Overview

This is a fun project that plays with how to visualize information using only light and sound. What better way to have some fun with this than to make an Internet connected cloud that connects to an open source weather API?

This CircuitPython based project will walk you through connecting to lights, sound, and a network cloud service. Next we add some code to make it all go, along with a few sound files. There is plenty of room for your artistic side as we turn our electronics into a super fluffy cloud. Hang it up and watch it go.

This cloud is set to display the following weather updates common to the OpenWeather API (https://adafru.it/KeU)

- Clear
- Clouds, Fog, Mist
- Drizzle, Rain
- Thunderstorm
- Snow
Tools and Supplies

There is a little bit of soldering to add connectors for this project, but nothing super difficult. Mostly we will be doing a bit of crafting to make the cloud itself.

You are also going to want to use a hot glue gun as well.

1 x High Temp Glue Gun
This will help you glue things together.

Digital Genuine Hakko FX-888D (936 upgrade)
$129.95
IN STOCK
Add To Cart

Hakko Professional Quality 20-30 AWG Wire Strippers
$14.95
IN STOCK
Add To Cart

Slice Craft Knife with Ceramic Blade
$12.50
IN STOCK
Add To Cart
Supplies

There are quite a few things you will need to make the cloud itself. Feel free to get creative and add whatever you want to make it your own.

1 x Poly-Fil polyester fiber
Commonly used as stuffing for pillows and stuffed animals.

BUY NOW

You will need something to make the cloud frame from, and though I have used laser cut wood and acrylic, I have actually found that cardboard and foam board work just as well, even for larger clouds.

1 x Foam Board, White
White foam core board for the frame of the cloud.

BUY NOW

1 x Hot Glue Sticks
You will be doing a fair bit of gluing.

BUY NOW

1 x Zip Tie Locking Ties
White plastic zip ties for holding down electronics and LEDs.

BUY NOW
Wiring

A bit of wiring is necessary to connect everything to the Metro M4 microcontroller. For this project, connect the NeoPixel strip and the speaker power directly to the 5V DC power supply so that they don't overload the Metro M4. Other than that, we will just be connecting a button and a few components for the audio output.

Audio Wiring

We will be using the Analog 0 Pin as an Audio out pin, but we do need to add some components to trim the signal and make this sound good.

1. Solder the the **Black** wire to the common pin on the 3.5mm Stereo Headphone Jack.
2. Use some **Blue** wire to jump the outer two pins on the Headphone Jack together since we will be using a Mono signal.
3. Connect one of the outer pins from the Headphone Jack to the negative side, the one with the white stripe, of a 100uf Capacitor.
4. Connect the other side of the Capacitor to one of the outer pins of a 10k Potentiometer.
5. Finally connect a **Blue** wire to the center pin, or common pin, on the Potentiometer.

The **Black** and **Blue** wires should now be set to connect with the GND and A0 pins on the **Metro M4**.
NeoPixel Wiring

Because we will be using a fair amount of NeoPixels, we need to connect the power directly to the 5V DC so that we don't overload the Metro M4.

1. Make sure that each of the following wires are about 10 inches long.
2. Connect the Red wire to the positive side, the side without the white stripe, of a 4700uF Capacitor.
3. Connect the Black wire to the negative side, the side with the white stripe, of the Capacitor.
4. Add an Orange wire to the Din pin, usually a White wire, at the beginning of the NeoPixel string.

The Orange wire should now be ready to connect to pin 2 on the Metro M4.

Power Connector Wiring

Along with the NeoPixels, we also need to power the speakers with our 5V DC connection. The speakers have a USB connector as it's power connection but this can be removed to expose a Red and Black wire.

1. Use wire cutters to cut the wire just behind the speakers USB connector.
2. Strip the outer wire casing off about 3 inches down to get to the Red and Black wires.
3. Strip about 0.25 inches off of the ends of the Black and Red wires for the Speaker and the NeoPixels.
4. Twist the two Red wire ends together and insert them into the + side of the Female DC Power adapter screw terminal.
5. Twist the two Black wire ends together and insert them into the - side of the Female DC Power adapter screw terminal.

This is now ready to be connected to the DC Barrel Jack Splitter later.
Loading CircuitPython

Mu Editor
The Mu editor works really well with the Metro M4 Express AirLift and it is the recommended choice for editing CircuitPython code.

Updating Your Metro M4 Express AirLift
This project was set up and tested using CircuitPython version 5 or higher. You will want to update your Metro M4 Express AirLift and Libraries to match the version you are using.

CircuitPython Library Installation
Next you’ll need to install the necessary libraries to use the hardware carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafruit.it/zdx) matching your version of CircuitPython. This project uses CircuitPython version 5 or later.

CircuitPython Library files used in this tutorial
Plug your Metro M4 Express AirLift into your computer via a known, good USB data + power cable (not the kind that comes with USB battery packs, those are power only). A new flash drive should show up in your computer's file explorer/finder named CIRCUITPY. If it's not there, check your cable and that you installed CircuitPython correctly earlier.

The following libraries are used here. Copy the corresponding file from the library bundle to your CIRCUITPY drive in a subdirectory named lib. Create this subdirectory, if necessary, then copy these files/directories:

- adafruit_bus_device
- adafruit_esp32spi
- adafruit_fancyled
- adafruit_requests.mpy
- neopixel.mpy

Before continuing make sure your board's lib folder has the following files and folders copied over. Note that the non-library files will be retrieved in the page called “The Full Code”.

© Adafruit Industries
OpenWeather API setup

Open Weather Maps API Key

We'll be using OpenWeatherMaps.org to retrieve the weather info through its API. In order to do so, you'll need to register for an account and get your API key.

Go to this link [https://adafru.it/EeH](https://adafru.it/EeH) and register for a free account. Once registered, you'll get an email containing your API key, also known as the "openweather token".

Copy and paste this key into your `secrets.py` file that is on the root level of your CIRCUITPY drive, so it looks something like this:

```python
# This file is where you keep secret settings, passwords, and tokens!
# If you put them in the code you risk committing that info or sharing it

secrets = {
    'ssid': 'my_ssid',
    'password': 'my_pass',
    'timezone': "America/New_York", # http://worldtimeapi.org/timezones
    'openweather_token': 'putYourOpenWeatherTokenHere',
}
```
Sound Files

The Metro M4 can play **PCM 16-bit Mono Wave files at 22KHz sample rate**. Thankfully you can find some files that were made just for this project that will work.

You can download the following sounds and place them into a new folder called `sounds` in the CIRCUITPY drive.

https://adafru.it/Kex
https://adafru.it/KeA
https://adafru.it/KeB
https://adafru.it/KeC
https://adafru.it/KeD
https://adafru.it/Ke
https://adafru.it/Key
https://adafru.it/Kez
https://adafru.it/KeA
https://adafru.it/KeB
https://adafru.it/KeC
https://adafru.it/KeD

If you would like to make your own sounds, the following link will take you to a guide that can help you format the WAV files. Keep in mind that you will want to make your files small, as WAV files can fill up your storage space quickly.

https://adafru.it/s8f
Here is the code that you will need for the code.py file in the CIRCUITPY drive.

Click Download Zip to get all the sounds as well

```python
import time
import random
import audioio
import board
import busio
from digitalio import DigitalInOut
import digitalio
import neopixel
from adafruit_esp32spi import adafruit_esp32spi
from adafruit_esp32spi import adafruit_esp32spi_wifimanager
import adafruit_fancyled.adafruit_fancyled as fancy

print("ESP32 Open Weather API demo")

button = digitalio.DigitalInOut(board.A1)
button.switch_to_input(pull=digitalio.Pull.UP)

wave_file = open("sound/Rain.wav", "rb")
wave = audioio.WaveFile(wave_file)
audio = audioio.AudioOut(board.A0)

# Get wifi details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# Use cityname, country code where countrycode is ISO3166 format.
# E.g. "New York, US" or "London, GB"
LOCATION = secrets['timezone']

# Set up where we'll be fetching data from
DATA_SOURCE = "http://api.openweathermap.org/data/2.5/weather?q=\"+secrets['timezone']\"
DATA_SOURCE += "&appid=\"+secrets['openweather_token']\"

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
status_light = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.2)  # Uncomment for Most Boards
pixels = neopixel.NeoPixel(board.D2, 150, brightness=1.0, auto_write=False)
pixels.fill(0x050505)
pixels.show()

# clouds palette
```
cloudy_palette = [fancy.CRGB(1.0, 1.0, 1.0),  # White
                 fancy.CRGB(0.5, 0.5, 0.5),  # gray
                 fancy.CRGB(0.5, 0.5, 1.0)]  # blue-gray

# sunny palette
sunny_palette = [fancy.CRGB(1.0, 1.0, 1.0),  # White
                 fancy.CRGB(1.0, 1.0, 0.0),  # Yellow
                 fancy.CRGB(0.5, 0.5, 0.0),  # Orange
                 fancy.CRGB(0.5, 0.5, 1.0)]  # blue-gray

# thunderstorm palette
thunder_palette = [fancy.CRGB(0.0, 0.0, 1.0),  # blue
                   fancy.CRGB(0.5, 0.5, 0.5),  # gray
                   fancy.CRGB(0.5, 0.5, 1.0)]  # blue-gray

last_thunder_bolt = None

class A:
    palette = None  # current palette
    pal_offset = 0  # Positional offset into color palette to get it to 'spin'
    levels = (0.25, 0.3, 0.15)  # Color balance / brightness for gamma function
    raining = False
    snowing = False
    thundering = False
    has_sound = False

weather_refresh = None
weather_type = None
button_mode = 4
button_select = False
cloud_on = True

while True:
    if not button.value:
        button_mode = button_mode + 1
        print("Button Pressed")
    if button_mode > 4:
        button_mode = 0
        print("Mode is:", button_mode)
    pressed_time = time.monotonic()
    button_select = True
    weather_refresh = None
    while not button.value:  # Debounce
        pass
    if (time.monotonic() - pressed_time) < 4:
        print("Turning OFF")
    cloud_on = False
    pixels.fill(0x000000)  # bright white!
    pixels.show()
    while not cloud_on:
        if not button.value:
            pressed_time = time.monotonic()
            print("Button Pressed")
            cloud_on = True
            button_select = False
            weather_refresh = None

            if button_mode == 0:
                weather_type = 'Sunny'
            if button_mode == 1:
                weather_type = 'Clouds'
            if button_mode == 2:
if button_mode == 2:
    weather_type = 'Rain'
if button_mode == 3:
    weather_type = 'Thunderstorm'
if button_mode == 4:
    weather_type = 'Snow'

# only query the weather every 10 minutes (and on first run)
if (not weather_refresh) or (time.monotonic() - weather_refresh) > 600:
    try:
        if not button_select:
            response = wifi.get(DATA_SOURCE).json()
            print("Response is", response)  
            weather_type = response['weather'][0]['main']
        if weather_type == 'Clear':
            weather_type = 'Sunny'
            print(weather_type)  # See https://openweathermap.org/weather-conditions
        # default to no rain or thunder
        raining = snowing = thundering = has_sound = False
        if weather_type == 'Sunny':
            palette = sunny_palette
            wave_file = open("sound/Clear.wav", "rb")
            wave = audioio.WaveFile(wave_file)
            has_sound = True
        if weather_type == 'Clouds':
            palette = cloudy_palette
            wave_file = open("sound/Clouds.wav", "rb")
            wave = audioio.WaveFile(wave_file)
            has_sound = True
        if weather_type == 'Rain':
            palette = cloudy_palette
            wave_file = open("sound/Rain.wav", "rb")
            wave = audioio.WaveFile(wave_file)
            raining = True
            has_sound = True
        if weather_type == 'Thunderstorm':
            palette = thunder_palette
            raining = thundering = True
            has_sound = True
            # pick next thunderbolt time now
            next_bolt_time = time.monotonic() + random.randint(1, 5)
        if weather_type == 'Snow':
            palette = cloudy_palette
            wave_file = open("sound/Snow.wav", "rb")
            wave = audioio.WaveFile(wave_file)
            snowing = True
            has_sound = True
        weather_refresh = time.monotonic()
    except RuntimeError as e:
        print("Some error occured, retrying! -", e)
        time.sleep(5)
        continue

if not audio.playing and has_sound:
    if not thundering:
        audio.play(wave)

if palette:
    for i in range(len(pixels)):
        color = fancy.palette_lookup(palette, pal_offset + i / len(pixels))
color = fancy.gamma_adjust(color, brightness=levels)
pixels[i] = color.pack()
pixels.show()
pal_offset += 0.01  # Bigger number = faster spin

if raining:
    # don't have a droplet every time
    for i in range(random.randint(1, 5)):  # up to 3 times...
        pixels[random.randint(0, len(pixels)-1)] = 0x0000FF  # make a random pixel Blue
        pixels.show()

if snowing:
    # don't have a droplet every time
    for i in range(random.randint(1, 5)):  # up to 3 times...
        pixels[random.randint(0, len(pixels)-1)] = 0xFFFFFF  # make a random pixel white
        pixels.show()

# if its time for a thunderbolt
if thundering and (time.monotonic() > next_bolt_time):
    print("Ka Bam!")
    # fill pixels white, delay, a few times
    for i in range(random.randint(1, 3)):  # up to 3 times...
        pixels.fill(0xFFFFFF)  # bright white!
        pixels.show()
        time.sleep(random.uniform(0.01, 0.2))  # pause
        pixels.fill(0x0F0F0F)  # gray
        pixels.show()
        time.sleep(random.uniform(0.01, 0.3))  # pause
    # pick next thunderbolt time now
    Thunder = random.randint(0, 2)
    if Thunder == 0:
        wave_file = open("sound/Thunderstorm0.wav", "rb")
    elif Thunder == 1:
        wave_file = open("sound/Thunderstorm1.wav", "rb")
    elif Thunder == 2:
        wave_file = open("sound/Thunderstorm2.wav", "rb")
    wave = audioio.WaveFile(wave_file)
    audio.play(wave)
    next_bolt_time = time.monotonic() + random.randint(5, 15)  # between 5 and 15 s
Build the cloud

Cut out the Shapes

The first few clouds I made a few years back, I tried to make the base frame look more like a cloud thinking that it would make the outside of the cloud look more fluffy. No matter the shape of the frame, cloud is going to do what cloud will do. So I switched to just making an oval with a slightly flatter bottom and it looks just like all the other more complex ones. So keep things simple.

1. Place the two speakers onto your foam core or cardboard so you can trace them out for later. Be sure to leave plenty of space for the outer shape of the cloud.

2. Continue this for other parts of the frame, remembering to add holes for the button and wires to be strung through them.

3. Use a cutting tool to cut out all of the layers for your cloud. Do not forget to cut holes for your wires and the speakers. Keep in mind that all of this will be covered with fluffy stuff to hide the crimes.

I did four layers for this cloud but you can make it as big or small as you like.
1. You will want to install the button into the first layer so that it will end up on the bottom of the cloud for easy access.

2. Use some of the cut-out pieces from your foam core or cardboard to make two pieces that can be added to the button.

3. Glue these pieces to the button top so that it can extend further down through some of the cloud fluff to make it easy to press.

4. Place the first layer onto the speakers and glue them in place with the button facing out.
The Second Layer

1. Place the second layer down over the backs of the two speakers and down to the back of the button. Glue this to the speakers as well.
2. Use double sided tape or a zip tie to hold the Metro M4 onto place between the speakers.
3. Connect one end of the DC power cable splitter to the 2.1mm DC jack on the Metro M4.
4. Connect the button wires to the following pins:
   1. Black wire to GND pin
   2. Yellow wire to A2 pin
The Third Layer

1. Place part of the third layer about half way down over the Metro M4 power and USB connector.
2. You may want to also connect a USB cable to the
Metro M4 so that your can program it later.

3. You can glue the stereo headphone jack down to make it easy to connect the audio cable later.

4. Connect the 3.5mm headphone jack assembly wires to the following pins:
   1. Black wire to GND pin
   2. Blue wire to A0 pin

5. Connect the Orange NeoPixel wire to the 2 Pin

6. Get all of the wires gathered towards the center and glue the other side of layer 3 to the speakers.

7. Connect the power connector for the speakers to the other end of the DC Barrel Jack Splitter.

8. Connect the speakers audio plug into the 3.5mm headphone jack.
The Top Layer

1. To make it easy to change the volume of the cloud, I cut a slot to fit part of the volume control through the fourth layer and glued it into place.
2. Feed the end of the DC Barrel Jack Splitter through the hole in the fourth layer along with the NeoPixel strip and the USB cable if that you may have connected.

3. Glue the fourth layer to the top of the speakers to hold it in place.
Adding LEDs

Now we will be wrapping the NeoPixel strip around the cloud. This cloud uses 150 NeoPixels but you could add more if you want it brighter or to make a larger cloud.

1. Start at one end of the cloud and as best you can, wrap the NeoPixel strip securing it with zip ties.
2. Continue wrapping the cloud with the LED strip trying to keep things as evenly spaced as you can securing with zip ties where needed.
3. When you get to the other end of the cloud, use another zip tie to hold the end of the LED strip in place.
4. You can use one final zip tie to make a loop that will let you hang the cloud up when finished.
Adding some fluffy stuff

Clouds are fluffy and so we will use some Poly-Fil polyester fiber and hot glue to make it happen. It is a good idea to go slow and use smaller pieces for this to ensure that everything holds together nicely.

The following video shows you how I put a bit of hot glue down on the cloud frame and apply a bit of the Poly-Fil over
and over again.

I tend to leave a bit on the top without Poly-Fil just so I can access the volume control and the DC plug.
Usage

The cloud will automatically connect to the WiFi network that you set in the secrets.py file. Once connected the cloud will pull weather data from the OpenWeather API and display the animation for your location's current forecast.

Changing the Weather

You can cycle through each of the weather animations by pressing the button at the bottom of your cloud. The animations start with sunny/clear, overcast, rain, thunderstorm, snow, and back to sunny.

Turning your cloud off

If you want to turn your cloud off for awhile, just press and hold the button at the bottom of the cloud for 4 seconds. The cloud will turn off and be ready to show the weather later.

Turn your cloud on again

Whenever you want to see the weather forecast, just press the button at the bottom of the cloud. This will look for the latest forecast and switch to that animation.

Going farther

This is a fun and entertaining project but there is plenty of room to make it better.

- Make your own sounds for the weather.
- Add some lightning wrangling air ships to your cloud.
- See if you can change the number of lights that turn blue based on how much rain the forecast predicts.
- Try to add an alert for severe weather or some other weather-based alert.
- Show that you know your stuff by connecting your cloud to Adafruit.io for full IoT control.