Weather Wise Wifi Umbrella Stand

Created by Erin St Blaine

https://learn.adafruit.com/weather-wise-wifi-umbrella-stand

Last updated on 2024-03-08 03:55:40 PM EST

Table of Contents

Overview

> You Must Have Seen One Of The Spirits Of The Forest, And That Means You're A Very Lucky Girl. You Can Only See The Spirits If They Want You To. Let's Go Give Them A Proper Greeting.
>
> Tatsuo, to Mei, "My Neighbor Totoro"

Rainy days can get us down. Some mornings taunt us with gorgeous blue skies, promising spring breezes and sunshine all day, but then turn dark and stormy in the afternoon. Did you have time to check a weather app, and remember to bring your umbrella? You did not.

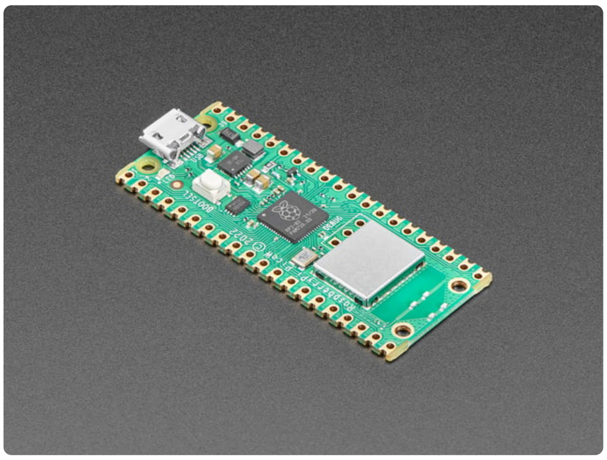
Let the spirits of the forest work for you. Build yourself a smart umbrella stand that knows the weather and lets you know with a beautiful glow if you are going to need your umbrella that day. My umbrella stand is themed after "My Neighbor Totoro" and gives me a smile every time I walk past it.

The Raspberry Pi Pico W connects to your WiFi and performs a simple task: is it going to rain today? If the answer is yes, your umbrella stand lights up and glows, reminding you to grab your umbrella.

This is a fairly easy project that's suitable for beginners. You'll need basic soldering skills and a little ingenuity. The code is written in CircuitPython, which is easy to install and update on the Pico W.

You'll also need a free account with Adafruit IO (https://adafru.it/V5A), Adafruit's online cloud service. This is a great project for getting started with internet-connected making.

Parts



Raspberry Pi Pico W
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...
https://www.adafruit.com/product/5526

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

1 x USB Cable
USB Micro to USB A Cable
https://www.adafruit.com/product/592

5v switching USB Power Supply

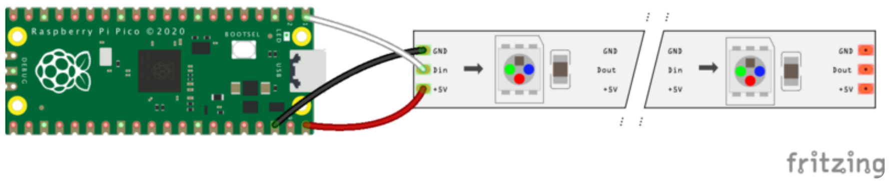
1 x Power Supply 5v switching USB Power Supply	https://www.adafruit.com/product/1994
1 x Heat Shrink Clear Heat Shrink Tubing	https://www.adafruit.com/product/1020
3 x Silicone Stranded Wire Silicone Stranded wire - Get 3 colors	https://www.adafruit.com/product/2513

Additional Materials & Tools

- Hot glue gun
- Heat gun
- Soldering Iron & Accessories
- Devcon Silicone Glue (https://adafru.it/18Cd) - available at most hardware stores

I decorated my umbrella stand with a 3d-printed figurine of Totoro, and also a large vinyl sticker. I used an Elegoo Saturn resin printer and a Cricut Maker vinyl cutter to make these.

Wiring Diagram



My Umbrella stand uses 30 NeoPixels at 60/m, which is just the right amount to go around my glass vase. If your project uses more than around 30 pixels, you'll want to power the pixels directly from the power supply instead of drawing power through the board. With small projects of up to 30 pixels, this simple wiring method works fine.

- Pico **VBUS** to NeoPixel +**5v**
- Pico **G** to NeoPixel **G**
- Pico **GP0** to NeoPixel **DIN**

We will power our project by plugging a USB cable into the Raspberry Pi Pico, which will power the NeoPixels as well.

If your project has a lot of pixels, check out this guide for wiring tips and tricks:

NeoPIO: Drive lots of LEDs with Raspberry Pi Pico (https://adafru.it/18Cg)

Installing 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

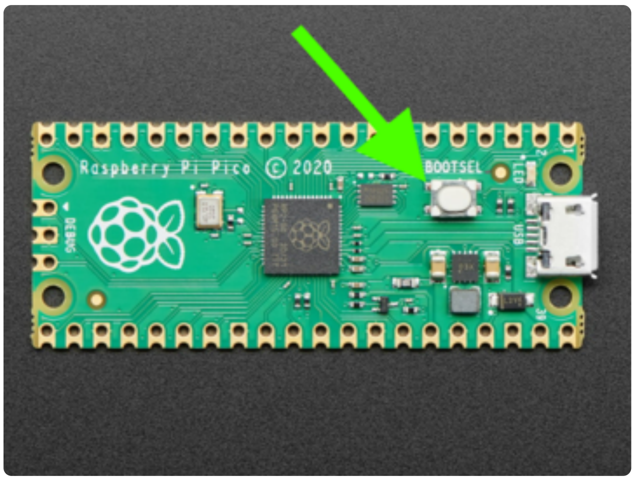
Follow this step-by-step to quickly get CircuitPython working on your board.

Download the latest version of CircuitPython for the Raspberry Pi Pico from circuitpython.org

https://adafru.it/QaP

Click the link above and download the latest UF2 file.

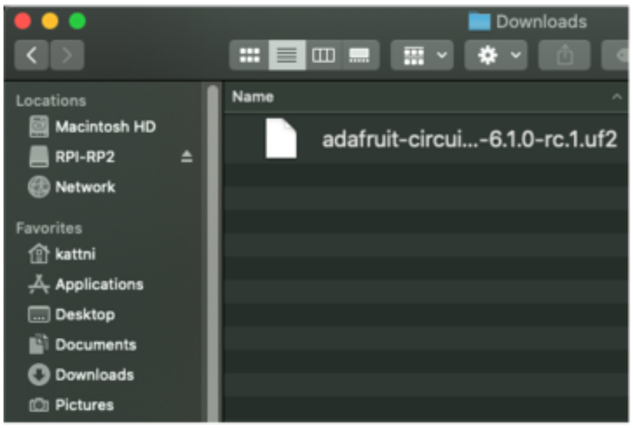
Download and save it to your desktop (or wherever is handy).



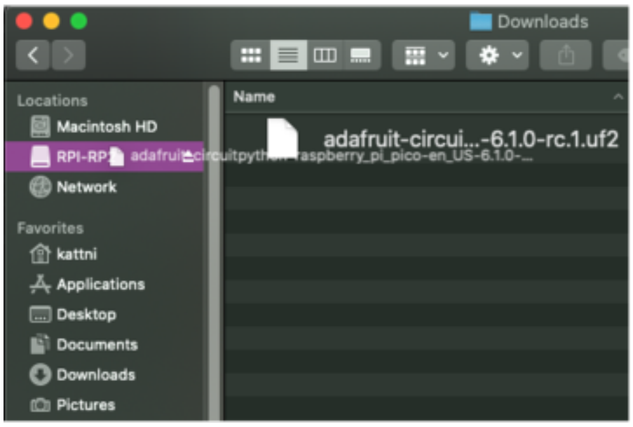
Start with your Pico unplugged from USB. Hold down the **BOOTSEL** button, and while continuing to hold it (don't let go!), plug the Pico into USB. **Continue to hold the BOOTSEL button until the RPI-RP2 drive appears!**

If the drive does not appear, unplug your Pico and go through the above process again.

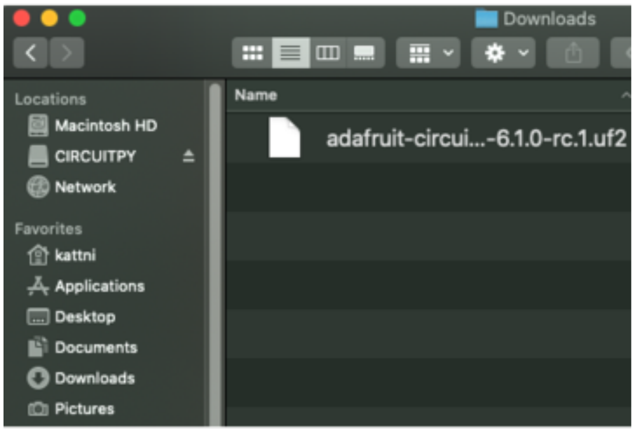
A lot of people end up using charge-only USB cables and it is very frustrating! **So make sure you have a USB cable you know is good for data sync.**



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! :)

Flash Resetting UF2

If your Pico ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try installing this 'nuke' UF2 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 nuking, re-install CircuitPython

flash_nuke.uf2

https://adafru.it/QAJ

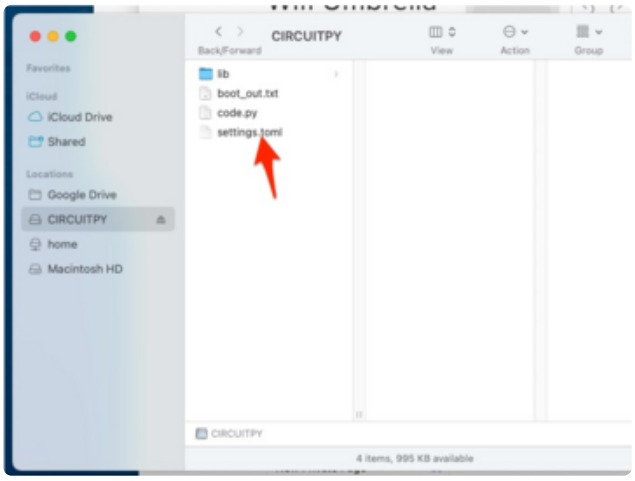
WiFi and Adafruit IO Setup

To make our code work, first we need to give our Pico W our WiFi credentials and our personal key to Adafruit IO, Adafruit's free cloud service.

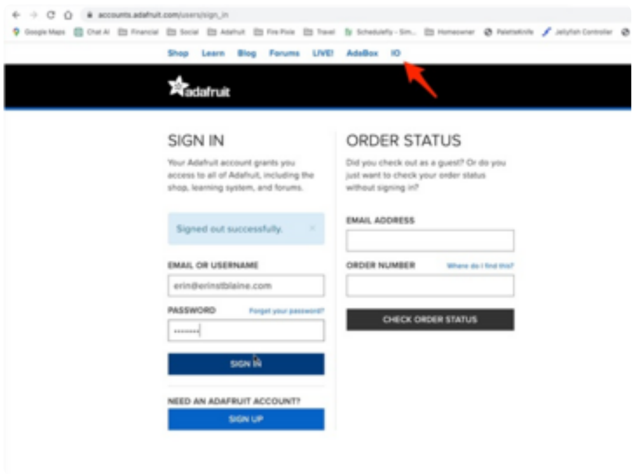
If you are not familiar with Adafruit IO, see this Welcome tutorial (https://adafru.it/BRB) to get started.

For this project we're using Adafruit IO to keep track of the time of day so we can automatically turn our stand off at night time.

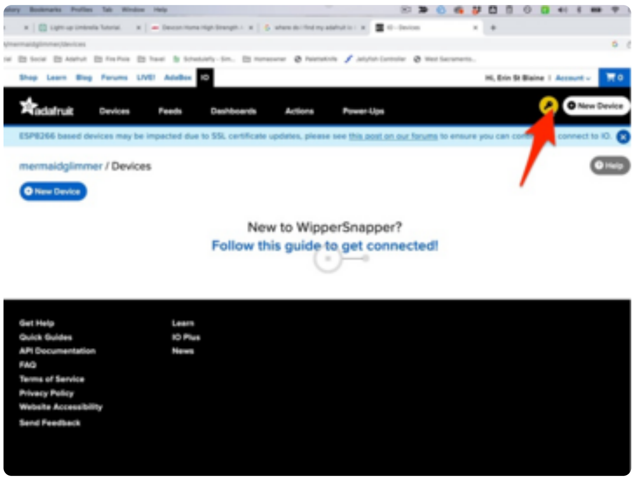
Adafruit IO is included with your Adafruit shop account, so there's no need to sign up for anything new. This is a free service we provide for our customers to help you make the magic happen.



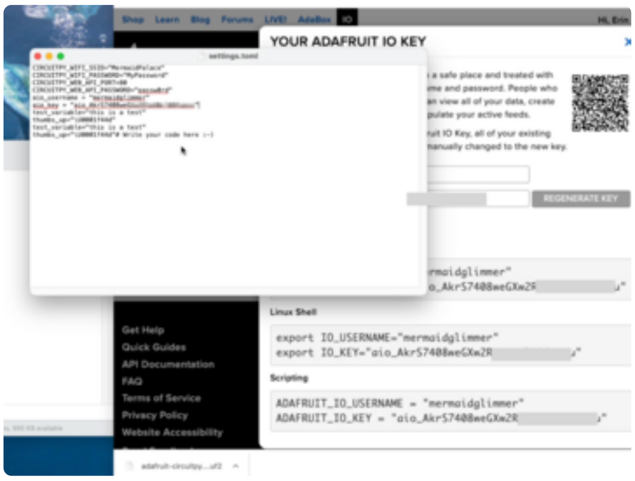
Once you've got CircuitPython installed on your Pico W, take a look at the **CIRCUITPY** drive to see what files are on there. You should find a file called **settings.toml** at the root of your drive. Open this up with a text editor - this is where to add your personal data.



Log in to your Adafruit account. Then click the IO tab at the top of the screen.



Click on the yellow key icon in the upper right corner.



```
export IO_USERNAME="mermaidglimmer"
export IO_KEY="aio_AkrS7408weGXm2R[redacted]"
```

```
ADAFRUIT_IO_USERNAME = "mermaidglimmer"
ADAFRUIT_IO_KEY = "aio_AkrS7408weGXm2R[redacted]"
```

Copy your Adafruit IO username and password into your **settings.toml** file using the following format, then save the file back to your **CIRCUITPY** drive:

```
aio_username = "username here"
aio_key = "aio key here"
```

That's it for **settings.toml**. [You can learn more about this file and what it's used for in this guide.](https://adafru.it/18f9) (https://adafru.it/18f9)

Code Install & Walkthrough

Once you've finished setting up your Pico W 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 as a zipped folder.

```
# SPDX-FileCopyrightText: 2023 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import os
import ssl
import time
import microcontroller
import board
import wifi
import socketpool
import adafruit_requests
import neopixel
from adafruit_ticks import ticks_ms, ticks_add, ticks_diff
from adafruit_io.adafruit_io import IO_HTTP

# latitude
lat = 38.58
# longitude
long = -121.49
# hours in 24 hour time that the pixels should be off
hours_off = (0, 6)
# color of the pixels when rain is expected
PIXELS_COLOR = (0, 0, 255)

# neopixel setup
NUMPIXELS = 30  # number of neopixels
BRIGHTNESS = 0.5  # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.
PIN = board.GP0

pixels = neopixel.NeoPixel(PIN, NUMPIXELS, brightness=BRIGHTNESS, auto_write=False)

# turn on NeoPixels on boot to check wiring
pixels.fill(PIXELS_COLOR)
pixels.show()

# API request to open-meteo
weather_url = "https://api.open-meteo.com/v1/forecast?"
# pass latitude and longitude
# will return sunrise and sunset times
weather_url += "latitude=%d&longitude=%d&timezone=auto&hourly=rain&forecast_days=1"
% (lat, long)

#  connect to SSID
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

pool = socketpool.SocketPool(wifi.radio)

# adafruit IO info
aio_username = os.getenv('aio_username')
aio_key = os.getenv('aio_key')

# io HTTP for getting the time from the internet
io = IO_HTTP(aio_username, aio_key, requests)

def reset_on_error(delay, error):
    print("Error:\n", str(error))
    print("Resetting microcontroller in %d seconds" % delay)
    time.sleep(delay)
    microcontroller.reset()

# function for making http requests with try/except
def get_request(tries, ping):
    for i in range(tries):
        try:
            n = ping
        except Exception as error:
            print(error)
            time.sleep(10)
            if i < tries - 1:
                continue
            raise
        break
    return n

# pylint: disable=broad-except
# function to make a request to open-meteo & time from IO
def rain_check():
    # gets current time
    now = get_request(5, io.receive_time())
    h = now.tm_hour
    time.sleep(1)
    # make the API request
    weather_call = get_request(5, requests.get(weather_url))
    # packs the response into a JSON
    response_as_json = weather_call.json()
    # gets rain forecast
    _chance = response_as_json['hourly']['rain']
    return h, _chance

# ticks time tracker
clock = ticks_ms()

# tracker for initial start-up state
first_run = True

# 15 minutes in milliseconds
time_check = 900000

while True:
    try:
        # every 15 minutes...
        if first_run or ticks_diff(ticks_ms(), clock) > time_check:
            print("pinging Open-Meteo")
            hour, rain_chance = rain_check()
            print("Rain expected: %.2f mm" % rain_chance[hour])
            if hour in hours_off:
                print("sleeping, please don't tell me it's raining")
                color = (0, 0, 0)
            else:
                # if rain is expected. turn pixels blue
                if rain_chance[hour] > 0:
                    print("tut tut, looks like rain")
                    color = PIXELS_COLOR
                # otherwise turn pixels off
                else:
                    print("no rain expected")
                    color = (0, 0, 0)
            if first_run:
                first_run = False
            pixels.fill(color)
            pixels.show()
            # reset clock
            clock = ticks_add(clock, time_check)
    except Exception as e:
        reset_on_error(10, e)
```

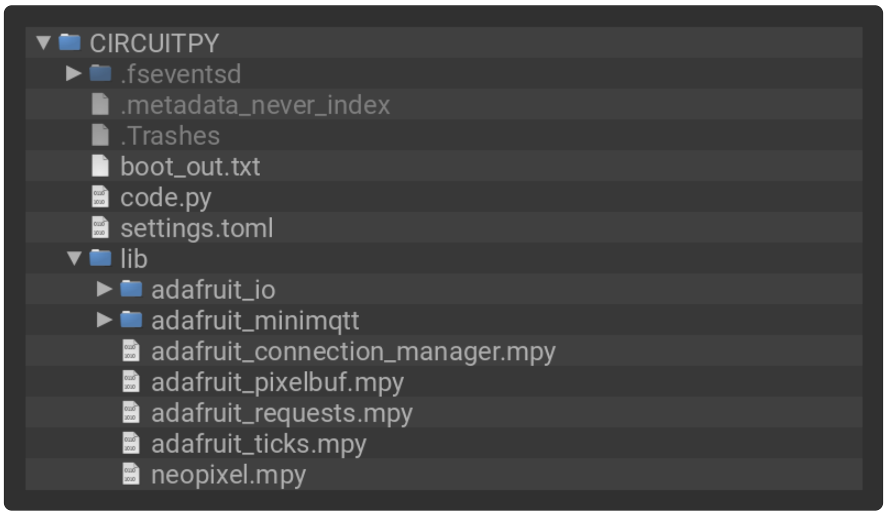
Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W 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 Pico W's **CIRCUITPY** drive.

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

Your Pico W **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file:



```
▼ CIRCUITPY
    ▶ .fseventsd
      .metadata_never_index
      .Trashes
      boot_out.txt
      code.py
      settings.toml
    ▼ lib
        ▶ adafruit_io
        ▶ adafruit_minimqtt
          adafruit_connection_manager.mpy
          adafruit_pixelbuf.mpy
          adafruit_requests.mpy
          adafruit_ticks.mpy
          neopixel.mpy
```

Add Your **settings.toml** File

As of CircuitPython 8, there is support for Environment Variables (https://adafru.it/11wE). These Environmental Variables are stored in a **settings.toml** file. Similar to **secrets.py**, the **settings.toml** file separates your sensitive information from your main **code.py** file. Be sure to add your **settings.toml** file to your **CIRCUITPY** drive. You'll need to include your `CIRCUITPY_WIFI_SSID`, `CIRCUITPY_WIFI_PASSWORD`, `aio_username` and `aio_key` in the file.

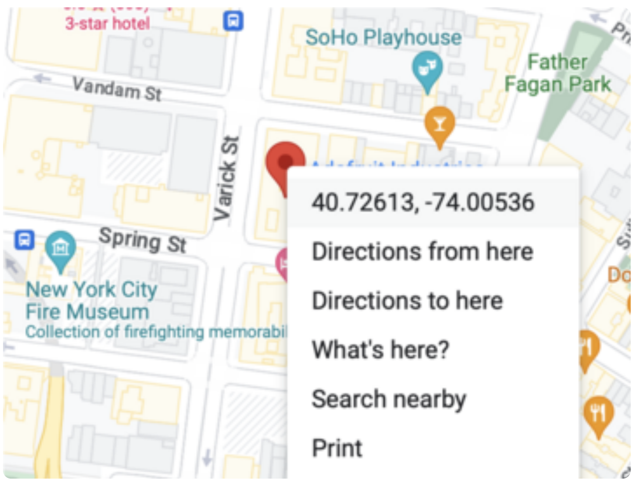
```
CIRCUITPY_WIFI_SSID = "your-wifi-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-wifi-password-here"

aio_username = "your-Adafruit-IO-username-here"
aio_key = "your-Adafruit-IO-key-here"
```

How the CircuitPython Code Works

At the top of the code, you can define some variables for your location's latitude and longitude, the hour range that you want the NeoPixels to be off and the color of the NeoPixels when rain is predicted.

```
# latitude
lat = 40.73
# longitude
long = -74.01
# hours in 24 hour time that the pixels should be off
hours_off = (0, 6)
# color of the pixels when rain is expected
PIXELS_COLOR = (0, 0, 255)
```



TIP: You can find your latitude and longitude in most phone navigation apps, or by entering your address in Google Maps and right-clicking the “pin.” First number is latitude (northern hemisphere locations are positive, southern are negative), second is latitude (negative is west of the Greenwich prime meridian, positive is east).

1–2 decimal place is sufficient for this task, weather being a regional phenomenon.

NeoPixel Setup

Then, the NeoPixels are setup to use pin `GP0` on the Pico W.

```
# neopixel setup
NUMPIXELS = 30  # number of neopixels
BRIGHTNESS = 0.5  # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.
PIN = board.GP0

pixels = neopixel.NeoPixel(PIN, NUMPIXELS, brightness=BRIGHTNESS, auto_write=False)

# turn on NeoPixels on boot to check wiring
pixels.fill(PIXELS_COLOR)
pixels.show()
```

Weather Request

The `weather_url` holds the URL for the Open-Meteo HTTP request. The URL passes your latitude and longitude that was defined earlier in the code to get the hourly rain forecast for your location.

```
# API request to open-meteo
weather_url = "https://api.open-meteo.com/v1/forecast?"
# pass latitude and longitude
# will return sunrise and sunset times
weather_url +=
"latitude=%d&amp;longitude=%d&amp;timezone=auto&amp;hourly=rain&amp;forecast_days=1"
% (lat, long)
```

Connect to WiFi

The Pico W connects to your WiFi network by accessing your SSID and SSID password in the **settings.toml** file with `os.getenv()`. Additionally, the Pico W connects to Adafruit IO to get the current time.

```
#  connect to SSID
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

pool = socketpool.SocketPool(wifi.radio)

# adafruit IO info
aio_username = os.getenv('aio_username')
aio_key = os.getenv('aio_key')
location = "America/Los Angeles"

# io HTTP for getting the time from the internet
io = IO_HTTP(aio_username, aio_key, requests)
```

Error Functions

To handle any errors that come up, two functions are defined. `reset_on_error()` runs in the event of an error. It prints the error to the REPL and then resets the Pico W to start the code over. `get_request()` makes an HTTP request and wraps it in a `try` / `except` loop so that multiple attempts can be made in the event of an error or timeout.

```
def reset_on_error(delay, error):
    print("Error:\n", str(error))
    print("Resetting microcontroller in %d seconds" % delay)
    time.sleep(delay)
    microcontroller.reset()

# function for making http requests with try/except
def get_request(tries, ping):
    for i in range(tries):
        try:
            n = ping
        except Exception as error:
            print(error)
            time.sleep(10)
            if i &lt; tries - 1:
                continue
            raise
        break
    return n
```

What's the Weather?

The `rain_check()` function uses the `get_request()` function to get the time from Adafruit IO and make a request to Open-Meteo with the previously defined `weather_url`. The function returns the current hour and the rain forecast.

```
def rain_check():
    # gets current time
    now = get_request(5, io.receive_time())
    h = now.tm_hour
    time.sleep(1)
    # make the API request
    weather_call = get_request(5, requests.get(weather_url))
    # packs the response into a JSON
    response_as_json = weather_call.json()
    # gets rain forecast
    _chance = response_as_json['hourly']['rain']
    return h, _chance
```

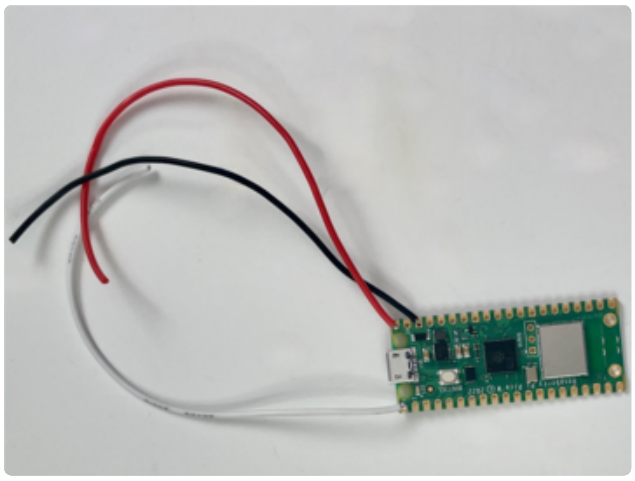
The Loop

In the loop, every 15 minutes the `rain_check()` function runs to get the rain forecast. If the current time falls in the range of the defined `hours_off`, then the NeoPixels are turned off. Otherwise, if any rain is expected, the NeoPixels turn on and if no rain is expected they are turned off.

```
try:
        # every 15 minutes...
        if first_run or ticks_diff(ticks_ms(), clock) &gt; time_check:
            print("pinging Open-Meteo")
            hour, rain_chance = rain_check()
            print("Rain expected: %.2f mm" % rain_chance[hour])
            if hour in hours_off:
                print("sleeping, please don't tell me it's raining")
                color = (0, 0, 0)
            else:
                # if rain is expected. turn pixels blue
                if rain_chance[hour] &gt; 0:
                    print("tut tut, looks like rain")
                    color = PIXELS_COLOR
                # otherwise turn pixels off
                else:
                    print("no rain expected")
                    color = (0, 0, 0)
            if first_run:
                first_run = False
            else:
                pixels.fill(color)
                pixels.show()
                # reset clock
                clock = ticks_add(clock, time_check)
    except Exception as e:
        reset_on_error(10, e)
```

Electronics Assembly

Wire the Pico W

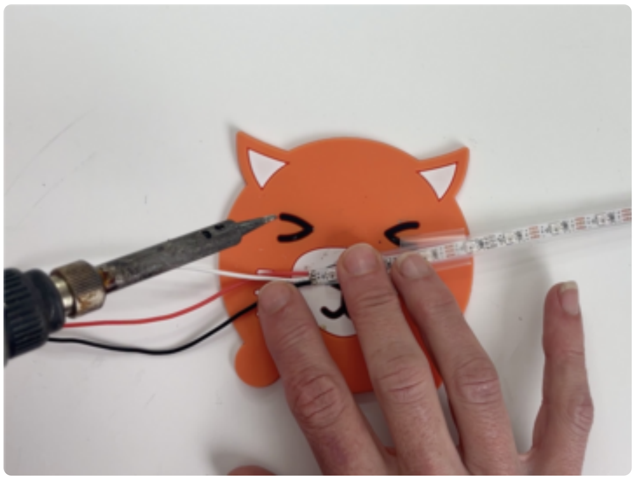


Cut a red, black, and white wire to about 3-4 inches, or longer depending on your build base. Solder the red wire to **VBUS**, the black to **G**, and the white to **GP0**.

You can see a full pinout of the Pico here (https://adafru.it/Qsd).

Solder NeoPixels

Since this project may get dripped on, it's best to seal both ends of our pixel strip so no stray splashes can get inside. Start by cutting a couple pieces of your clear heat shrink tubing and slipping them onto your pixel strand. We'll solder up our strip, test, then seal it up once we know it's working.

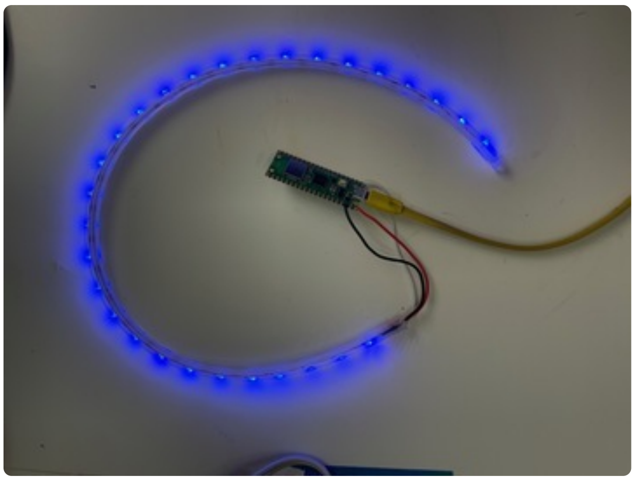


Cut your pixel strip to length through the middle of the copper pads. My strip has 30 pixels. Be sure to count twice before you cut.

Trim and tin the wires coming from the Pi, and tin the pads on the NeoPixel strip. Make sure you're soldering to the IN end - these strips are directional and won't work if you connect the wrong end.

If you're new to soldering pixel strips, check out this guide on How to Solder NeoPixels (https://adafru.it/LDV).

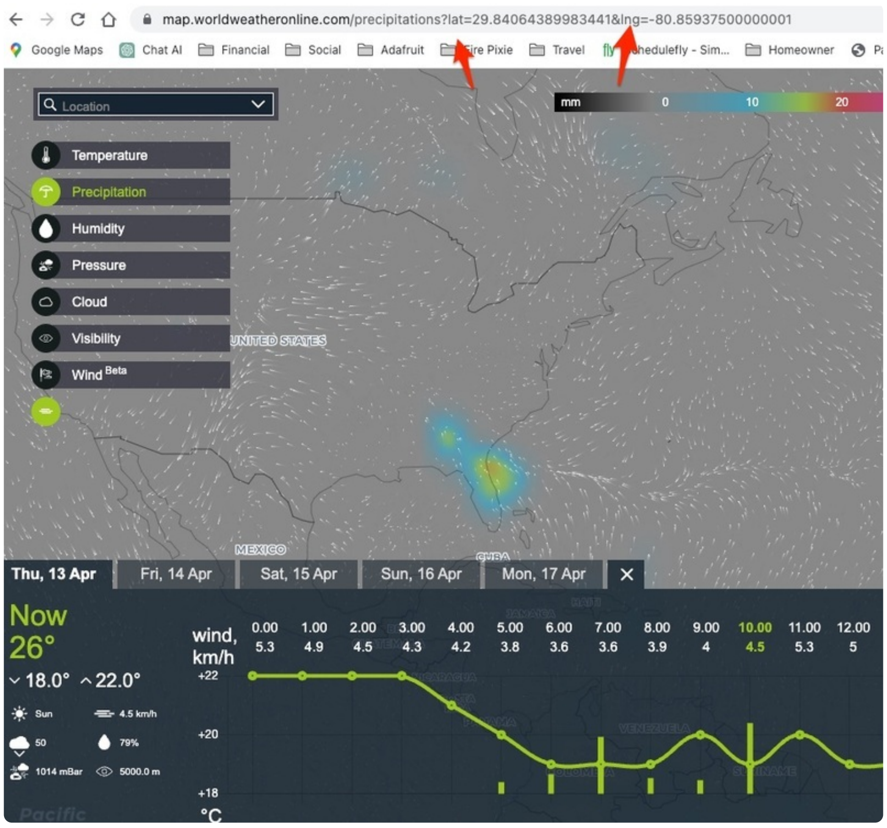
Testing



Plug your Pico W in via the USB port. Wait a few seconds for the WiFi to connect. If it's going to rain in your area, the lights will come on.

This project can be a little tricky to test. If it's not about to rain where you are, the pixels won't turn on, which means they're working correctly. But that doesn't help when you're testing solder joints. What to do?

In order to test, we need to find a place where it's currently raining and temporarily change our latitude and longitude in our **code.py** file. Here's a nifty tool: the World Weather Map (https://adafru.it/18Cm) shows weather patterns across the whole world. It's often raining in New Orleans in the spring, and the South American rain forests are full of rainy areas. Pick your favorite, click on it, and copy the latitude and longitude from the URL.



Then open up **code.py** and edit these lines to reflect the rainiest spot you can find:

```
# latitude
lat = 29.04
# longitude
long = -90.36
```

If you've got everything wired up and installed correctly, your lights will turn on. Success! Time to seal up the strips and install your lights.

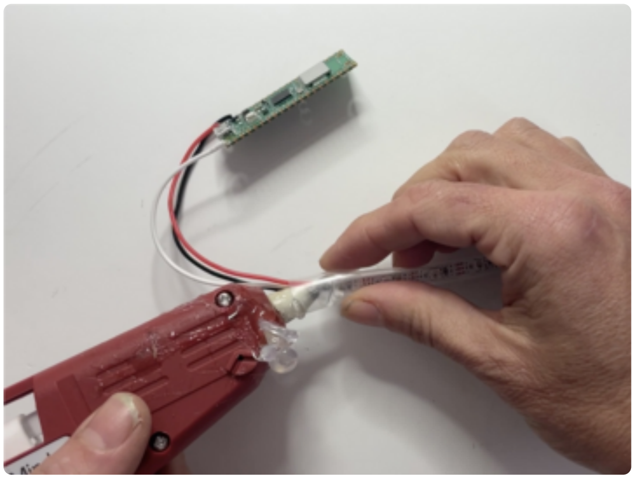
Troubleshooting

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

1. Try a different latitude and longitude, just in case you missed a rainstorm by a few ticks.
2. Check to be sure you soldered to the IN end of the pixel strand and not the OUT end. It won't work if you solder to the wrong end.
3. If your lights come partially on, or come on in wonky colors, check your power supply. If you're plugging into a surge protector, try plugging directly into the wall or into your computer and see if that fixes things. This board seems to need a full 5v to get the pixels working right, so a cheap power supply may cause problems.

4. Double check your code and your **settings.toml** file. Be sure you've entered both your WiFi credentials and your Adafruit IO key in **settings.toml**, and be sure they're correct. The code will reset the board if it's missing any data, so if your board is resetting again and again this is likely your problem.
5. Be sure you wired to GP0 on the Pi. If you wired to a different pin, you can change it in **code.py.**

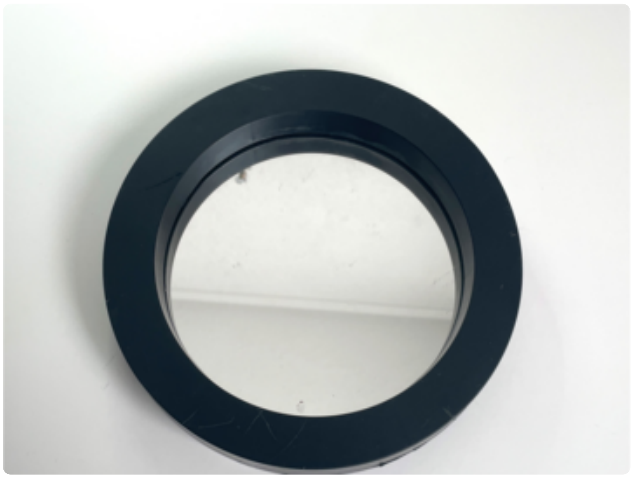
Seal the Strip



Once you're sure it's working correctly, seal up both ends of your NeoPixel strip by sliding the heat shrink over the solder joints then filling the heat shrink with some hot glue. While the glue is still wet, use a heat gun to shrink the heat shrink. This will form a waterproof seal on your pixels that will keep any rain splashes out.

TIP: do not use silicone glue for this step, as some varieties will corrode electronics.

Umbrella Stand Assembly



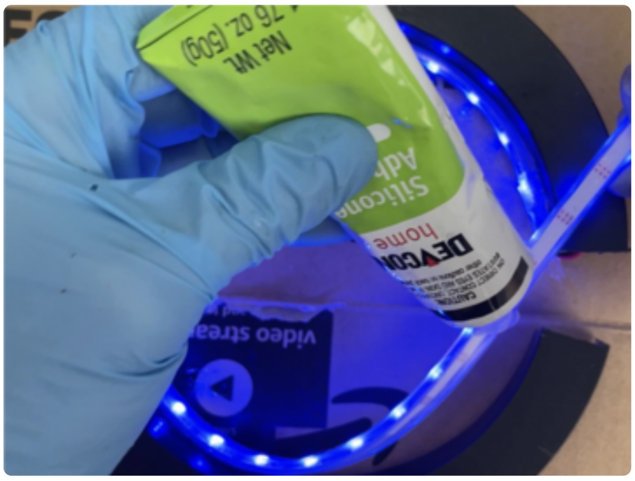
I found a large colored glass vase at my local Goodwill for around $8. It's the perfect size for one large umbrella or a couple small ones.

To house the electronics and lights, I found a round mirror with a black plastic frame, also at the Goodwill. The mirror's interior diameter is just a bit too small to fit the vase neatly, but I can make it work with a little bit of maker ingenuity.

I removed the frame from the mirrored back, then cut through the side with a dremel. Then I heated the plastic frame gently with a heat gun until the whole thing was a bit soft and pliable, but not so much that it got melty.

I plan to put the electronics inside the hollow frame, so I dremeled a channel at the back of the frame for my USB cable to pass.

I plugged in the USB cable and fit my Pico W inside the frame, securing it with some hot glue. I threaded the USB cable through the channel at the back and glued that in place as well.

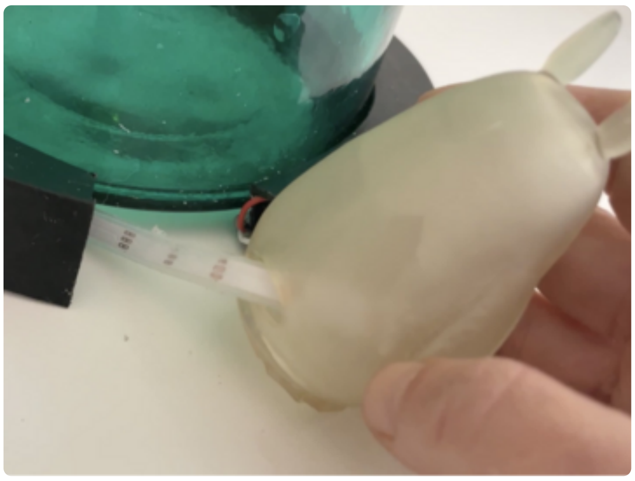


Next I glued the pixels in place around the inner edge of the frame, right at the bottom. The silicone coating on these pixels is tricky to glue -- very few glues will stick to silicone. My favorite is Devcon Silicone Glue (https://adafru.it/18Cn). It dries quickly, and clear-ish, and holds the silicone strip casing in place really well.

I left a few extra pixels at the end of my strip to illuminate Totoro.

Decorate

I filled the gap in the base with a 3D-printed Totoro figurine. I found him on Thingiverse, thank you ElectricBeard (https://adafru.it/18Cp)! I printed him on a resin printer using clear resin, which got a little cloudy during the curing process.



I drilled a hole in the back of Totoro and slipped in the last few LEDs from my NeoPixel strip. The clear resin clouded up just enough to diffuse the lights perfectly. Glowing Totoro, Hooray!

I used some Thermoplastic and silicone glue to wedge him neatly into the gap in the base.

Finally, I used my Cricut vinyl cutter to cut out a silhouette image of Totoro that I found online. I fixed it to the glass vase using frisket to keep all the pieces in place. I love the combination of 3D and silhouette Totoro characters! The sticker really brings this design to life.

Place the stand near your door and never forget your umbrella again.