off" mode. I grabbed `on button A click` from the **INPUT** tab and selected `touch A2`. First, I reset all the lights to off by calling the `off` function, so the previous mode doesn't bleed into the new mode. Then, I set all modes to `false` except the one I want active, which I set to `true`.

Finally, I copy/pasted this block a few times, setting each touch pad to trigger the desired season's animation.

I'm sure there's a fancier way to do this with math and less repetitive code, but this way makes intuitive sense to me and it works just fine.

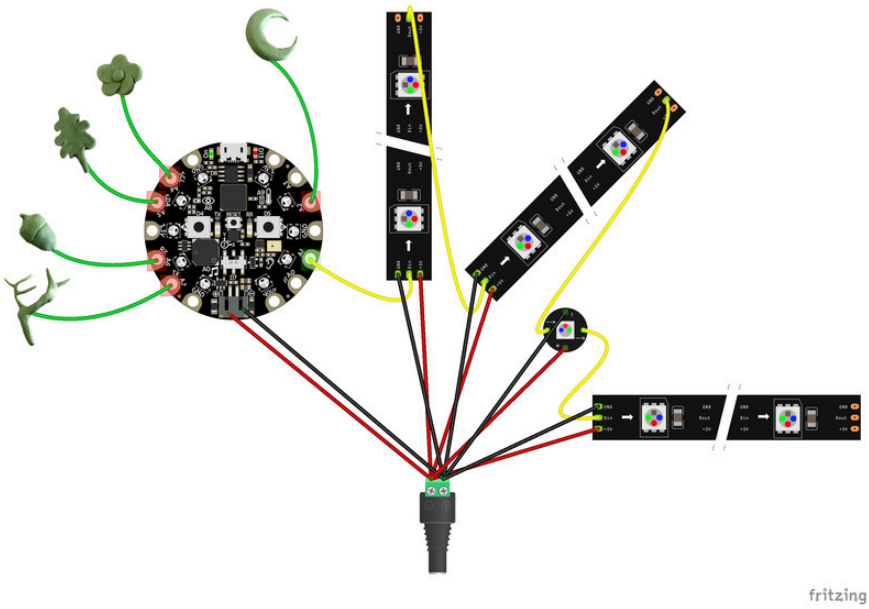
Give your code a name (I called mine Fairy Bottles) and save it. Click the Download button and a file will download to your computer. Plug your Circuit Playground Express into your USB port and click the **reset** button. You should see all the lights on the face of your Circuit Playground Express turn green, and a drive will appear on your computer called **CPLAYBOOT**. If you see a drive called **CIRCUITPY**, press the **reset** button a second time to get to **CPLAYBOOT**.

Drag the downloaded file onto this drive to program the Circuit Playground Express.

Once you've got your lights hooked up, touch the pads and see the modes change!

If you're having trouble with the download process, head over to the Circuit Playground Express (https://adafru.it/AQW) guide for some more detailed instructions and troubleshooting ideas.

Wiring



The wiring is the hardest part of this project. There are a lot of connections to make, and a lot of places where things can go wrong. Test each connection as you go, and you'll minimize any headaches later on.

Power Wires

Power that has to travel a long way down wires or through a lot of pixels can degrade with the resistance in the wires. In practice, this means the pixels at the end of a long run of wire will "brown out" -- they look dimmer than the pixels at the beginning. To avoid this, I've set up my wiring to run power directly to the bottles instead of following the data line, which must run in sequence through all the bottles.

Most bottles have a direct link to the power source. It's ok to daisy-chain one or two bottles, but you may need to add an additional ground wire between the final bottles in the chains if you find you've got a lot of flickering.

So, the power wires are set up in a "star-fish" or "home-run" configuration, where each bottle is on its own power line.

Data Wires

NeoPixel data wires MUST flow from **IN** to **OUT** in one unbroken chain if you want to be able to animate all the bottles individually.

The data wire is attached to the CPX pin **A1**, then goes to **IN** on the first bottle. Another line attaches to the **OUT** pin at the end of the NeoPixel strip, and heads to the next bottle in line where it will attach to **IN**. Fairly simple, yes?

However, NeoPixel data does not like to travel long distances over wire either. The signal can begin to degrade if there is more than around 3-4 feet of wire between your bottles.

Some of my bottles were placed further than this, so my solution was to use a single pixel or a bit of NeoPixel strip as a "repeater" for those long wire runs. The pixel can be hidden among the branches (or can be covered with black tape, if it's not easy to hide) and this keeps your data clean and flowing.

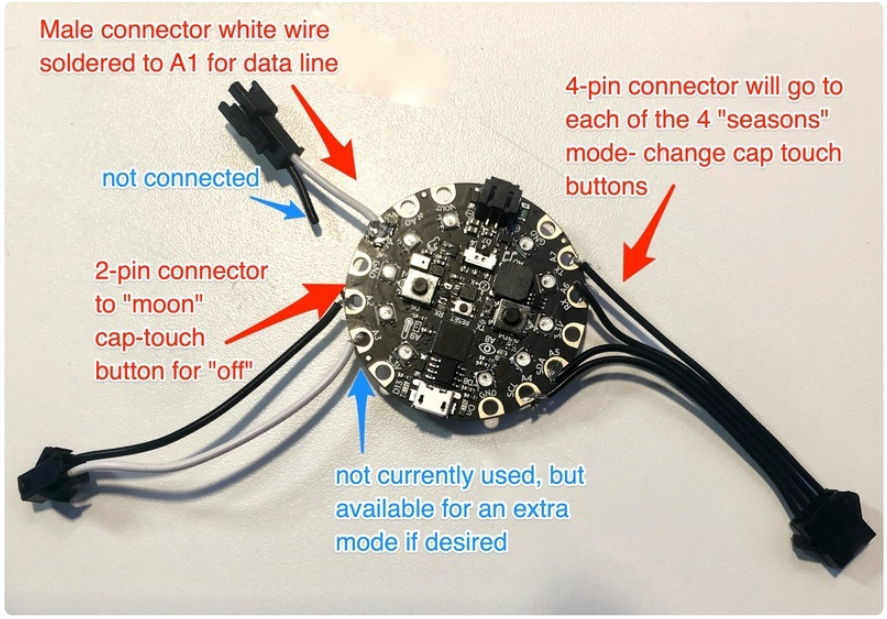
In a couple of places, I added a whole NeoPixel strip along the top of the branches, which shines light up at the ceiling and provides more ambient light for the room. Solving the data-run problem this way really added another beautiful dimension to my final project.

Connectors

Since most of the assembly needed to be done in place instead of on my soldering table, I found it really useful to use JST connectors and splitters to attach each bottle to the power / data grid. If something was flickering, or not lighting up, I could easily swap out the LED bottle for a different bottle. This helped tremendously with troubleshooting.

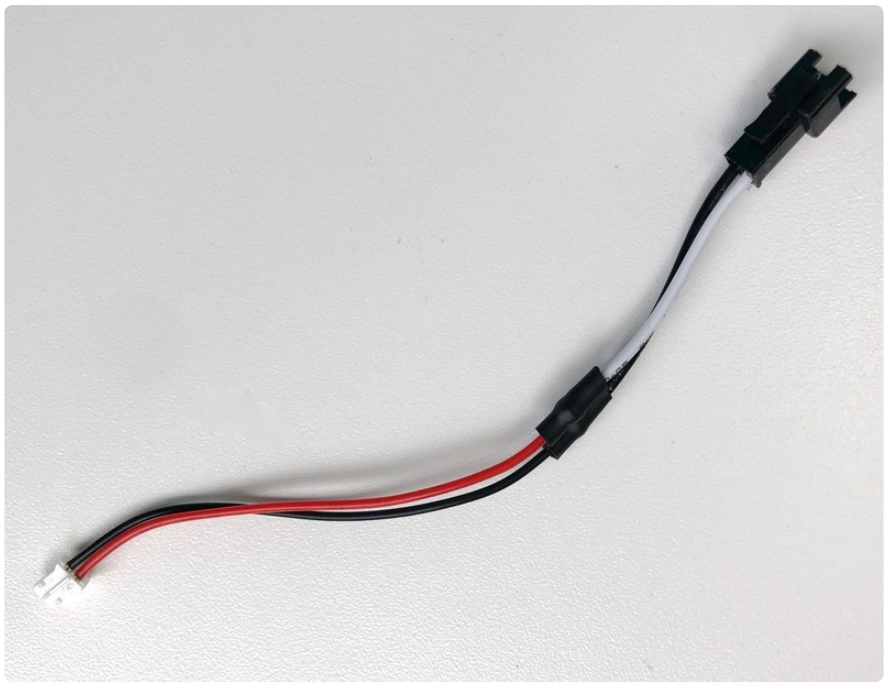
Using JST connectors means you'll be making more than twice as many solder connections, but I really recommend doing it this way - it will make your life much easier in the long run.

Circuit Playground Wiring

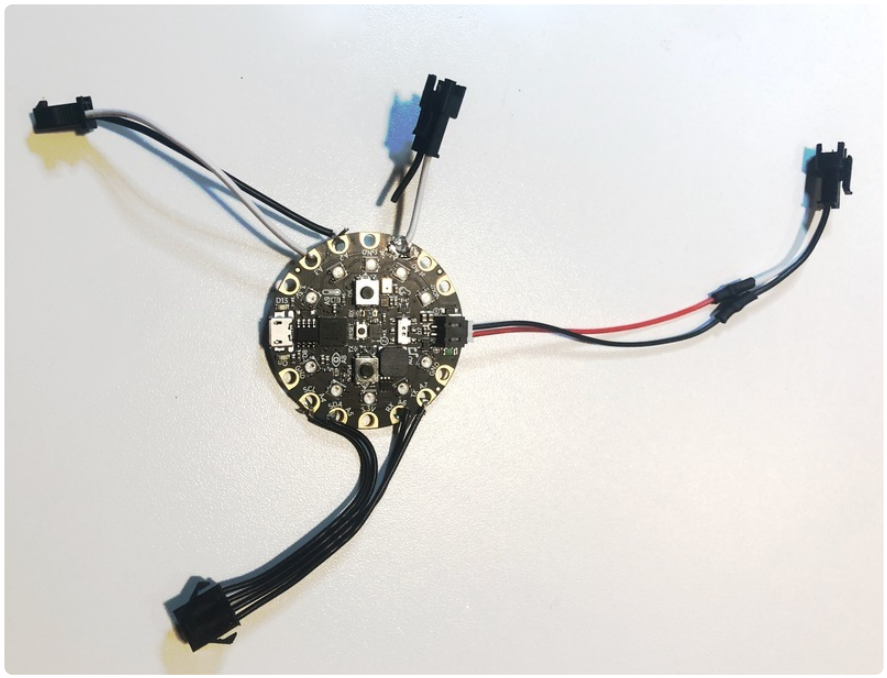


I've got three connectors soldered to my Circuit Playground:

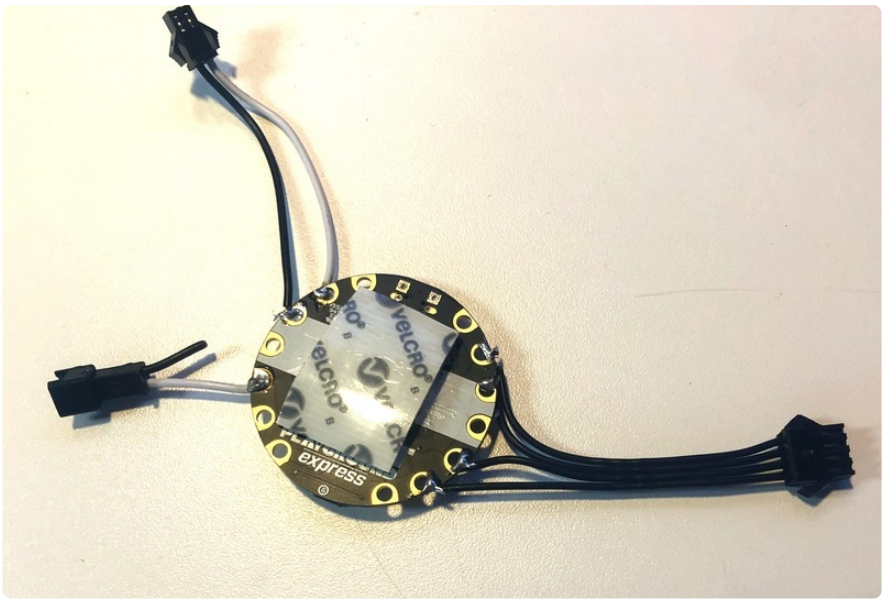
1. A 2-pin male connector with white wire soldered to **A1.** This will connect to the **data IN** pin of the first bottle. The black wire is not used.
2. A 4-pin female connector soldered to **A4, A5, A6** and **A7**. These 4 wires will connect to the 4 metal "season" buttons, for changing modes.
3. A 2-pin female connector soldered to **A2** and **A3**. **A2** will connect to the moon-shaped button and will turn the bottles dark. The **A3** connector is not currently used in my project but could be hooked up to another mode button if desired. (Maybe a secret mode hooked up to a knot hole or mushroom!)



To power the Circuit Playground, we'll use the onboard JST connector. Create an adapter using a female JST battery connector soldered to a male 2-pin JST connector, for easy interfacing to the power system we'll build later on. Here's the whole thing in all its glory.

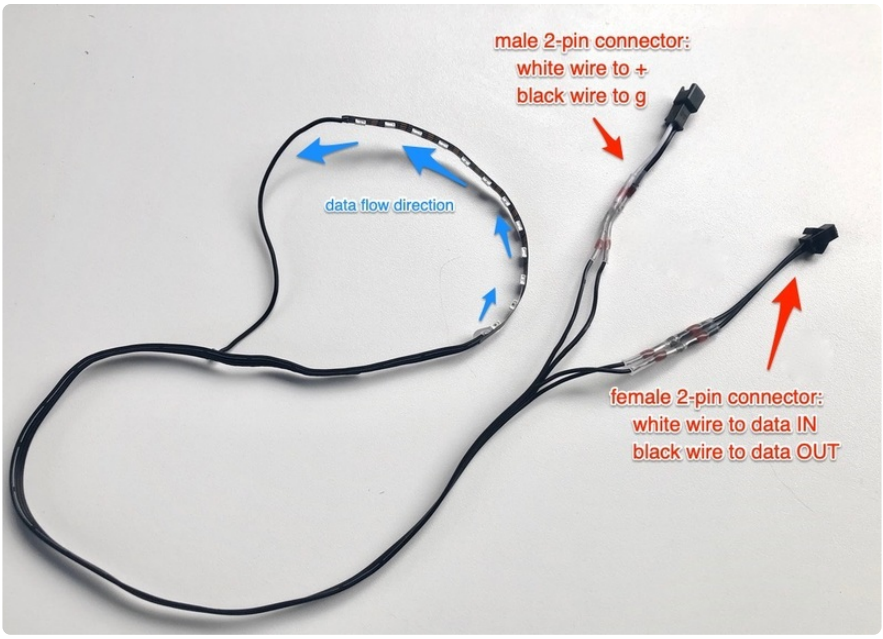


Lastly, I stuck some sticky-back velcro on the back of my Circuit Playground for mounting inside the tree.

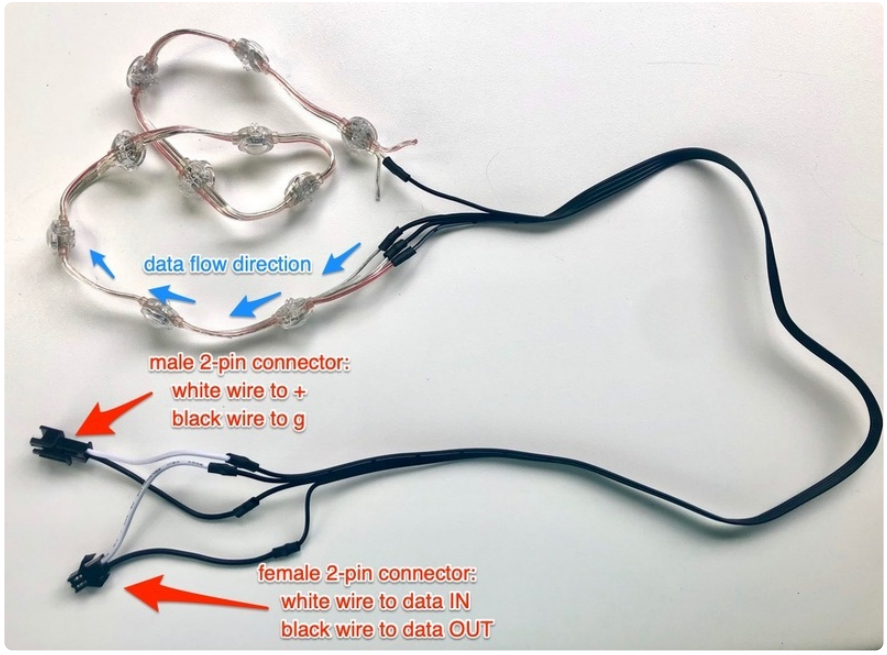


Bottle Lights Wiring

Each bottle has a strip of Skinny NeoPixels or a strand of NeoPixel Dots inside. I used the dots in the larger bottles and the strips in the smaller ones, depending on what would fit easily through the neck of the bottle. I soldered connectors onto each bottle - a male connector for the **power** and **ground** wires and a female for data **IN** and data **OUT.** This way the bottles are interchangeable, so I can move them around and switch them out for troubleshooting or aesthetic reasons.



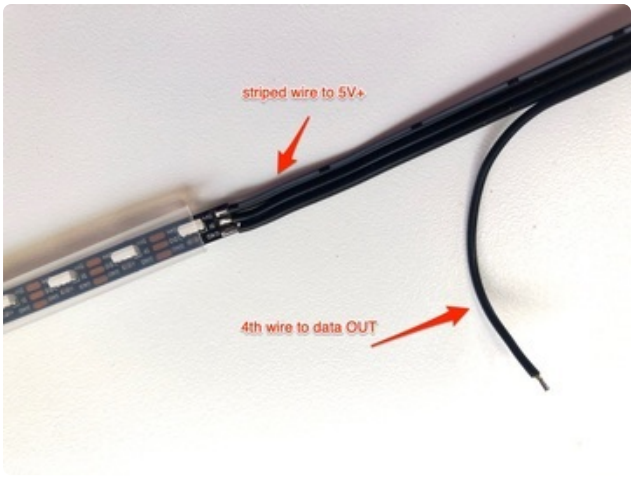
The same exact wiring applies if you're using dots:



Being 100% consistent with wiring all the bottles is essential to keep your sanity for this project, especially as you start adding more bottles. If even one bottle is wired up differently, you could easily fry the whole system, so be consistent!

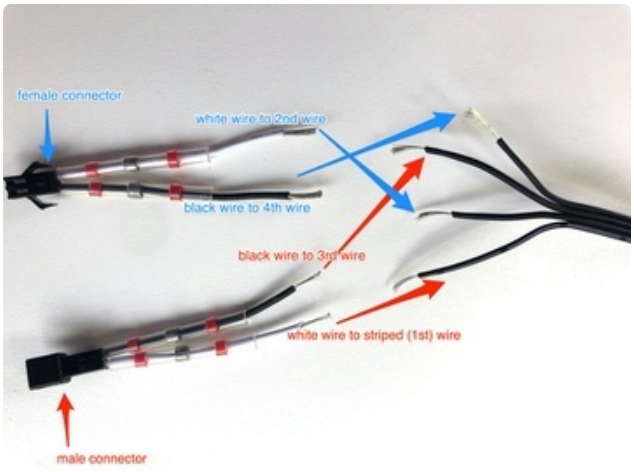
I'll go through a step-by-step process for wiring up the strip. If you're using NeoPixel dots, the connections will be exactly the same but you'll be soldering to wires instead of pads.

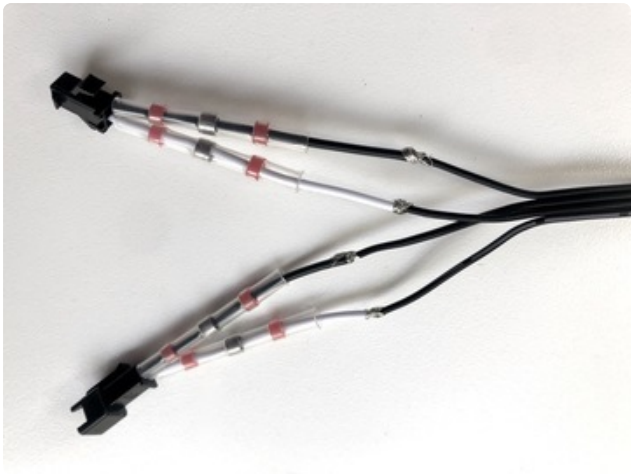
Start by cutting your 1m, 4-strand ribbon cable (http://adafru.it/3892) in half. This is just the right length to hang the bottles close to the tree. If you want lower hanging bottles, adjust the ribbon cable length accordingly -- you want the wire to reach up and over the branch so you can hide the connectors.



Cut your NeoPixel strip to length. I have about 12 pixels in every bottle, give or take. This is a great time to use up those leftover bits from other projects.

Solder one end of your 4-wire ribbon cable to the strip as shown. The fourth wire is soldered to the **data OUT** pad at the OUT end of the strip.

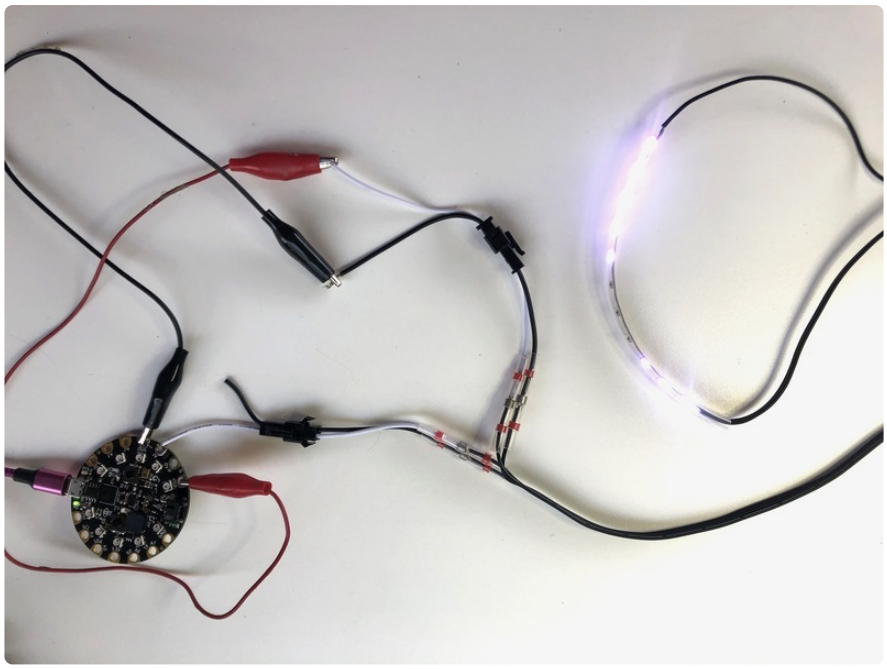




Separate the wires at the other end of your ribbon cable and strip 1/4" of shielding off.

Slip your heat shrink connectors onto a male and female 2-pin JST connector. Connect the four wires as shown. Don't shrink the heat shrink down just yet -- just twist the wires securely together. We'll test before securing.

Note: These connectors were salvaged from the ends of a NeoPixel strip so they're nicely color coded in white and black. You may have connectors that have two black wires, making it a bit harder to tell which is which. If you have that kind, it's a good idea to slip a bit of colored heat shrink, or some colored electrical tape or other marker to one wire so you don't get mixed up and solder anything backwards. It doesn't matter which wire you mark as long as you're consistent on every single connector.



Testing

Test each strip or strand before securing the heat shrink and sealing the ends of the NeoPixels. To test the strip:

1. Load the code from the previous page onto your Circuit Playground
2. Plug the strip's female (data) connector into your Circuit Playground's **A1** connector
3. Plug the strip's male (power) connector into a female connector with exposed / stripped ends
4. Attach alligator clips from the exposed female connector to Circuit Playground's **VBAT** and **G** pads, as shown
5. Make sure none of the un-shielded wire connections are touching each other!
6. Plug your Circuit Playground (with code loaded) into power with a USB cable or battery

If all is working, hooray! You can now slide the heat shrink connectors over your wire joints and use a heat gun to seal the connections. You can also fill the ends of the silicone tubing with hot glue to secure the solder joints to the pads. Now it's time to do the next bottle. Make one for each bottle you plan to hang.

This method does NOT test the **data OUT** connection from each strip, but as long as the solder joint looks good and the strip lights up, you know you're 90% good with all the wires soldered to the right places, and you can pretty much move on.

Troubleshooting

If the strand does not light up, here are a few things to check:

1. Do you have the code correctly loaded onto the Circuit Playground? Try uploading it again and make sure you're calling **A1**. If you're still not sure, try loading a simple rainbow "test" animation.
2. Are any of the unshielded wires touching each other? They can cause a short if they touch.
3. Did you mix up the wires on the connectors? This is very easy to do! If the Circuit Playground crashes or resets itself you may have power and ground switched. If the board seems fine, but there's just no light, you may have soldered to the **OUT** end of the pixels instead of the **IN** end. Power can flow either way but data must flow from **IN** to **OUT**.

Tree Connectors Wiring

Now for the tricky part.

Hang your bottles with the light strands inside. I closed the top of each bottle with a cork, to help provide strain relief to the wires. Use the leather thong to tie the bottle securely to your support structure. Let the thong take the weight of the bottle, so it's not hanging by the wires, which could pull out or break.

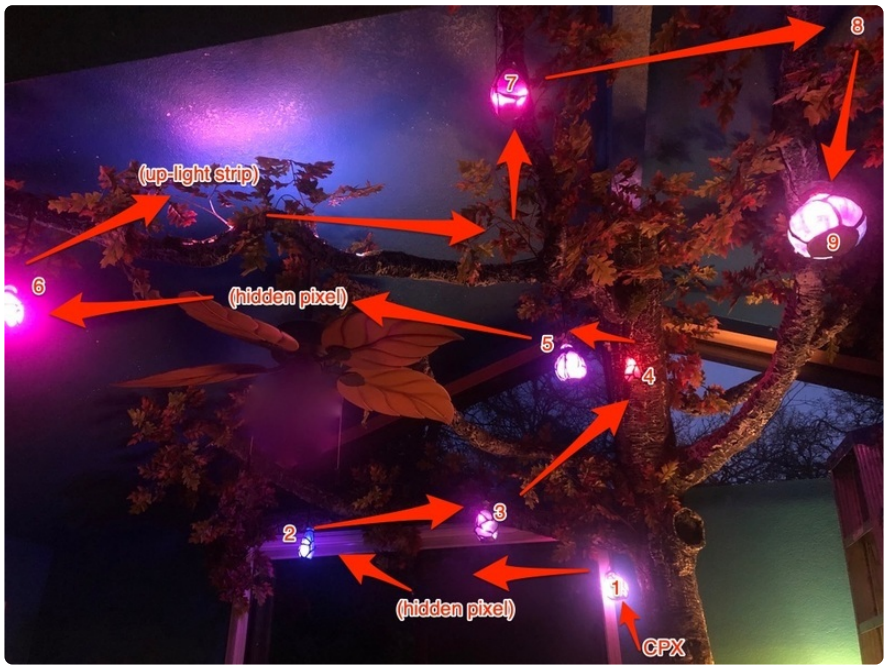
Don't worry yet about wiring or power or anything, just put the bottles where you want them to go. Move them around until they are just right.

Thread the wires from the bottle up and around the thong so the connectors are hidden up above your structure.

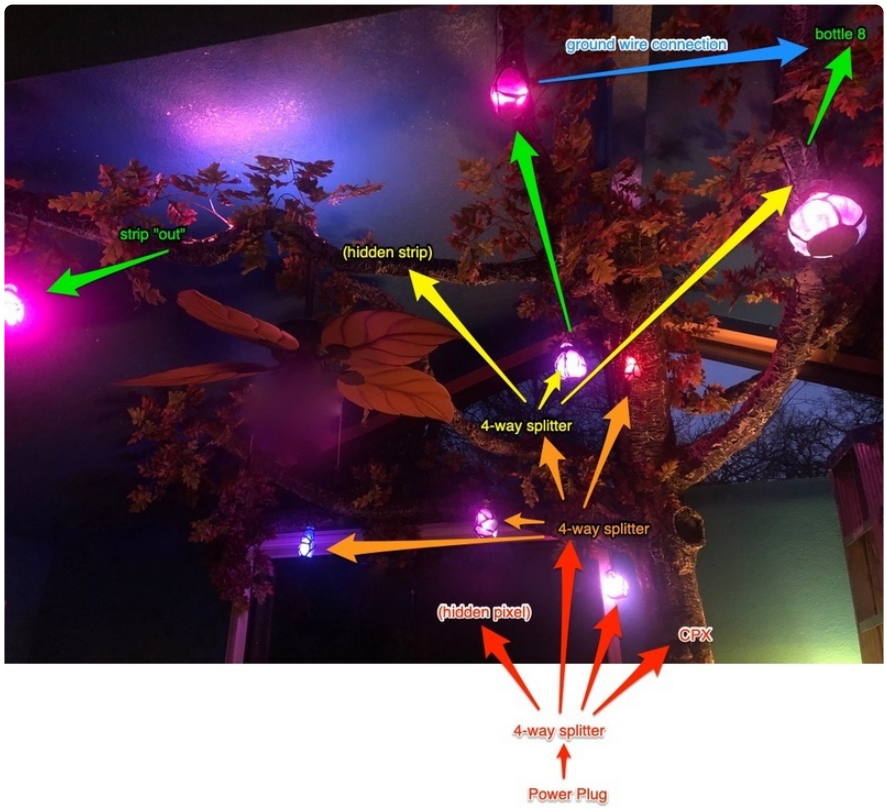
Once you're happy with the placement, sketch out a diagram for data flow. Remember, long wire runs between bottles will degrade your signal, so figure out the most efficient way to get from bottle to bottle without a whole bunch of wire showing.

For extra-long wire runs, where the bottles are spaced far apart, you'll need to add a NeoPixel or bit of NeoPixel strip as a "repeater." Otherwise, your more complex animations may flicker and fail.

Here is a diagram of my data flow. All the wiring is hidden behind branches except the run between bottle #7 and bottle #8, which I hid behind some leaves instead.

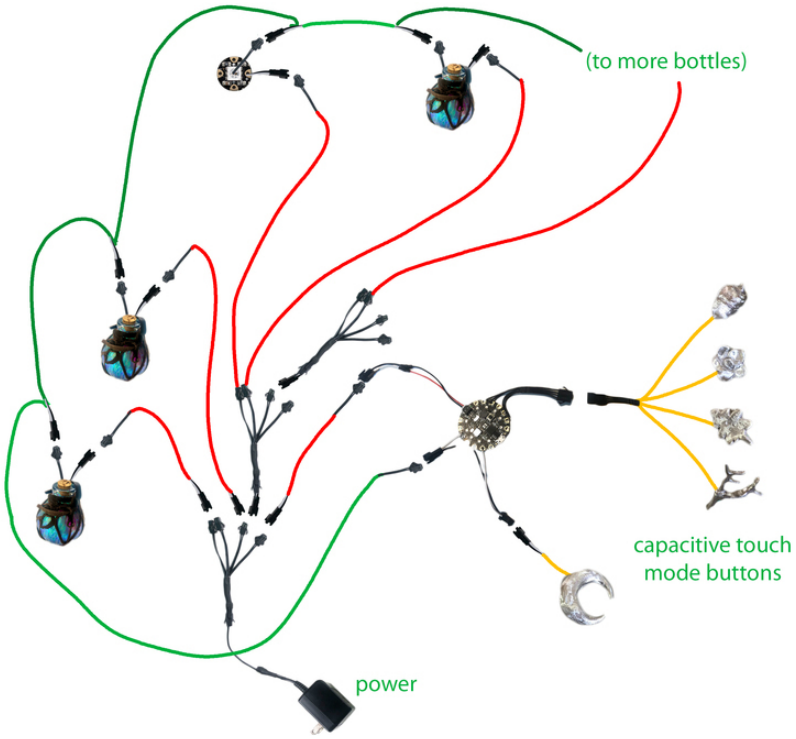


Next, sketch out a diagram of your power flow. I connected each bottle to power directly instead of following the data flow, in order to avoid brown-outs near the end of the strip. If you have just a few bottles with only a few lights, this may not be necessary -- you could just run ribbon cable between each bottle. But I've got around 200 LEDs in total so wanted to be sure everything was adequately powered, even for bright-white "lightning flash" animations.



This graphic shows my power and ground wires. Most bottles connect directly to power (or a power splitter). A few of the further out ones are chained from other bottles, but I don't have more than two bottles in any chain.

The other thing to note is the ground wire connection in the upper right corner. With the power flowing along different branches, I was getting a pretty serious flicker in the bottles all the way to the right. Adding a ground wire connection between the two furthest bottles got rid of the flicker.



Connecting Wires

It's easiest to test the system if you build all the connections for each bottle before moving on to the next bottle, rather than doing the whole power system and then the whole data system.

Power Connector Wires

Start at the wall with your power supply and work outwards from there.

I cut off the jack on my power supply, and soldered on a female connector. The red (power) wire in the power supply connects to the white wire on the connector (or if yours aren't color coded, the one on the left, as you look at the splitter from the top).

Power will ALWAYS be on the left for a female connector, or on the right (still the white wire) for a male connector. Write this down somewhere, or maybe tattoo it on your forehead. Get one single connector wrong and the whole system will short out.

In order to connect to the splitter, you need a **male** connector. Your bottles are expecting a **female** connector (remember, you used a **male** for the two power wires for every bottle, right?). So, all extension wires for your power system will go from **male** on one end to **female** on the other.

Adafruit sells pre-made extension cables that are 1m long (http://adafru.it/616) with a male-to-female connector! So if your bottles are 1 meter (or less) apart, you can save yourself some work by ordering a bunch of these. You could also chain them together for longer power runs. Or, you can be an overachiever like me and solder up your own cables that are the perfect length, to make wire-hiding easier later on.

Don't forget that you'll need power to your NeoPixel repeaters as well.

Data Connector Wires

I am using a male connector on the Circuit Playground's **A1** pin, so my extension wire will need a female connector. The white wire on the connector will always be data **IN**, and the black will always be data **OUT**, to match up with the connectors we soldered to the bottles.

OUT from one bottle becomes **IN** to the next bottle in the series, and so forth and so on.

The bottles' data lines are soldered to a **female** connector, so they are expecting a **male**. So the first extension wire will be **female** to **male**.

Every other extension wire for data will be **male** to **male**, with the data flowing **IN** one connector wire and **OUT** the other. Adafruit doesn't sell **male** to **male** extension cables so you'll need to solder up your own.

If you're finding all this complicated, well, I warned you.

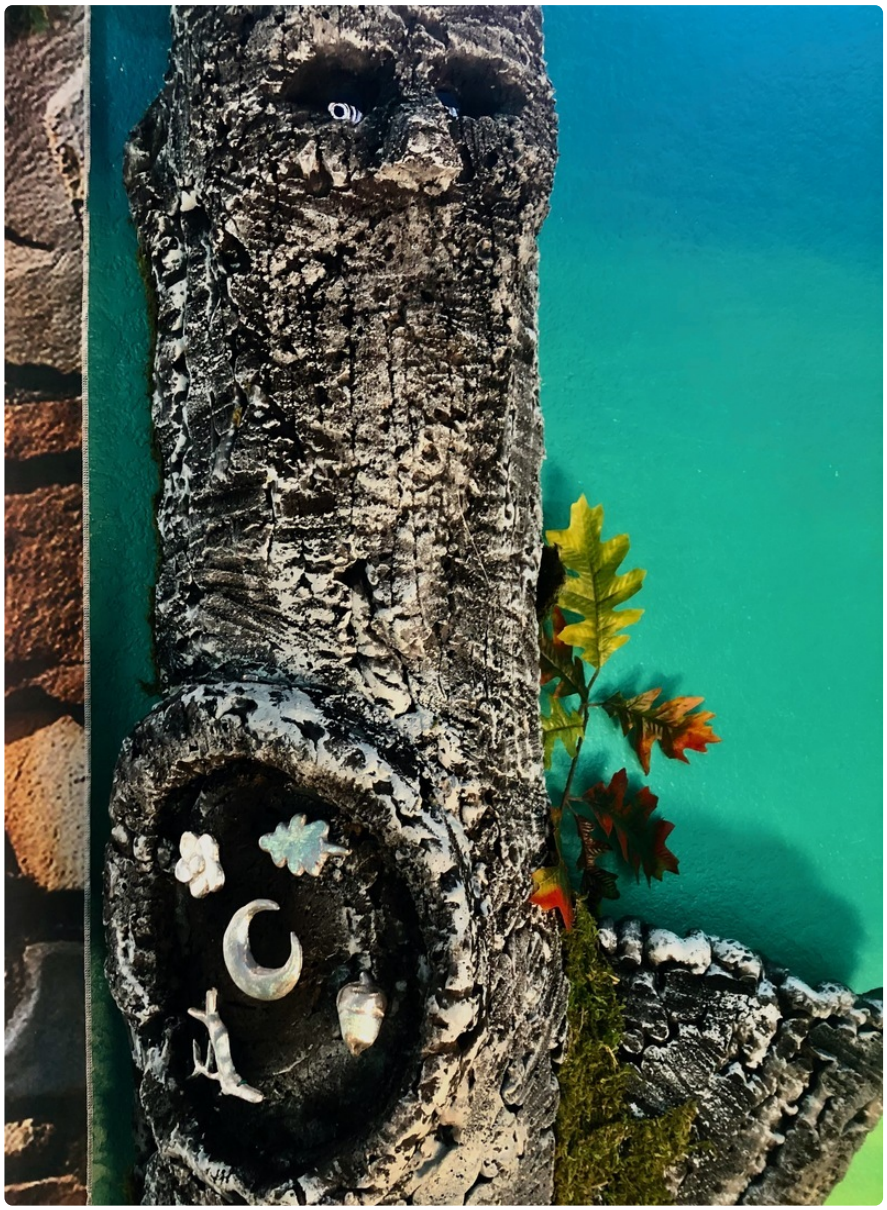
Putting it All Together

Work in sequence:

1. Wall power to splitter
2. Power to Circuit Playground
3. Data & Power to bottle #1
4. Data & Power to bottle #2
5. Data & Power to repeater (if needed)
6. Data & Power to bottle #3
7. Etc, until all bottles are finished

Order more connectors than you think you need! And have plenty of wire and wire connectors on hand.

Capacitive Touch Controller



Our code has four different modes, one for each of the four seasons, plus a mode for "off". We've hooked up five capacitive touch pads on the Circuit Playground Express to trigger the different modes. We can leave it like this and mount the Circuit Playground on the outside of the tree, or we can get a little fancy and make a custom controller using metal buttons or copper tape, or really anything that conducts electricity. Check out this fruit sequencer (https://adafru.it/DhR) for inspiration!

Craft stores sell a lot of charms or jewelry findings that will often work well. Use a file to scrape off any top coating so you're touching bare metal, and solder a wire to the back of the charm.

I went one step further and made cast pewter "icons," embedding the end of a wire in the molten pewter before it set up. This connection is really solid and steady, so I really don't get any misfires and there's no chance of the solder connection failing.

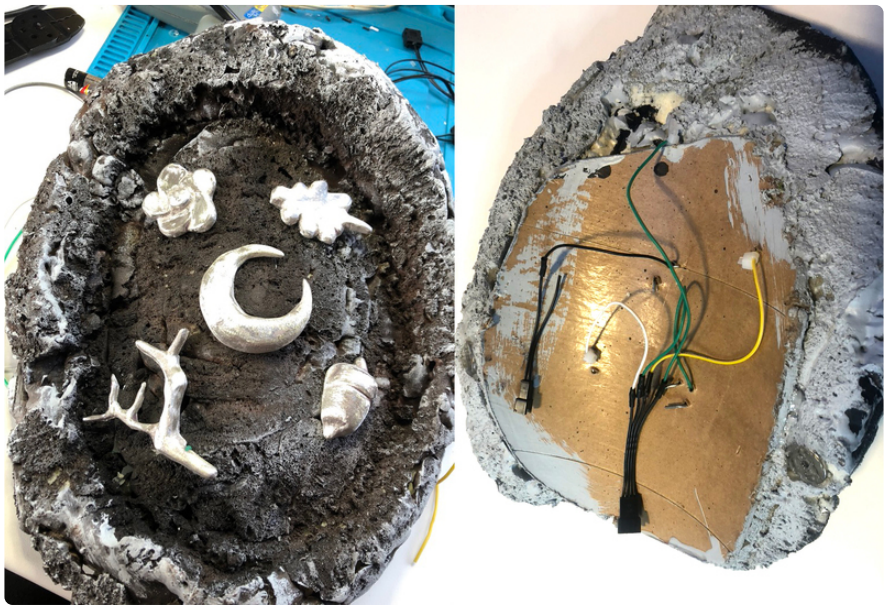
The details are beyond the scope of this (already complicated enough) guide, but I'll go over the basics because it was a fun bit of silliness to create.

First, I made some little icons out of polymer clay: a flower blossom for spring, a leaf for summer, an acorn for autumn, and a moon for "off".

I pressed some kinetic sand in a metal tray and made an impression of each icon. Then I melted a pewter ingot in my crucible and poured some pewter into the impression. While the pewter was still molten, I inserted a wire (with the shielding removed) to connect to the Circuit Playground, and also a u-shaped bit of galvanized wire to use as an attachment point to the tree.

The kinetic sand is not great at capturing fine details, but since my icons were simple shapes that didn't matter too much. I polished them up with the sanding bit on my Dremel, and they came out shiny and lovely!

Some of them took a couple tries to get right. The nice thing about pewter is that it's pretty waste-free -- if you don't get it right, you can just melt it down and try again.



I have a removable access hatch on my tree, so I mounted the buttons there. I drilled a small hole for the wire to poke through, and used the galvanized wire points and some E6000 glue to help secure the buttons to the bark.

I attached the four seasons buttons to a male 4-pin connector, and the moon pin to a 2-pin connector, for easy attaching to the Circuit Playground Express. Now I can change modes just by brushing my fingertips over one of the four season icons.

Sleeping under the tree is such a delight. This project was a lot of work, but the results are simply stunning and 100% worth the effort. There's a fairyland in my guest room. Who wants to come visit?