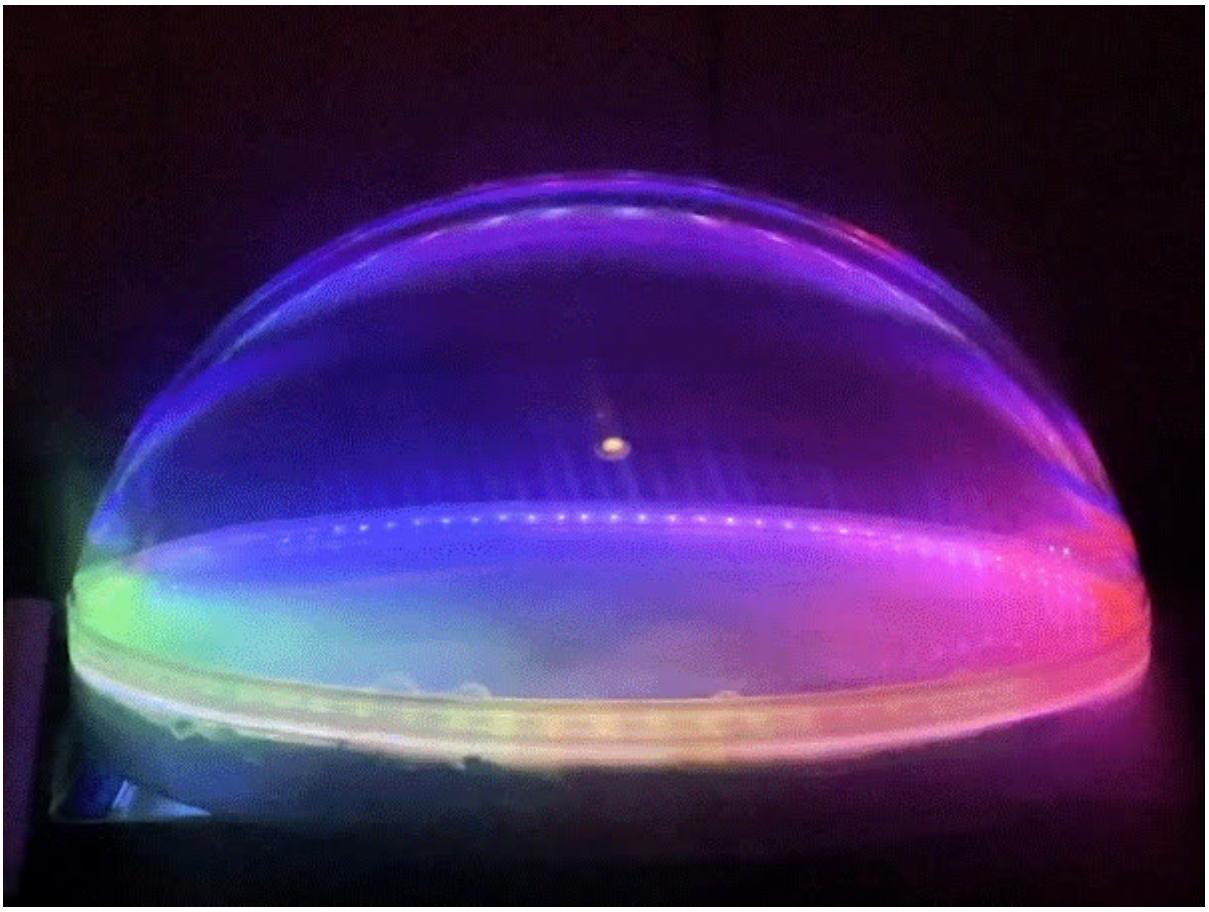
Bubble Table with LED Animations and IR Remote Control

Created by Erin St Blaine



https://learn.adafruit.com/bubble-table-with-led-animations-and-ir-remote-control

Last updated on 2024-06-03 03:47:22 PM EDT

Table of Contents

Overview 3

- Parts
- Additional Materials
- Tools

Wiring Diagram 7

- Battery Charging

Install CircuitPython 8

- CircuitPython Quickstart
- Safe Mode
- Flash Resetting UF2

Software 12

- Upload the Code and Libraries to the Feather RP2040
- How the CircuitPython Code Works
- NeoPixel Setup
- Speed & Brightness Control
- Animations
- IR Mapping
- Solid Color Keys
- Main Loop

Electronics Assembly 20

- Testing
- Troubleshooting

Table Build 24

- Mirror Film
- Mirror Effect Spray Paint
- Final Assembly

Bubbles 32

- Bubble Recipe
- Fog-filled Bubbles

Overview

Bubbles capture the human imagination like nothing else. Ephemeral and magical, their rainbow sheen will delight and amaze anyone who gazes upon them. The magic of bubbles stems largely from their impermanence.

Watching a bubble float through the air can bring the mind's attention fully into the present moment. Blink, and the bubble pops. The moment is over. Were you paying attention?

Enhance the beauty of your bubble time with this LED bubble table. We've made a waterproof, lipped tray for bubble solution with infinity mirror style diffusion underneath. Sculpt a masterpiece out of stacked and smoke-filled bubbles. Delight your audience and let your mind be at peace, enjoying the impermanence of your creations.

Add as many light modes as you want using CircuitPython's easy-to-use LED Animations Library. Control the modes and brightness with an infrared (IR) remote control. The Feather RP2040 also includes battery charging via the onboard USB port. Use the table plugged in or use it onstage with a hefty 2500mAh battery.

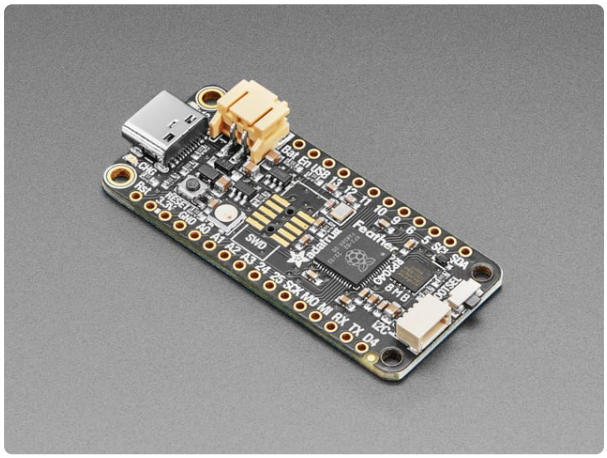
We've also included our favorite bubble solution recipe, which is something every mad scientist should have in their arsenal. Although, we love staring at this thing even when it's not covered in bubbles.

Difficulty

This is an intermediate project. You'll need to know your way around a soldering iron. I used some wood shop tools as well -- a table saw and a band saw -- but you could order the plastic pieces pre-cut for just a little more money if you don't have those available.

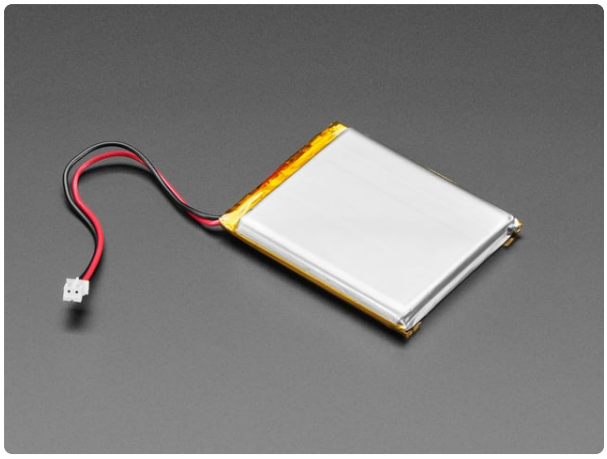
The code is ready to use as-is with lots of fun animations, and is pretty easy to customize. The CircuitPython LED Animations library is designed to be straightforward and intuitive to use. This is a great project for learning to create your own LED animations.

Parts



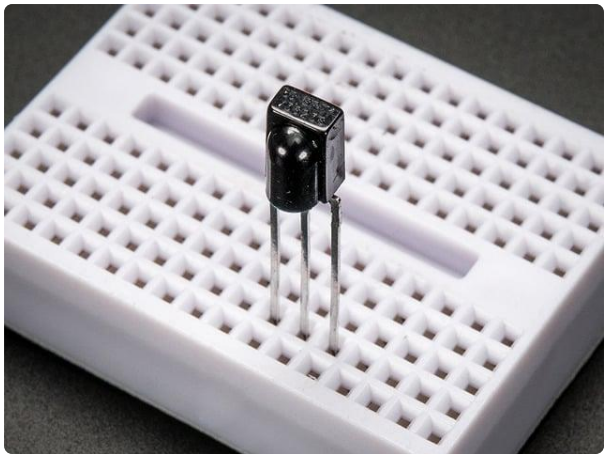
Adafruit Feather RP2040
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...
https://www.adafruit.com/product/4884

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/2959



[Lithium Ion Polymer Battery - 3.7v 2500mAh]
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/328

[JST 2-pin Extension Cable with On/Off Switch - JST PH2]
By popular request - we now have a way you can turn on-and-off Lithium Polymer batteries without unplugging them.This PH2 Female/Male JST 2-pin Extension...
https://www.adafruit.com/product/3064



[IR (Infrared) Receiver Sensor]
IR sensor tuned to 38KHz, perfect for receiving commands from a TV remote control. Runs at 3V to 5V so it's great for any microcontroller.To use, connect pin 3 (all the...
https://www.adafruit.com/product/157

[Mini Remote Control]
This little remote control would be handy for controlling a robot or other project from across the room. It has 21 buttons and a layout we thought was handy: directional buttons and...
https://www.adafruit.com/product/389

USB C Cable for Programming

1 x USB C Cable — https://www.adafruit.com/product/4474

USB C Cable for Programming

---

1 x Clear Heat Shrink — https://www.adafruit.com/product/1020

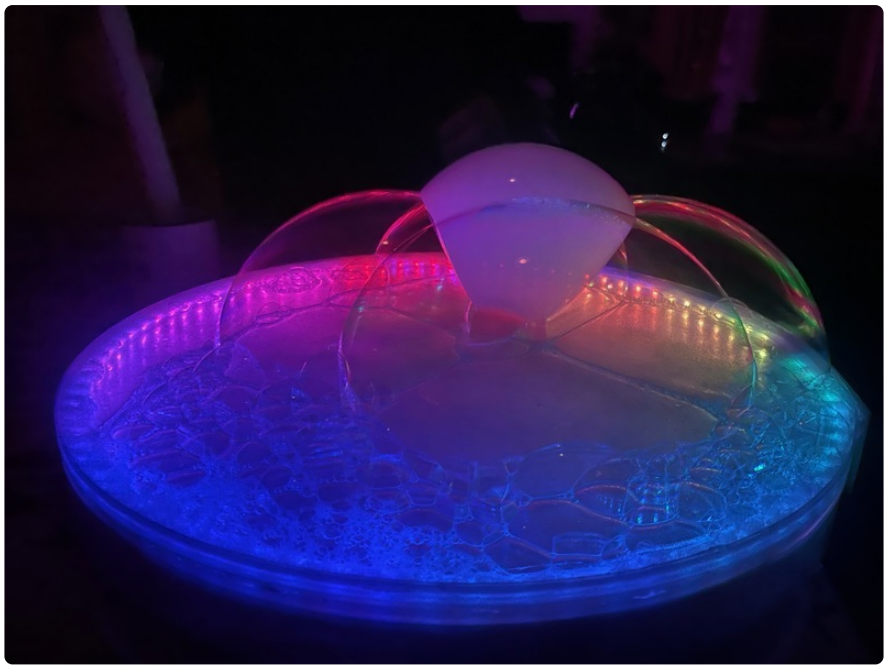
3/4" Clear Heat Shrink Tubing

Additional Materials

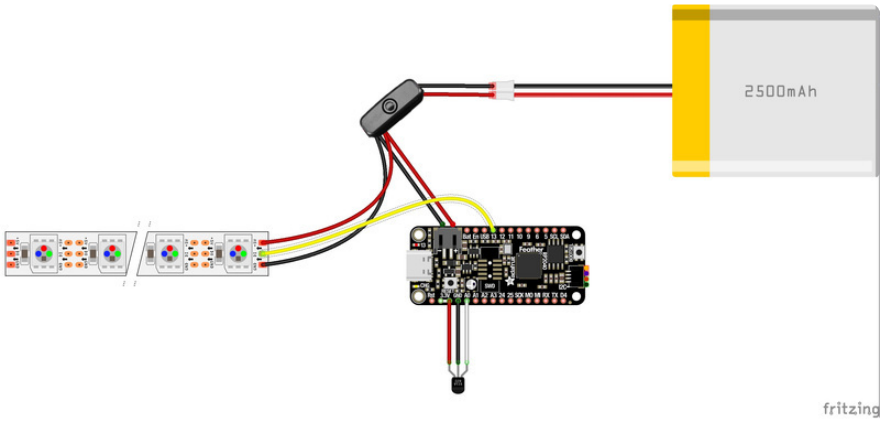
- 2 acrylic or polycarbonate 18" x 1/8" circles: available at Tap Plastics (https://adafru.it/18zE) or online, or cut your own from a sheet of plastic with a band saw
- 3" x 60" x 1/8" strip of acrylic or polycarbonate for the table edge. If you're making a bigger table, make the length the circumference of your circle plus a few inches for overlap.
- Clear silicone caulk (https://adafru.it/18A0) for waterproofing
- E6000 glue (https://adafru.it/18A1)
- Mirror Film
- Mirror Effect Spray Paint (https://adafru.it/18A2) (more diffused) or 2-way mirror film (https://adafru.it/18A1) (more reflective for an infinity mirror effect)

Tools

- Soldering iron & accessories
- Table saw
- Band saw (optional)
- Hot Glue Gun
- Heat Gun



Wiring Diagram



The NeoPixel strip is attached as follows:

- **NeoPixel DI to Feather pin 13**
- **+5v to red wire spliced to on/off switch**
- **G to black wire spliced to on/off switch**

Remember, NeoPixel strips are directional. Be sure you're soldering to the "in" end of the strip **(DI)** and not the "out" end (**DO**).

The IR Sensor is attached as follows. With the sensor facing up and the legs down, and the bump facing you:

- **Out** (leftmost pin) to Feather pin **A0**

- **G** (middle pin) to **G**
- **Power** (rightmost pin) to **3V**

Plug the battery into the on/off switch and the switch into the board.

Battery Charging

This project is designed to run from an onboard battery. The Feather microcontroller has onboard battery charging, so you can permanently install your battery and recharge it by plugging in a USB cable to the Feather.

However, with this wiring setup, the battery will not charge with the on/off switch turned off, since the switch is between the battery and the controller. So when you plug in your table to charge, be sure the switch is clicked to "on".

Install CircuitPython

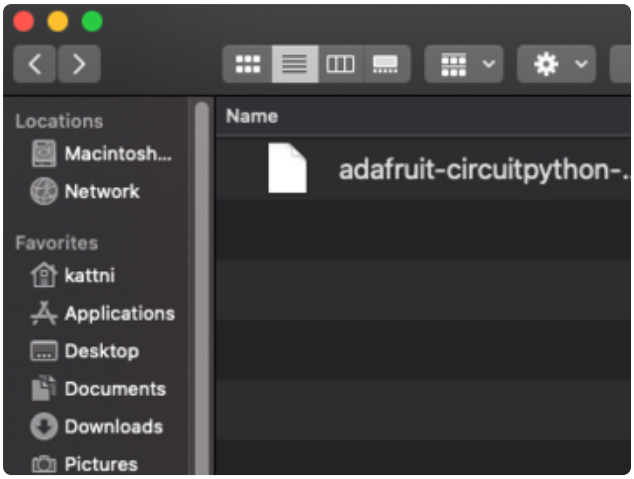
CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

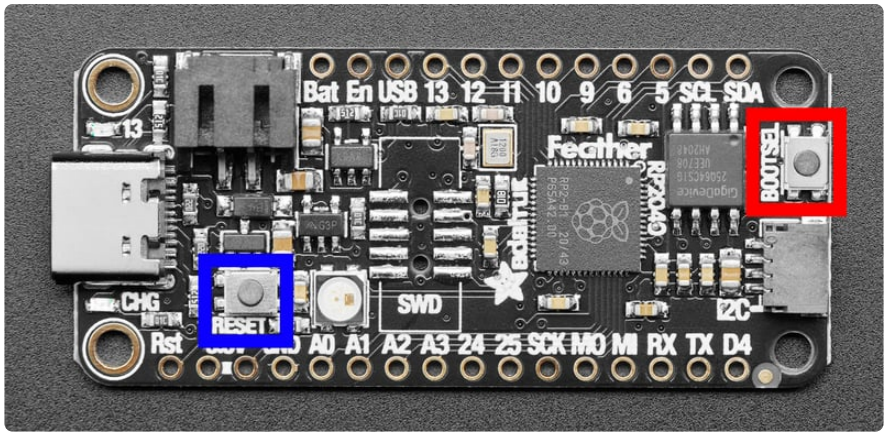
Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/R1D



Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

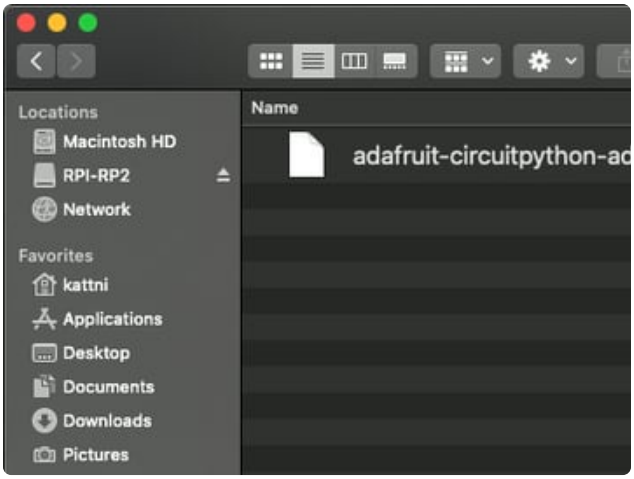


To enter the bootloader, hold down the **BOOT/BOOTSEL button** (highlighted in red above), and while continuing to hold it (don't let go!), press and release the **reset button** (highlighted in blue above). **Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!**

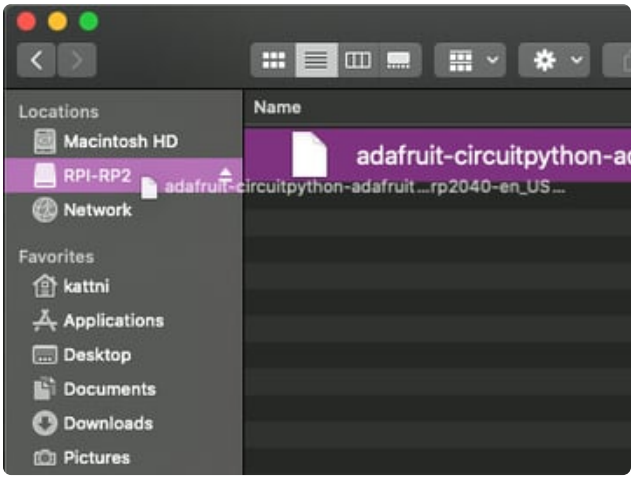
If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

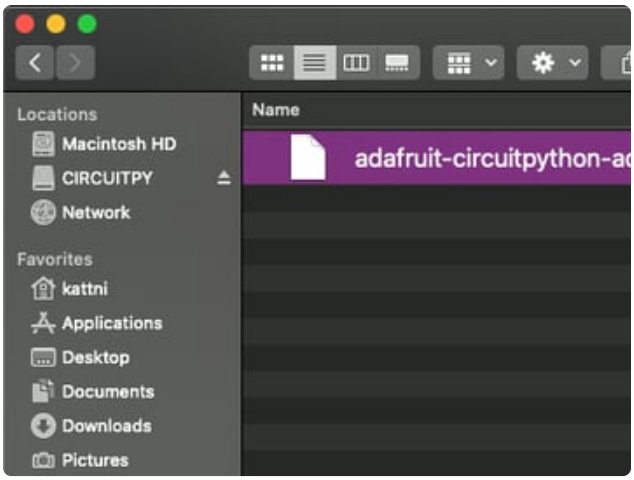
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



You will see a new disk drive appear called **RPI-RP2**.



Drag the **adafruit_circuitpython_etc.uf2** file to **RPI-RP2.**



The **RPI-RP2** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Safe Mode

You want to edit your **code.py** or modify the files on your **CIRCUITPY** drive, but find that you can't. Perhaps your board has gotten into a state where **CIRCUITPY** is read-only. You may have turned off the **CIRCUITPY** drive altogether. Whatever the reason, safe mode can help.

Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in **boot.py** (where you can set **CIRCUITPY** read-only or turn it off completely). Second, it does not run the code in **code.py**. And finally, it does not automatically soft-reload when data is written to the **CIRCUITPY** drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the **CIRCUITPY** drive.

Entering Safe Mode

To enter safe mode when using CircuitPython, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

If you successfully enter safe mode on CircuitPython, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.

CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.

Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the **CIRCUITPY** drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and CIRCUITPY doesn't show up as a disk drive after installing CircuitPython, try loading this 'nuke' UF2 to RPI-RP2. which

will do a 'deep clean' on your Flash Memory. **You will lose all the files on the board**, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

https://adafru.it/RLE

Software

Once you've finished setting up your Feather RP2040 with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download to your computer as a zipped folder.

```
# SPDX-FileCopyrightText: 2017 John Park for Adafruit Industries
# Modified 2023 by Erin St Blaine
#
# SPDX-License-Identifier: MIT


import board
import pulseio
import neopixel
import adafruit_irremote
from rainbowio import colorwheel
from adafruit_led_animation.sequence import AnimationSequence
from adafruit_led_animation.animation.solid import Solid
from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.sparkle import Sparkle
from adafruit_led_animation.animation.rainbowchase import RainbowChase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.animation.SparklePulse import SparklePulse
import adafruit_led_animation.color as color

NUMBER_OF_PIXELS = 85
pixels = neopixel.NeoPixel(board.D13, NUMBER_OF_PIXELS)

# Define the brightness levels and their corresponding values
# Start at a non-blinding brightness.
BRIGHTNESS_LEVELS = (0.025, 0.05, 0.1, 0.2, 0.4, 0.6, 0.7, 0.8, 1.0)
brightness_index = 2
pixels.brightness = BRIGHTNESS_LEVELS[brightness_index]

pulsein = pulseio.PulseIn(board.A0, maxlen=120, idle_state=True)
decoder = adafruit_irremote.GenericDecode()

SPEEDS = (0.25, 0.125, 0.1, 0.08, 0.05, 0.02, 0.01)  # Customize speed levels here
speed_index = 4

def setup_animations():
    """Set up all the available animations."""
    # Animation Setup
    rainbow = Rainbow(pixels, speed=SPEEDS[speed_index], period=2, name="rainbow",
step=3)
```

```
    sparkle = Sparkle(pixels, speed=SPEEDS[speed_index], color=color.WHITE,
name="sparkle")
    solid = Solid(pixels, color=colorwheel(0), name="solid")
    # Make the Solid animation changeable quickly.
    solid.speed = 0.01
    off = Solid(pixels, color=color.BLACK, name="off")
    rainbow = Rainbow(pixels, speed=SPEEDS[speed_index], period=6, name="rainbow",
step=2.4)
    rainbow_carousel = RainbowChase(pixels, speed=SPEEDS[speed_index], size=4,
spacing=1, step=20)
    party_chase = RainbowChase(pixels, speed=SPEEDS[speed_index], size=1, spacing=5,
step=6)
    rainbow_chase2 = RainbowChase(pixels, speed=SPEEDS[speed_index], size=10,
spacing=1, step=18)
    chase = Chase(pixels, speed=SPEEDS[speed_index], color=color.RED, size=1,
spacing=6)
    rainbow_comet2 = RainbowComet(
        pixels, speed=0.02, tail_length=104, colorwheel_offset=80, bounce=False)
    rainbow_comet3 = RainbowComet(
        pixels, speed=SPEEDS[speed_index], tail_length=25,
        colorwheel_offset=128, step=4, bounce=False)
    lava = Comet(pixels, speed=SPEEDS[speed_index],
                 color=color.ORANGE, tail_length=40, bounce=False)
    sparkle1 = Sparkle(pixels, speed=SPEEDS[speed_index], color=color.BLUE,
num_sparkles=10)
    pulse = Pulse(pixels, speed=0.1, color=color.AMBER, period=3)
    sparkle_pulse = SparklePulse(pixels, speed=0.05, period=2, color=color.JADE,
max_intensity=3)

    # Animation Sequence Playlist -- rearrange to change the order of animations
    # advance_interval is None, so the animations change only under user control.
    all_animations = AnimationSequence(
        rainbow,
        rainbow_chase2,
        rainbow_carousel,
        party_chase,
        rainbow_comet2,
        rainbow_comet3,
        sparkle_pulse,
        pulse,
        chase,
        rainbow,
        solid,
        sparkle,
        lava,
        sparkle1,
        off,
        auto_clear=True,
        auto_reset=True,
        advance_interval=None,
    )
    return all_animations

# IR Remote Mapping for the Adafruit mini IR remote
# https://www.adafruit.com/product/389

CMD_1 = 247          #  1: [255, 2, 247, 8]
CMD_2 = 119          #  2: [255, 2, 119, 136]
CMD_3 = 183          #  3: [255, 2, 183, 72]
CMD_4 = 215          #  4: [255, 2, 215, 40]
CMD_5 = 87           #  5: [255, 2, 87, 168]
CMD_6 = 151          #  6: [255, 2, 151, 104]
CMD_7 = 231          #  7: [255, 2, 231, 24]
CMD_8 = 103          #  8: [255, 2, 103, 152]
CMD_9 = 167          #  9: [255, 2, 167, 88]
CMD_0 = 207          #  0: [255, 2, 207, 48]

CMD_UP = 95          # ^ : [255, 2, 95, 160]
CMD_DOWN = 79        # v : [255, 2, 79, 176]
```

```
CMD_RIGHT = 175      # > : [255, 2, 175, 80]
CMD_LEFT = 239       # < : [255, 2, 239, 16]

CMD_ENTER_SAVE = 111 # Enter/Save: [255, 2, 111, 144]
CMD_SETUP = 223      # Setup: [255, 2, 223, 32]
CMD_STOP_MODE = 159  # Stop/Mode: [255, 2, 159, 96]
CMD_BACK = 143       # Back: [255, 2, 143, 112]

CMD_VOL_DOWN = 255   # Vol - : [255, 2, 255, 0]
CMD_VOL_UP = 191     # Vol + : [255, 2, 191, 64]
CMD_PLAY_PAUSE = 127 # Play/Pause: [255, 2, 127, 128]
CMD_REPEAT = True    # short code: repeat of previous command


def read_command():
    """Try to read an IR command. If none seen or if error, return None."""
    try:
        pulses = decoder.read_pulses(pulsein, blocking=False)
        if pulses:
            code = decoder.decode_bits(pulses)
            if len(code) > 3:
                print("Decoded:", code)
                return code[2]
        # if code is less than or equal to 3 characters long or no pulses received
        return None
    except adafruit_irremote.IRNECRepeatException:  # unusual short code!
        print("NEC repeat!")
        return CMD_REPEAT
    except adafruit_irremote.IRDecodeException as e:  # failed to decode
        print("Failed to decode:", e)
        return None
    except MemoryError as e:
        print("Memory error: ", e)
        return None

SOLID_COLORS = {
    CMD_0 : color.BLACK,
    CMD_1 : color.RED,
    CMD_2 : color.GREEN,
    CMD_3 : color.WHITE,
    CMD_4 : color.BLUE,
    CMD_5 : color.PINK,
    CMD_6 : color.YELLOW,
    CMD_7 : color.PURPLE,
    CMD_8 : color.TEAL,
    CMD_9 : color.ORANGE,
    }

# main program

animations = setup_animations()
last_command = None

while True:
    command = read_command()
    if command is None:
        # Nothing read, just keep animating.
        animations.animate() # Run one animation cycle.
        continue

    if command == CMD_REPEAT:
        command = last_command

    last_command = command
    print("Command", command)


    # See if the command was a number button. Fetch the animation color if it is.
    solid_color = SOLID_COLORS.get(command, None)
```

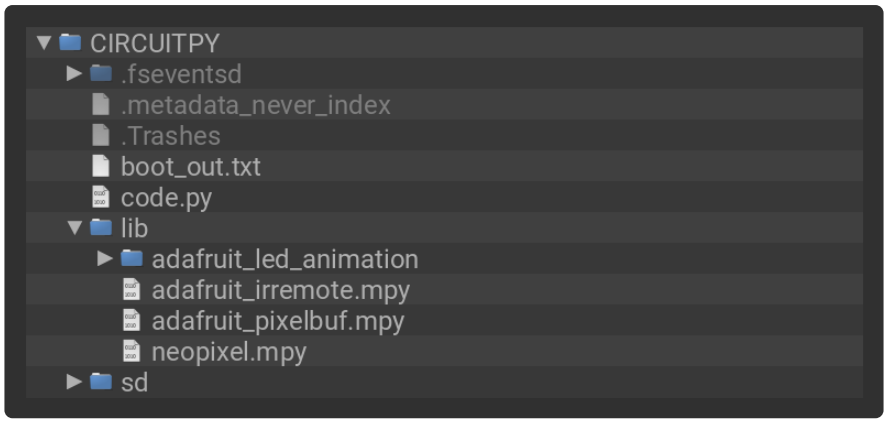
```
    if solid_color:
        # Jump to the "solid" animation. Set its color to
        # the chosen color.
        animations.activate("solid")
        animations.current_animation.color = solid_color
    elif command == CMD_LEFT:
        animations.previous()
    elif command == CMD_RIGHT:
        animations.next()
    elif command == CMD_DOWN:
        # Slow down current animation
        if speed_index > 0:
            speed_index -= 1
            animations.current_animation.speed = SPEEDS[speed_index]
        print("speed of current animation is now:",
animations.current_animation.speed)
    elif command == CMD_UP:
        if speed_index < len(SPEEDS) - 1:
            speed_index += 1
            animations.current_animation.speed = SPEEDS[speed_index]
        print("speed of current animation is now:",
animations.current_animation.speed)
    elif command == CMD_VOL_DOWN:
        if brightness_index > 0:
            brightness_index -= 1
            pixels.brightness = BRIGHTNESS_LEVELS[brightness_index]
        print("brightness:", pixels.brightness)
    elif command == CMD_VOL_UP:
        if brightness_index < len(BRIGHTNESS_LEVELS) - 1:
            brightness_index += 1
            pixels.brightness = BRIGHTNESS_LEVELS[brightness_index]
            print("brightness:", pixels.brightness)
```

Upload the Code and Libraries to the Feather RP2040

After downloading the Project Bundle, plug your Feather RP2040 into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Feather RP2040's **CIRCUITPY** drive.

- **lib** folder
- **code.py**

Your Feather RP2040 **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file.



How the CircuitPython Code Works

First, we import all the necessary libraries and animations we'll be using.

Check out the CircuitPython LED Animations Library guide (https://adafru.it/LAg) for more info on adding your own animations.

```
import board
import pulseio
import neopixel
import adafruit_irremote
from rainbowio import colorwheel
from adafruit_led_animation.sequence import AnimationSequence
from adafruit_led_animation.animation.solid import Solid
from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.sparkle import Sparkle
from adafruit_led_animation.animation.rainbowchase import RainbowChase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.animation.SparklePulse import SparklePulse
import adafruit_led_animation.color as color
```

NeoPixel Setup

Set your total number of pixels here. If you soldered to a different pin than 13, you can change the pin number here as well.

```
NUMBER_OF_PIXELS = 85
pixels = neopixel.NeoPixel(board.D13, NUMBER_OF_PIXELS)
```

Speed & Brightness Control

This code includes 8 brightness levels and 7 speed levels, which can be selected with buttons on the IR remote. You can edit them, add or subtract here.

```
# Define the brightness levels and their corresponding values
# Start at a non-blinding brightness.
BRIGHTNESS_LEVELS = (0.025, 0.05, 0.1, 0.2, 0.4, 0.6, 0.7, 0.8, 1.0)
brightness_index = 2
pixels.brightness = BRIGHTNESS_LEVELS[brightness_index]

pulsein = pulseio.PulseIn(board.A0, maxlen=120, idle_state=True)
decoder = adafruit_irremote.GenericDecode()

SPEEDS = (0.25, 0.125, 0.1, 0.08, 0.05, 0.02, 0.01)  # Customize speed levels here
speed_index = 4
```

Animations

Here is where you set up and define your animations. You can edit colors, spacing, and lots of other variables until the animations do exactly what you want.

Pressing the left and right arrow keys will cycle through the animations in this list. You can add, subtract, or reorder them so the animations appear in the order you want.

```
def setup_animations():
    """Set up all the available animations."""
    # Animation Setup
    rainbow = Rainbow(pixels, speed=SPEEDS[speed_index], period=2, name="rainbow",
step=3)
    sparkle = Sparkle(pixels, speed=SPEEDS[speed_index], color=color.WHITE,
name="sparkle")
    solid = Solid(pixels, color=colorwheel(0), name="solid")
    # Make the Solid animation changeable quickly.
    solid.speed = 0.01
    off = Solid(pixels, color=color.BLACK, name="off")
    rainbow = Rainbow(pixels, speed=SPEEDS[speed_index], period=6, name="rainbow",
step=2.4)
    rainbow_carousel = RainbowChase(pixels, speed=SPEEDS[speed_index], size=4,
spacing=1, step=20)
    party_chase = RainbowChase(pixels, speed=SPEEDS[speed_index], size=1,
spacing=5, step=6)
    rainbow_chase2 = RainbowChase(pixels, speed=SPEEDS[speed_index], size=10,
spacing=1, step=18)
    chase = Chase(pixels, speed=SPEEDS[speed_index], color=color.RED, size=1,
spacing=6)
    rainbow_comet2 = RainbowComet(
        pixels, speed=0.02, tail_length=104, colorwheel_offset=80, bounce=False)
    rainbow_comet3 = RainbowComet(
        pixels, speed=SPEEDS[speed_index], tail_length=25,
        colorwheel_offset=128, step=4, bounce=False)
    lava = Comet(pixels, speed=SPEEDS[speed_index],
                 color=color.ORANGE, tail_length=40, bounce=False)
    sparkle1 = Sparkle(pixels, speed=SPEEDS[speed_index], color=color.BLUE,
num_sparkles=10)
    pulse = Pulse(pixels, speed=0.1, color=color.AMBER, period=3)
    sparkle_pulse = SparklePulse(pixels, speed=0.05, period=2, color=color.JADE,
max_intensity=3)

    # Animation Sequence Playlist -- rearrange to change the order of animations
    # advance_interval is None, so the animations change only under user control.
    all_animations = AnimationSequence(
        rainbow,
        rainbow_chase2,
        rainbow_carousel,
        party_chase,
```

```
        rainbow_comet2,
        rainbow_comet3,
        sparkle_pulse,
        pulse,
        chase,
        rainbow,
        solid,
        sparkle,
        lava,
        sparkle1,
        off,
        auto_clear=True,
        auto_reset=True,
        advance_interval=None,
    )
    return all_animations
```

IR Mapping

The next section deals with mapping the button keys using the Adafruit IR remote (http://adafru.it/389). If you want to use a different remote, change the codes to match your remote here.

```
CMD_1 = 247          #  1: [255, 2, 247, 8]
CMD_2 = 119          #  2: [255, 2, 119, 136]
CMD_3 = 183          #  3: [255, 2, 183, 72]
CMD_4 = 215          #  4: [255, 2, 215, 40]
CMD_5 = 87           #  5: [255, 2, 87, 168]
CMD_6 = 151          #  6: [255, 2, 151, 104]
CMD_7 = 231          #  7: [255, 2, 231, 24]
CMD_8 = 103          #  8: [255, 2, 103, 152]
CMD_9 = 167          #  9: [255, 2, 167, 88]
CMD_0 = 207          #  0: [255, 2, 207, 48]

CMD_UP = 95          # ^ : [255, 2, 95, 160]
CMD_DOWN = 79        # v : [255, 2, 79, 176]
CMD_RIGHT = 175      # &gt; : [255, 2, 175, 80]
CMD_LEFT = 239       # &lt; : [255, 2, 239, 16]

CMD_ENTER_SAVE = 111 # Enter/Save: [255, 2, 111, 144]
CMD_SETUP = 223      # Setup: [255, 2, 223, 32]
CMD_STOP_MODE = 159  # Stop/Mode: [255, 2, 159, 96]
CMD_BACK = 143       # Back: [255, 2, 143, 112]

CMD_VOL_DOWN = 255   # Vol - : [255, 2, 255, 0]
CMD_VOL_UP = 191     # Vol + : [255, 2, 191, 64]
CMD_PLAY_PAUSE = 127 # Play/Pause: [255, 2, 127, 128]
CMD_REPEAT = True    # short code: repeat of previous command
```

Solid Color Keys

The 0-9 buttons will select solid colors from the rainbow, with 0 as "black", which will turn the LEDs off. You can edit the colors, or assign different animations to the buttons by editing this code.

```
SOLID_COLORS = {
    CMD_0 : color.BLACK,
```

```
    CMD_1 : color.RED,
    CMD_2 : color.GREEN,
    CMD_3 : color.WHITE,
    CMD_4 : color.BLUE,
    CMD_5 : color.PINK,
    CMD_6 : color.YELLOW,
    CMD_7 : color.PURPLE,
    CMD_8 : color.TEAL,
    CMD_9 : color.ORANGE,
    }
```

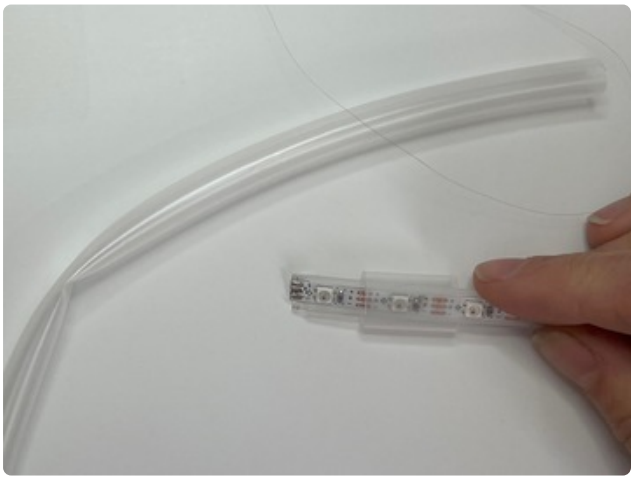
Main Loop

The main loop starts with the `while True:` line. First, the code checks to see if a button has been pressed. If you open the serial monitor and press some buttons on the remote, you'll see a readout of which code is being transmitted.

Finally, the code tells the NeoPixels how to behave when a button is pressed. This is the part you'll edit if you want to customize your button layout.

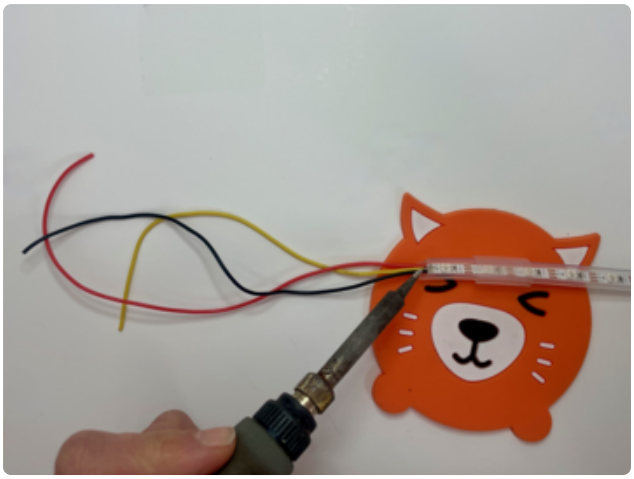
```
# if control mode is 0..
    # control the volume of the white noise
    if ctrl_mode == 0:
        # encoder neopixel is blue
        pixel0.fill(BLUE)
        # if the encoder moves..
        if pos0 != last_pos0:
            # if you increase the encoder
            # increase value by 0.1
            # maxed out at 1
            if pos0 &gt; last_pos0:
                volume = volume + 0.1
                if volume &gt; 1:
                    volume = 1
            # if you decrease
            # decrease value by 0.1
            # minimum value of 0
            if pos0 &lt; last_pos0:
                volume = volume - 0.1
                if volume &lt; 0:
                    volume = 0
            print(volume)
            # reset the position
            last_pos0 = pos0
```

Electronics Assembly



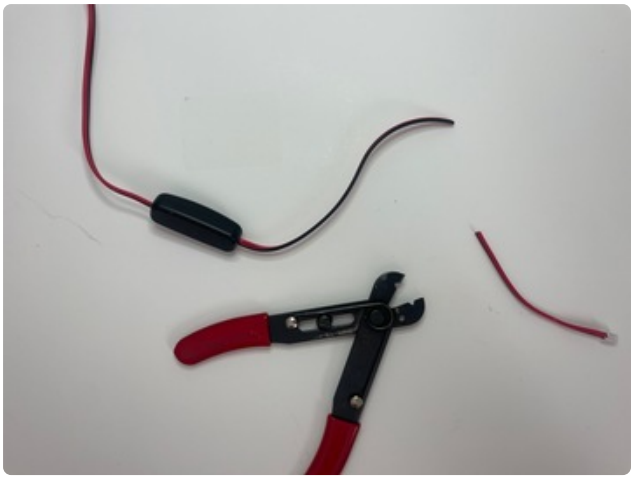
Cut your NeoPixel strip to length so it wraps all the way around your acrylic circle.

Since this project uses liquids in proximity to LEDs, we're going to make this strip fully waterproof, in case our table leaks. Slip a piece of clear heat shrink tubing over each end before you start soldering. Later, we'll seal it up using hot glue and a heat gun.

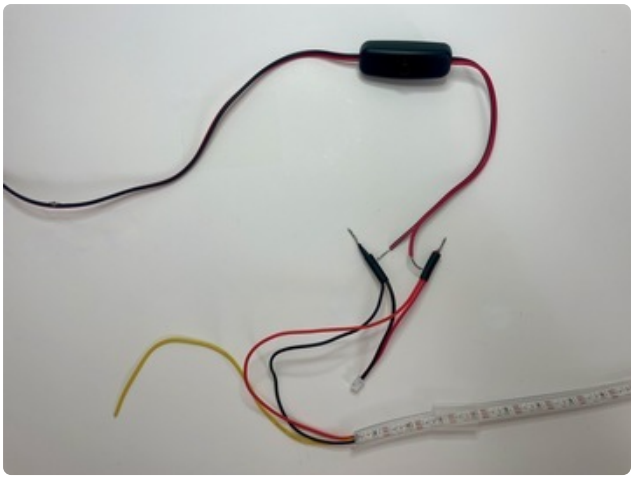


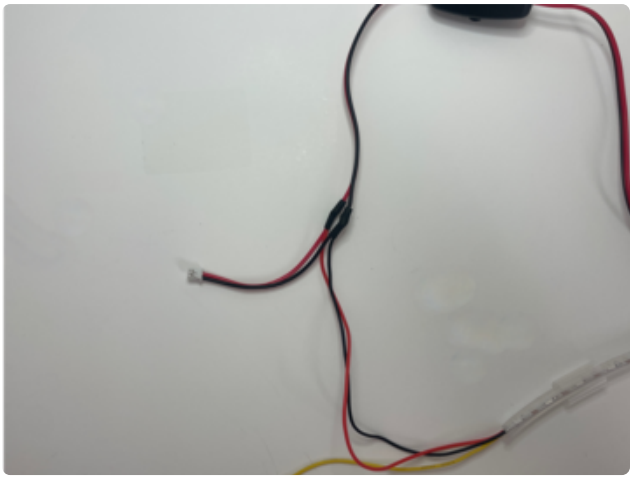
Solder 3 wires to the IN end of your pixel strip, or connect to the wires that are already there if you're using a brand-new strip. Red goes to +, yellow to **DI**, and black to **G**.

For detailed instructions check out our How to Solder NeoPixels guide. (https://adafru.it/LDV)



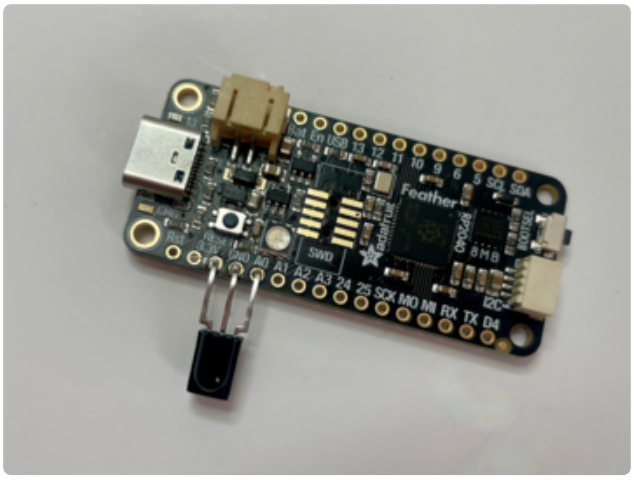
We'll splice the NeoPixel power and ground lines into our power switch cable. We could solder directly to the board, but drawing power for the pixels through the board can be problematic with more than 50 pixels. My project has 85, so I'm keeping my microcontroller safe by wiring power directly from the battery.



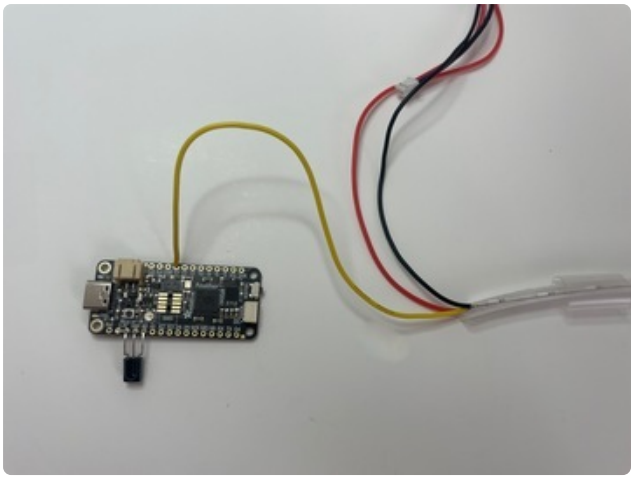


Cut the female connector off the switch extension, and twist the connector wires together with the red and black wires going to the NeoPixel strip. Slide a piece of heat shrink over the two twisted wires.

Re-connect the female JST connector to your twisted wires, and securely solder all three wires together.



Solder the IR sensor to the board with the bump facing up, using **3.3v**, **G** and **A0**, which are all helpfully lined up next to each other.



Solder the yellow wire from DI on the NeoPixels to pin 13 on the microcontroller. You can use any GPIO pin here, just be sure to change it in the code if you use a different one.

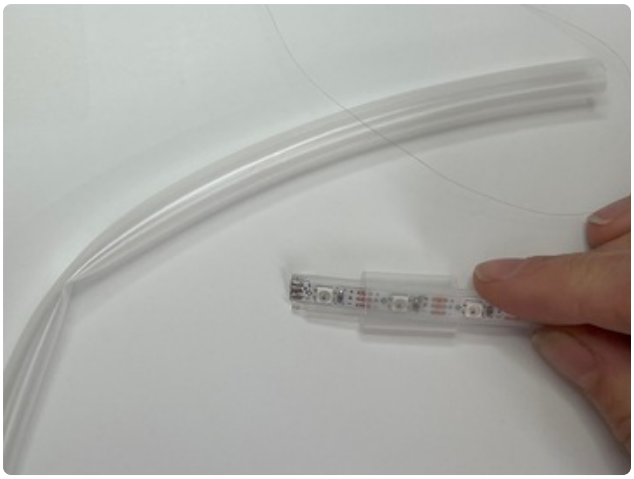
Testing

Upload your code, if you haven't already. Plug your battery into the switch, and the switch into the board. Click the switch and be sure your lights come on. Grab your remote and press some buttons to watch the animations change.

Troubleshooting

If your lights don't come on, here are a few things to try:

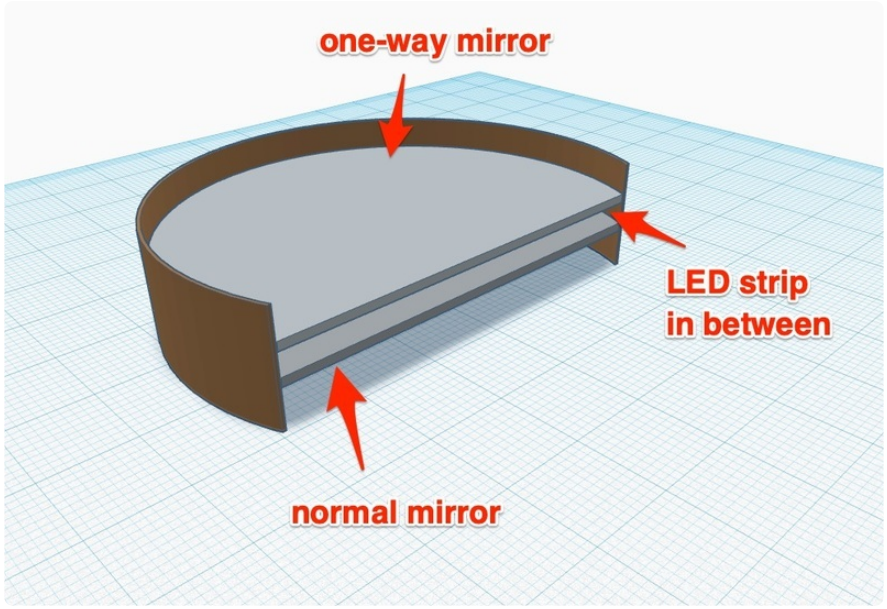
1. Check your solder joints. These solder pads are tiny! Even the littlest bridge of solder will make the strip not work.
2. Be sure you've soldered to the IN end of the strip and not the OUT end.
3. If you soldered to a pin other than 13, you need to change the code to reflect your pin number. Be sure they match.
4. If your IR remote isn't working, check to be sure you've removed the battery-saving plastic from the battery compartment.
5. Be sure the bump on the IR sensor is facing you as you look at the controller and check to be sure your pins are correct.
6. Click the switch! With this wiring diagram, the lights won't come on unless the switch is in the "on" position, even if you're plugged in via USB.
7. Try reloading the software and be sure you're getting readouts for your IR button presses in the serial monitor.



Once you're sure it's all working, slip a piece of clear heat shrink onto your LED strip. Squirt some hot glue into your heat shrink and position it over the open end of the strip. While the glue is still soft, use a heat gun to shrink the heat shrink securely to the strip. This will make a waterproof seal, so our bubble juice can't get into the strip even if our table leaks.

Seal the other end of the pixel strip as well.

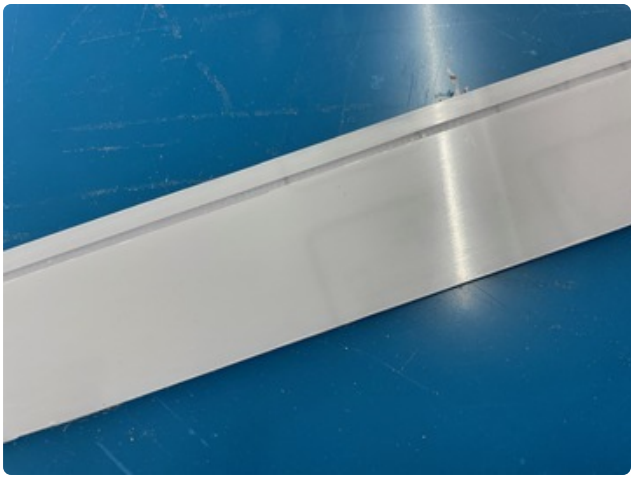
Table Build



An infinity mirror is an optical illusion that creates the appearance of an endless tunnel of light. It is created by placing two mirrors facing each other, with a light

source in between them. The mirrors reflect the light back and forth, creating the illusion of an infinite series of reflections.

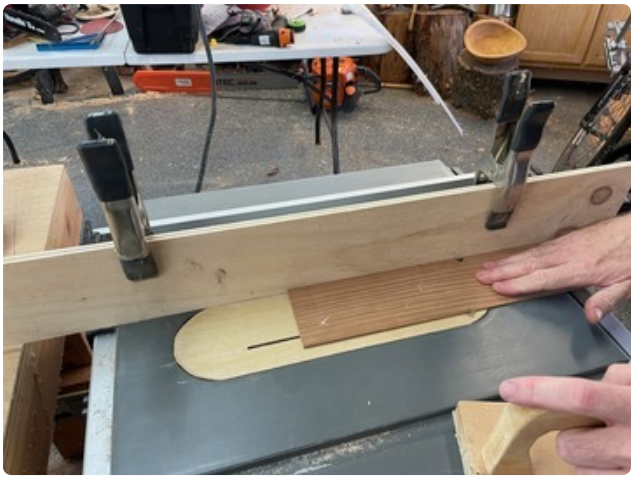
My bubble table is a modified, more diffused version of an infinity mirror. I used stick-on mirror film on the bottom mirror for maximum reflection power. Instead of one-way mirror film on the top, I used a few coats of mirror effect spray paint. This creates a more diffused surface where the individual LEDs aren't so visible - the whole table glows. You can use whichever method makes your bubble-making heart sing.



You'll need two plastic circles for the table top and the reflective mirror underneath. You have a few options, so depending on your budget and your access to tools, you can spend some money and get pre-cut pieces, or take a little more time and cut them yourself.

I used a pre-cut circle from Tap Plastics for the top of my table, since I want a really good seal on that piece. I used a band saw to cut the bottom circle out of some scrap plastic I had lying around, since the bottom circle doesn't show and doesn't need to have a perfect waterproof seal.

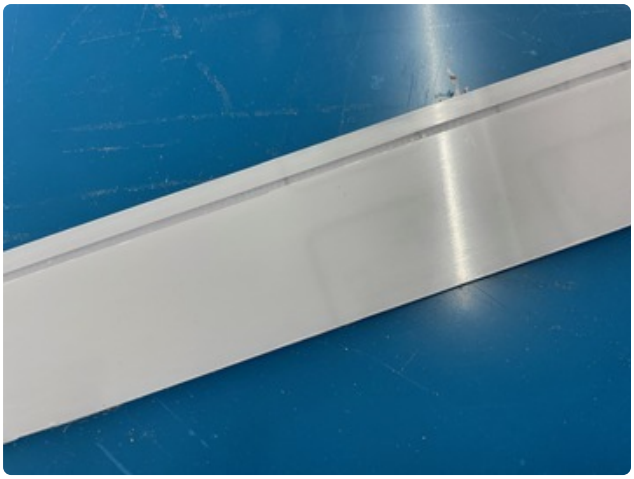
You'll also need a long strip of plastic to serve as the table edge. I used 1/8" polycarbonate from Tap Plastics and got them to cut it to size for me. It needs to be long enough to wrap all the way around your circle with a little extra for overlap.



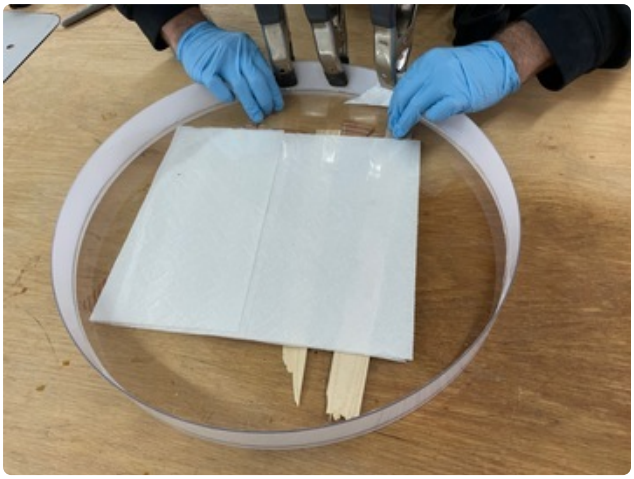
I used a table saw to cut a shallow groove in my plastic, to better hold the table top in place and make for a better seal. (You may be able to get the plastic store folks to do this for you as well).

This is a bit of a delicate operation. Set the saw blade so it's barely peeking up out of the table, and clamp an extra fence onto your table's fence or your plastic might slide underneath.

Do a test on a piece of scrap to be sure you've got the depth right. It took 2 passes to get a wide enough shallow groove to set the table top in.



When you're done, you should have a nice even groove that runs the length of your table edge plastic. Mine is about 1/4" down from the top of the edge, creating a 1/4" lip around the top of the table to hold the bubble juice in. If I were to do this again I'd make the lip a little bit taller, maybe 1/2" instead of 1/4", to minimize spills.

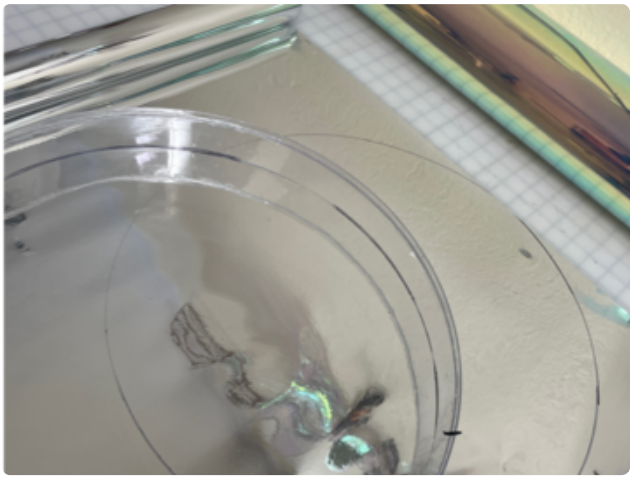


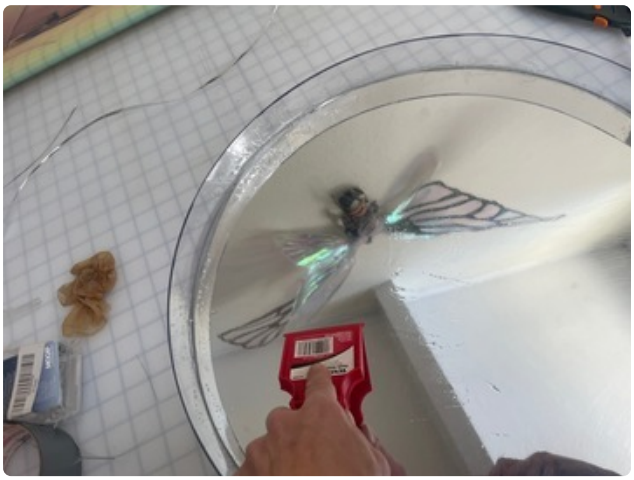
Set your circle on a towel to raise it up about 1/4" from your workbench. Wrap your table edge piece around so the circle fits neatly into the groove. Use some E6000 glue to secure the overlapped section of the table edge. Clamp it securely and let it dry overnight.

Add clear waterproof silicone sealant around the top edge of your table. Press it into the joint and smooth it out nicely. Be sure to cover the whole edge - we're looking for a 100% waterproof seal here.

Once the sealant dries, fill the table top with water and check carefully for any leaks.

Mirror Film





Apply your standard mirror film to the lower circle. Some films come with adhesive on one side, and some don't. Follow the manufacturer's directions to get the mirror smoothly applied onto your plastic.

If your film doesn't come with adhesive, you can use spray glue to attach it to the plastic. Don't worry too much if your film isn't perfect. This stuff is pretty hard to work with, but with our diffused surfaces it won't matter much if you have a few bubbles.

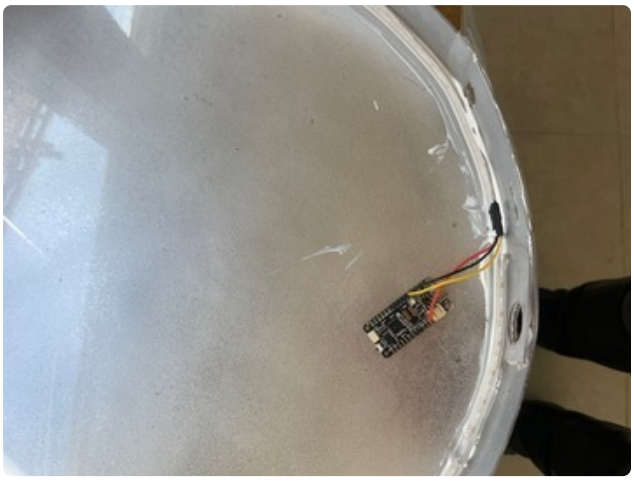
Mirror Effect Spray Paint

For my top circle I coated the underside with 3-4 very thin coats of mirror effect spray paint. Each coat adds a little more reflection. Keep adding coats until you have a one-way mirror that's got a nice amount of diffusion.

I finished it off with a coat of clear glaze spray paint to keep it from getting scratched.

You could instead use one-way mirror film here for a true infinity mirror look.

Final Assembly



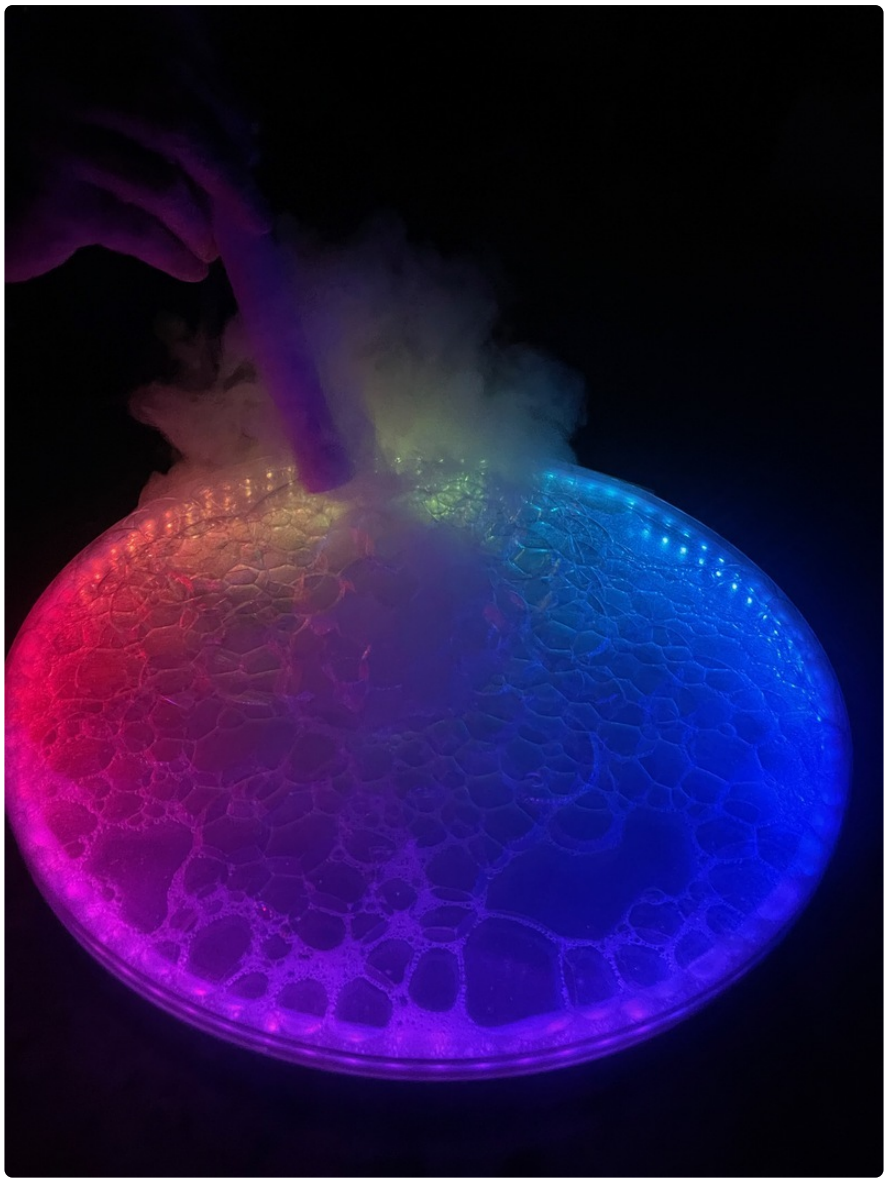
Once the silicone caulk on the top surface is dry, flip the table over and add another round of caulk on the underside of the table top. Grab your NeoPixel strip and press it into the caulk while it's wet. This silicone sealant will hold the pixels in place like glue.

Nestle the lower circle into place, resting it on the NeoPixel strip. Thread the Feather and wires around so the Feather isn't caught between the layers. Use a bit of hot glue to secure it to the bottom circle. It's a good idea to move it away from the edge a little bit, just to keep it extra-safe from any leaks.

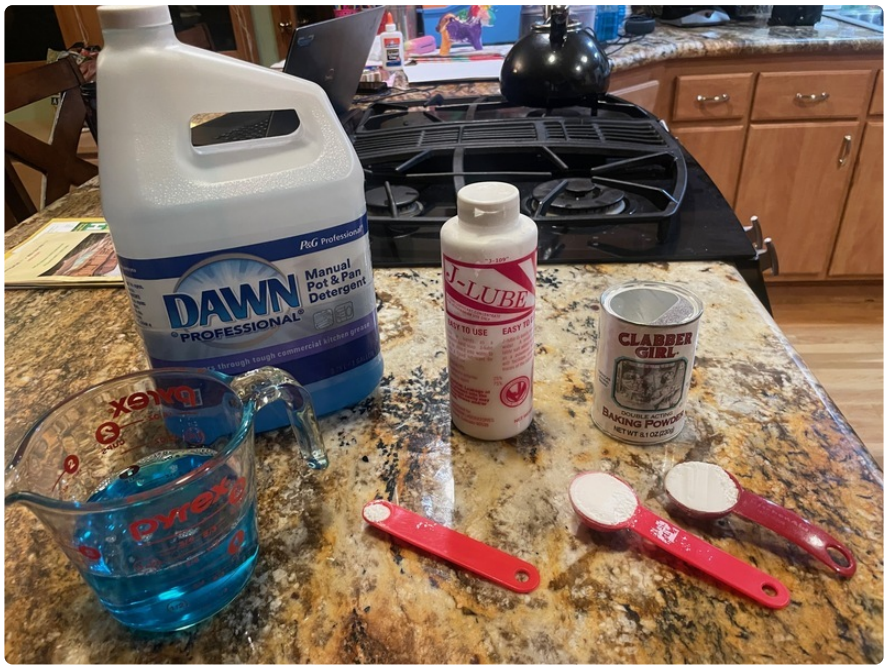
Add a third round of silicone caulking to the underside of the second layer to secure everything in place.

Similarly secure your on/off switch and battery to the underside of the table where they aren't in danger of getting wet.

The IR sensor will work right through the mirrored surfaces so no need to have it peeking out anywhere.



Bubbles



Bubble Recipe

This recipe comes from the Soap Bubble Fandom Wiki (https://adafru.it/18A4) and it works wonderfully.

Mike's "Stir-&-Go"

Recipe:

- 1 gallon of HOT tap water
- .5 gallon of COLD tap water
- 1.25 cups of Dawn Professional Manual Pot and Pan detergent
- 2 level teaspoons of Clabber Girl double acting baking powder (Other baking powder should work too)
- .5 level teaspoon (1.5g) of J-Lube (https://adafru.it/18A5)

Instructions:

1. Fill bucket with 1 gallon of the hottest tap water possible. (Mark your bucket at this level for future mixes so you can fill directly from the sink.)
2. Sprinkle in the J-Lube as slowly as possible to avoid clumping while quickly stirring the water with a chopstick. (I use a coated/lacquered chopstick to keep

the J-Lube from sticking and accumulating on it, you can probably use a knife or fork.)
3. Continue stirring for a minute.
4. Add .5 gallon of cold tap water.
5. (Mark the bucket at this level for future mixes.)
6. Pour in the Dawn and let it settle on the bottom of the bucket without stirring.
7. Now, sprinkle in the baking powder while quickly stirring the entire solution. You may feel the solution thicken after a few stirs!
8. Once all the baking powder on top has been mixed in, you're ready to make some awesome bubbles.
9. Don't forget to pray for gentle and steady wind, high humidity, and no bugs!

IMPORTANT NOTE: The amount of J-Lube you use may have to be adjusted. This recipe is based on full-potency J-Lube. The result will be a slightly stringy mix. The amount of J-Lube is about 8 times what we call the nominal minimum effective concentration (NMEC) of fresh J-Lube. If your mix is not "stringy" at all you may need to increase the amount by 2-4 times.

Fog-filled Bubbles

Smoke or fog-filled bubbles look amazing on this table and are really fun to make. You can use a fog machine or a nicotine-free vape to make the smoke. Nicotine-free vapes are around $15-20 for a one-time use disposable, or a little more for a rechargeable and refillable one. Or look for a fog machine that blows fog directly into the bubbles without having to spend any time in your lungs first.