Want to make a Little Signal of your own?
You have come to the right place!

Here, we will walk you through how to make your own version of the Air object.
This project is intended to:

- Be approachable regardless of making experience
- Encourage you to use materials you might have around the house
- Get you started with working electronics+code that can be incorporated into a number of forms

We are going to make a connected fan with basic materials.
If you would like to take it a step further, the 3D file is available for Air and all the other objects, so you can integrate your own electronics.

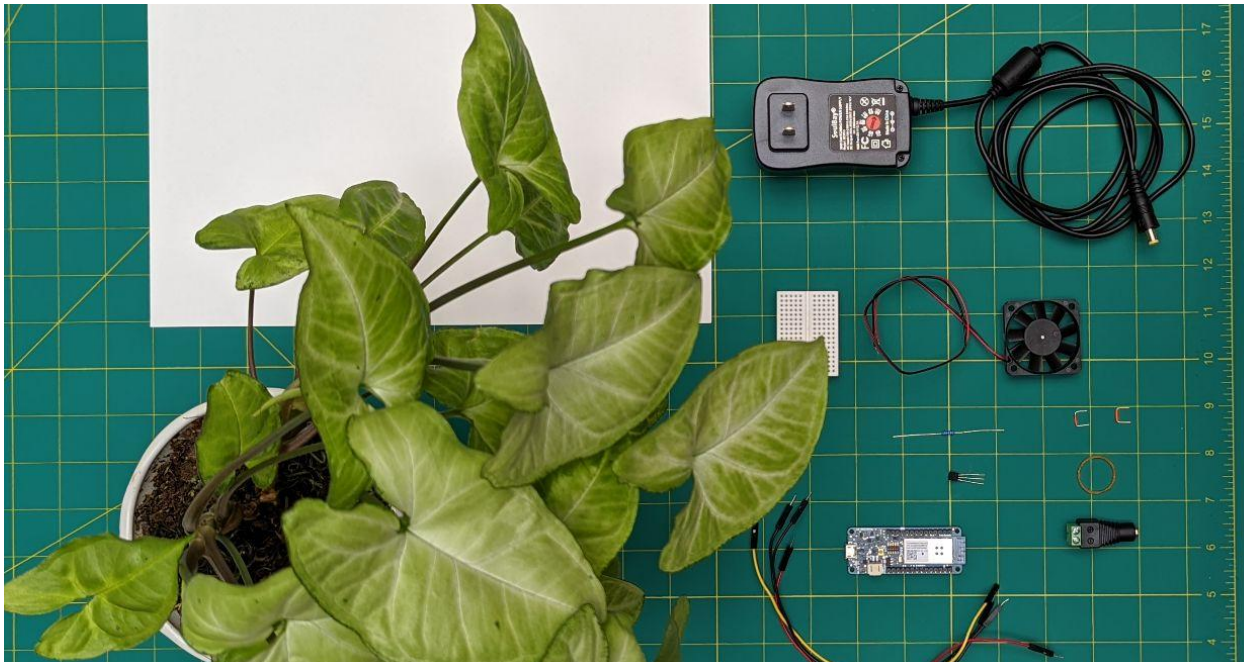
What you will need

Materials

- 12V computer fan
- Arduino MKR1000
- Transitor: NPN BC546B D6

- 220ohm resistor
- Jumper wires and connectors
- Small breadboard
- DC power supply receptacle
- 12v power supply

- Cardstock
- Rubber band
- 3M mounting tape
- A plant or other lightweight object to move



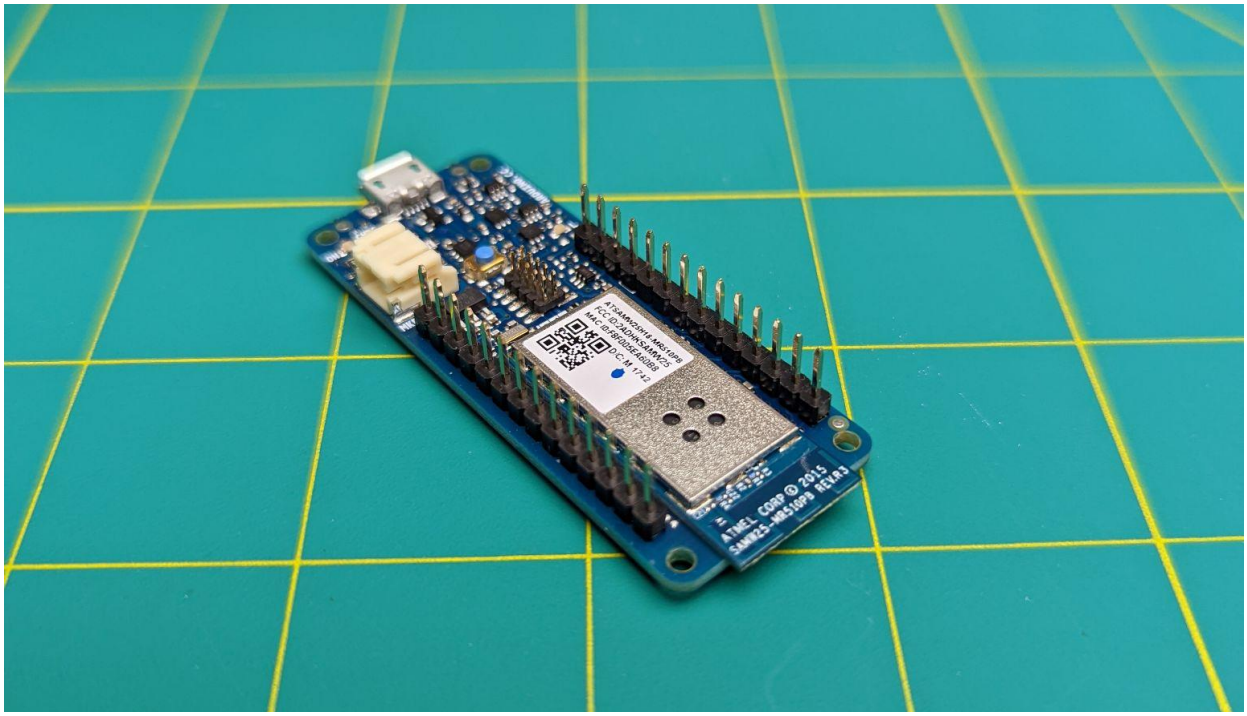
Tools

- Soldering iron
- Scissors
- Arduino IDE
- USB micro cable

Instructions

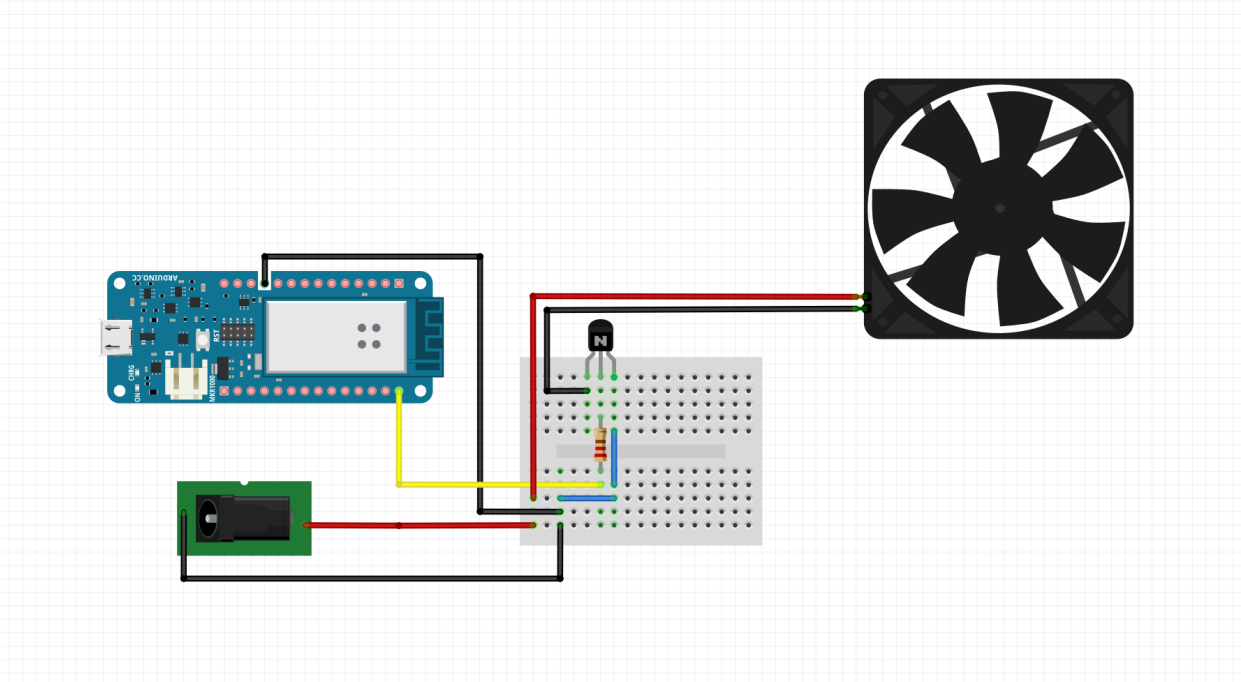
Prep the Arduino board

Solder the headers of your choice to the Arduino board if you don't already have some. You may find tips for soldering here: https://learn.adafruit.com/how-to-solder-headers



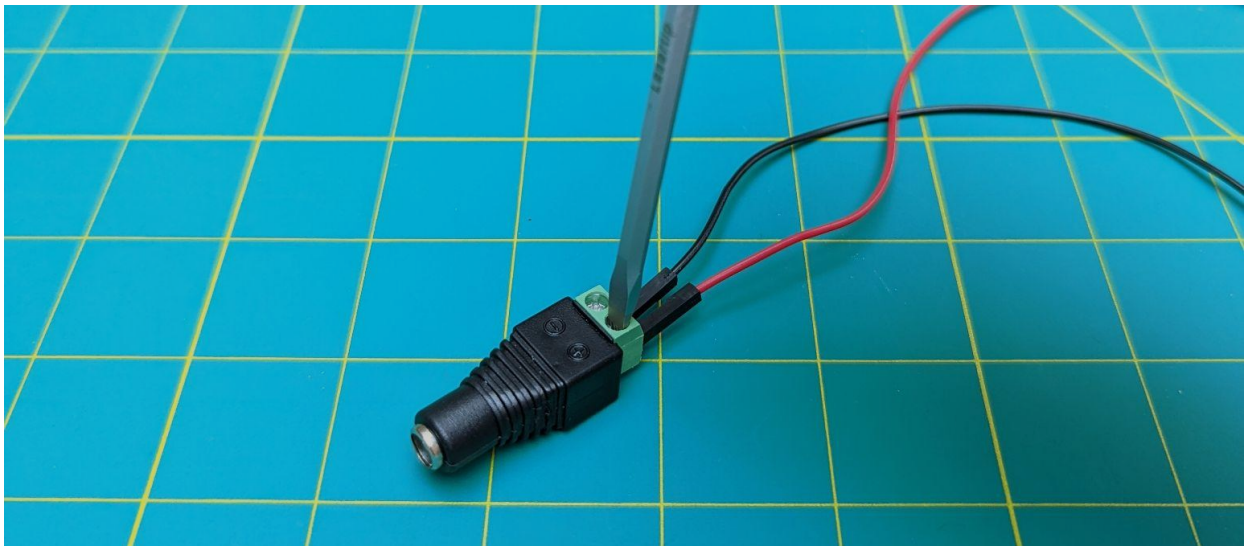
Assemble the components

Refer to this diagram for electronic assembly:

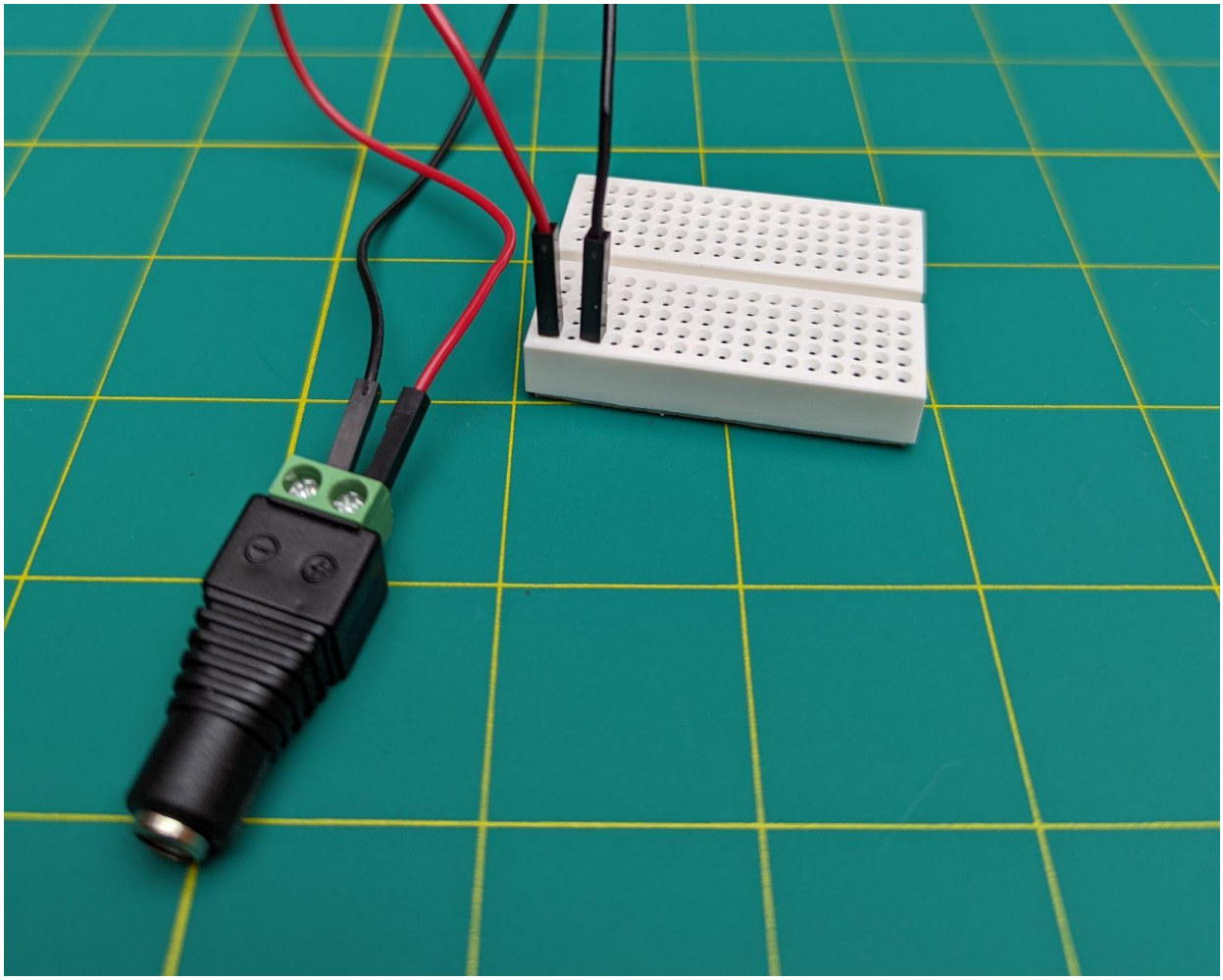


(diagram made with Fritzing)

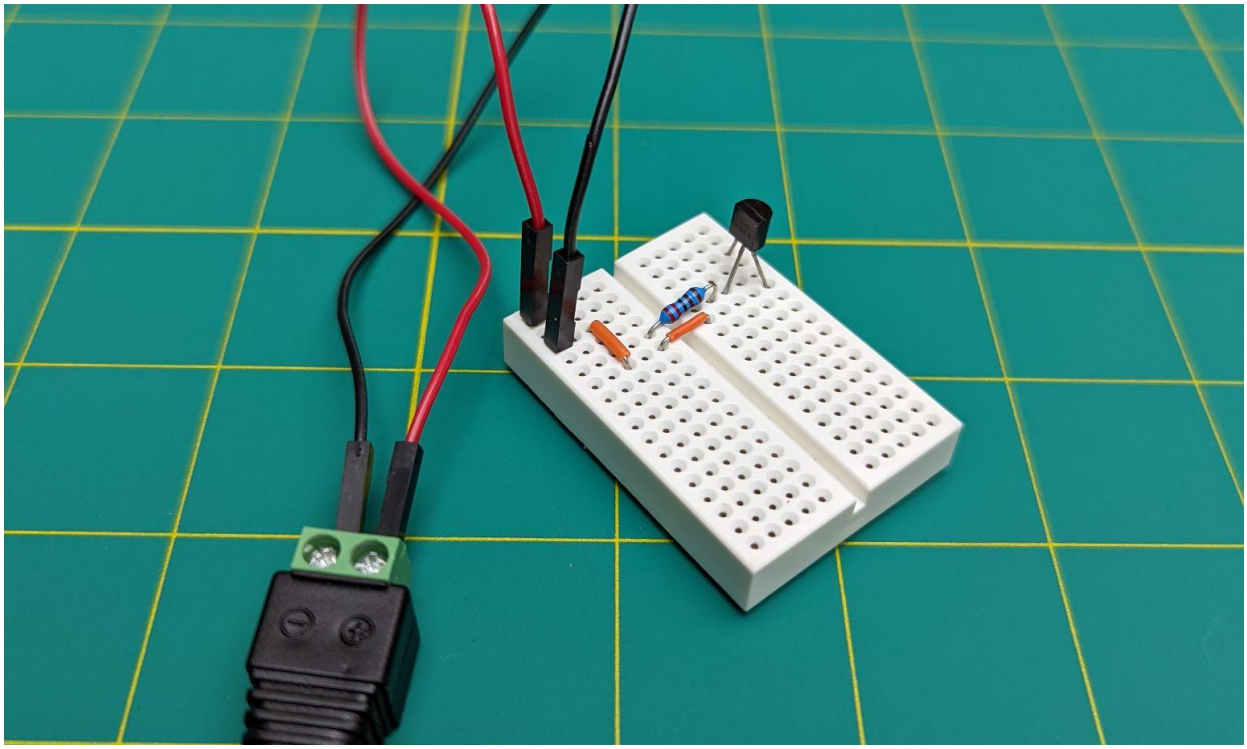
Screw connector wire into the DC power receptacle.



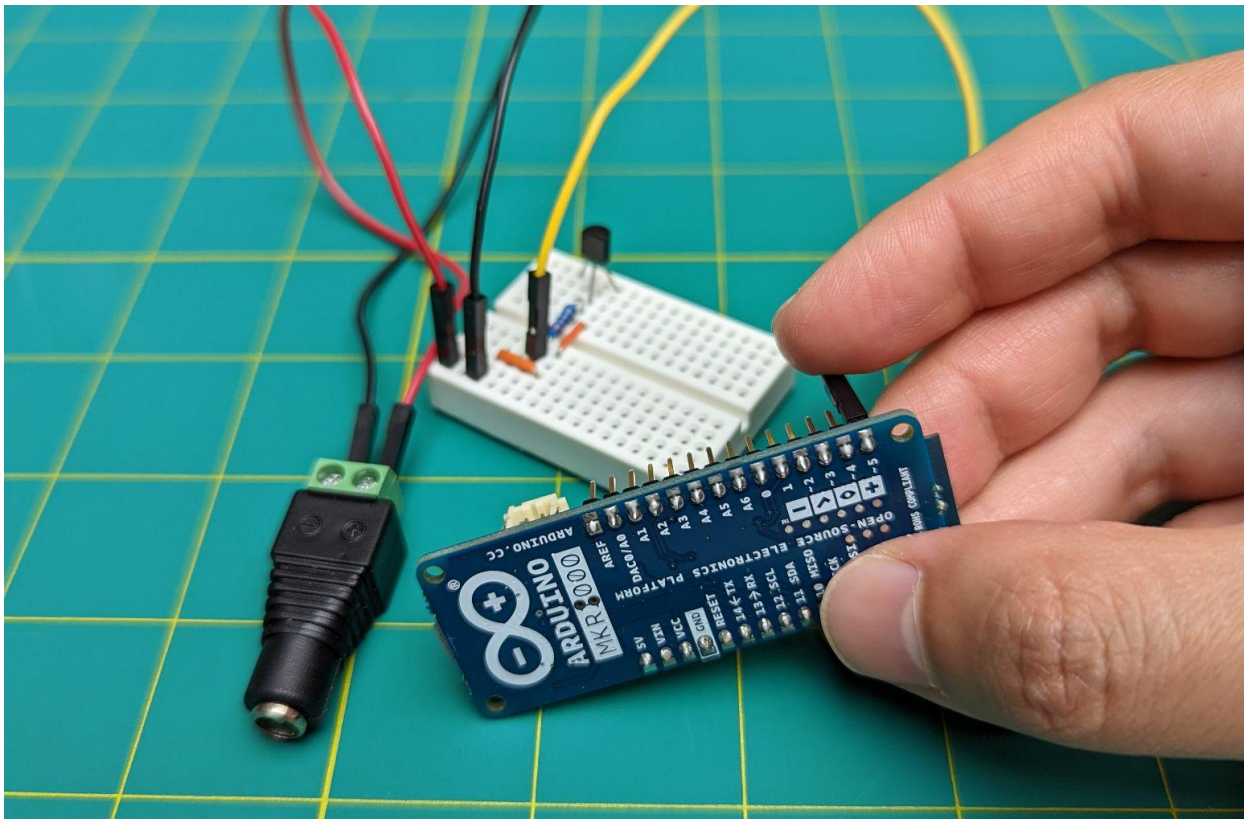
Connect these leads to the breadboard.



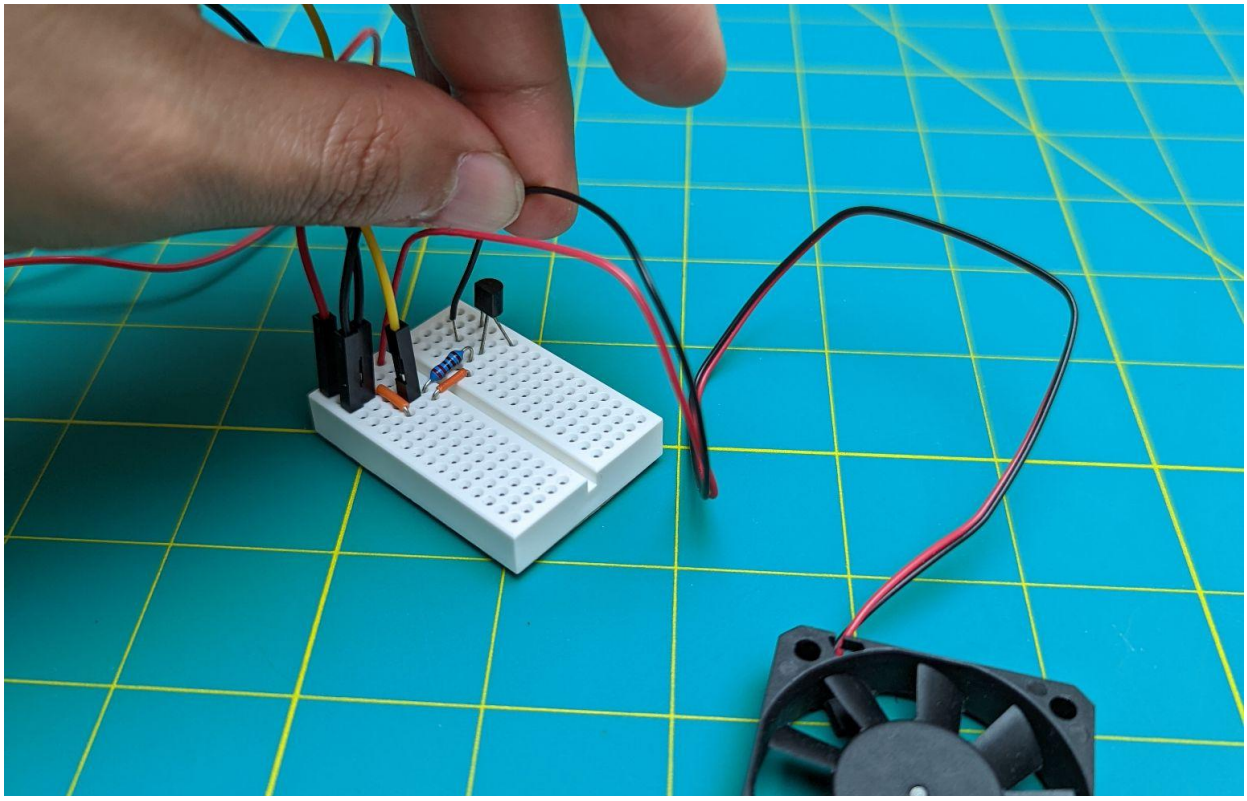
Place transistor, resistor on the breadboard. Use jumper wires to complete the circuit.

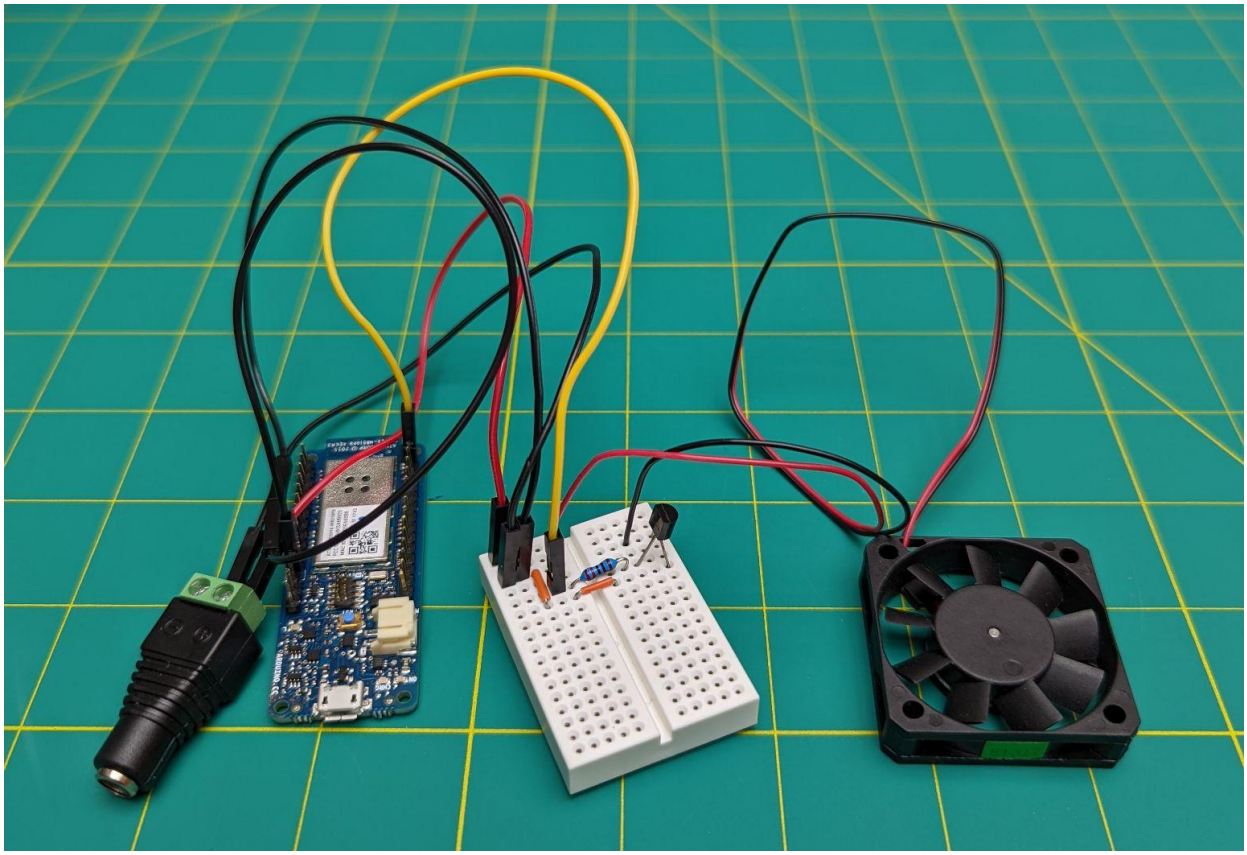


Connect the common GND on your breadboard to the GND pin on the Arduino.
Then connect the Base of the transistor to Arduino Digital Pin 5 as shown.



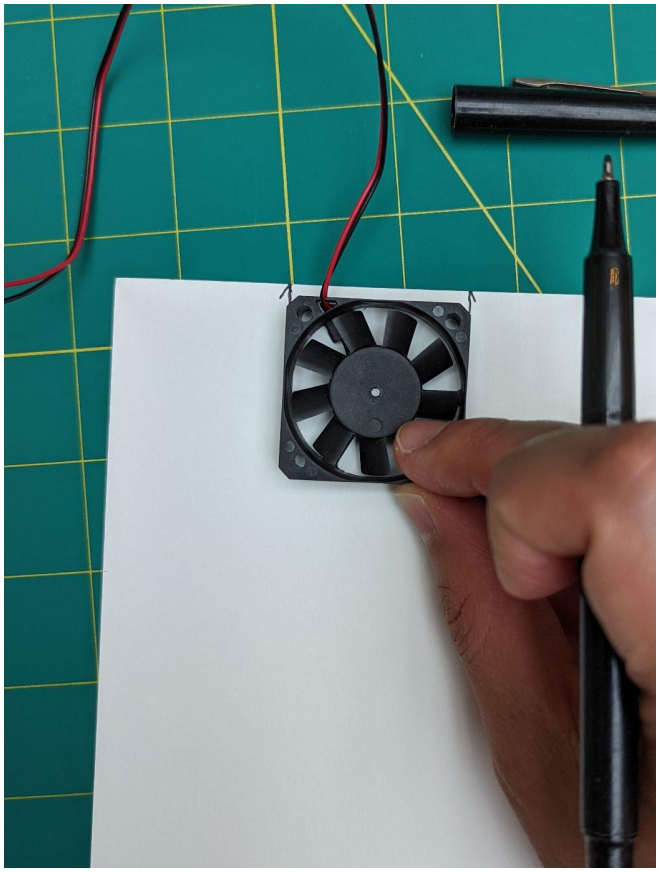
Connect the fan to the breadboard as shown in the diagram



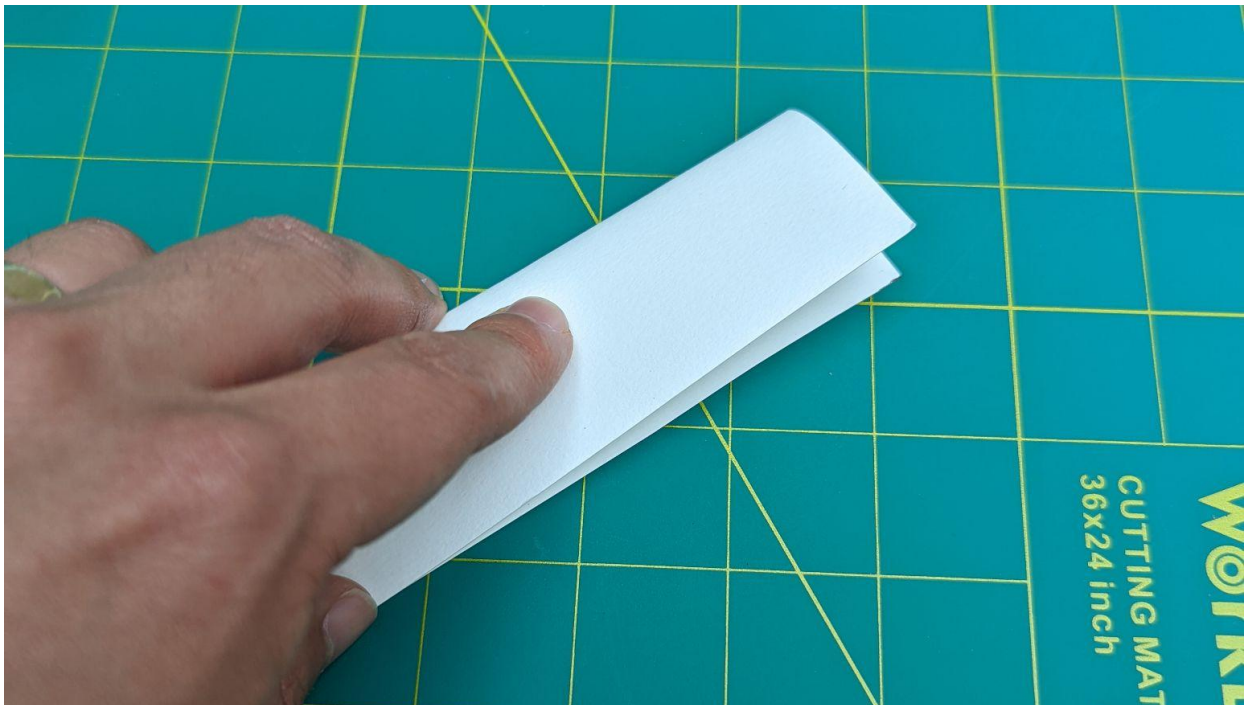


Mount the electronics

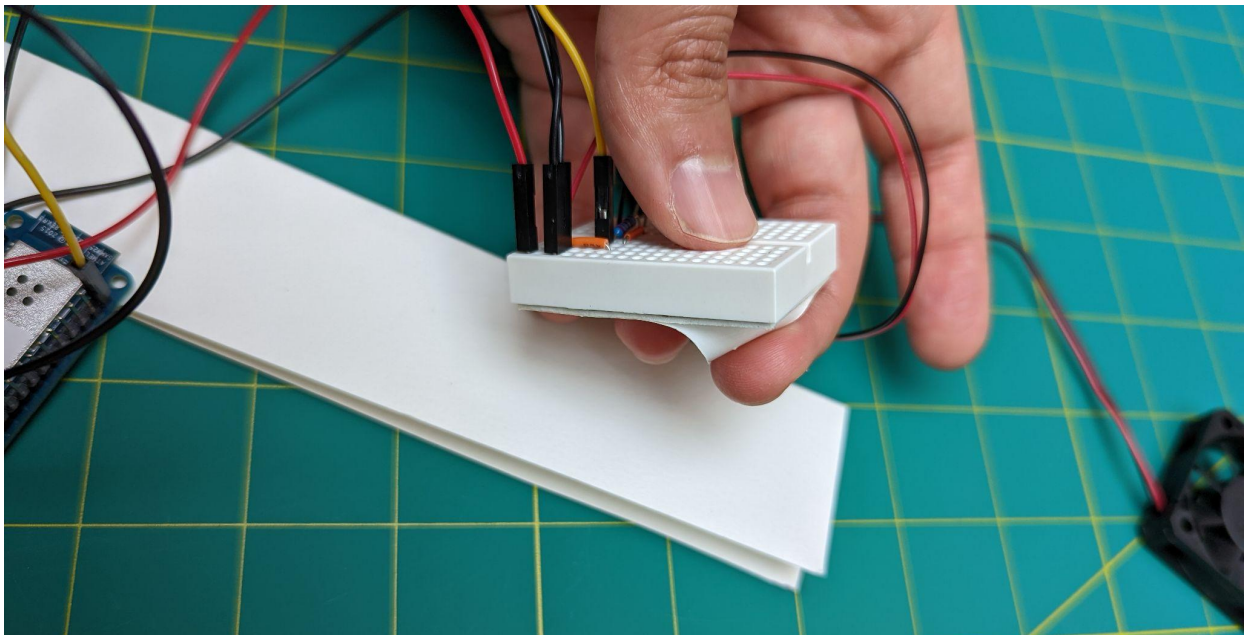
Cut cardstock in a strip approx. 2x the width of your fan



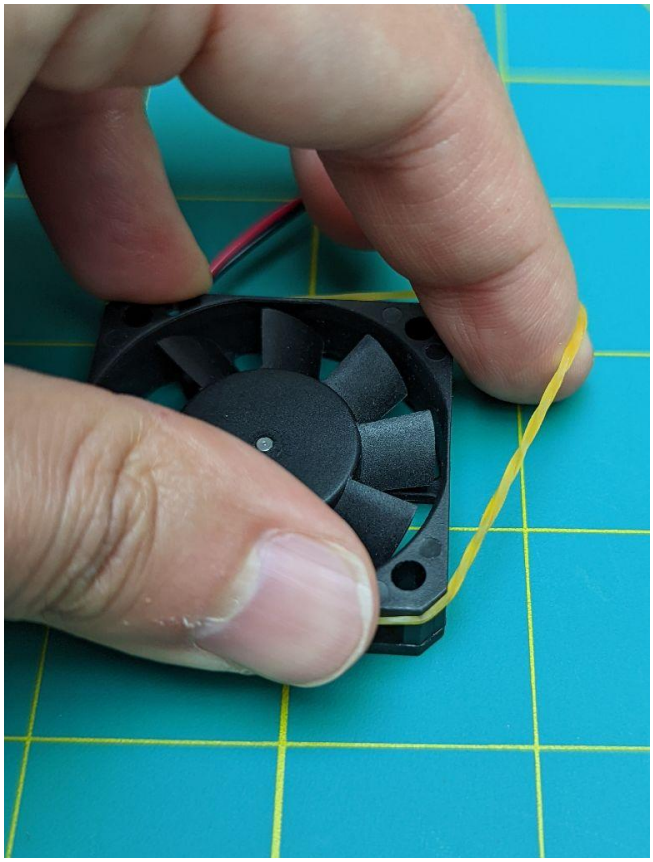
Fold it in half

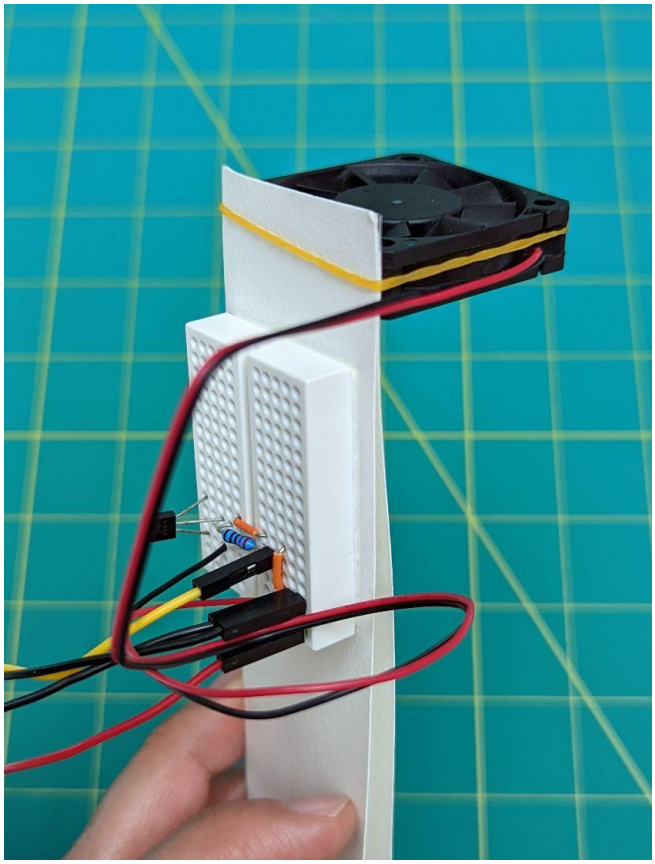


Mount the breadboard to one side of the cardstock



Slip the rubber band over the perimeter of the fan then slip the cardstock through the rubber band





Put your fan near a plant. We recommend a plant that likes a soft breeze. Choose a mounting method of your choice. We like putting the cardstock into a book, like a bookmark. This will allow you to adjust the height of your fan to accommodate plants of all sizes. You can also put it directly into the soil or tape it to the side of your plant pot, whatever works for your setup.

The Code: Setup

The sketch **little_signals_air.ino** is the code that you will upload to your arduino board to run the Little Signals experiment. It connects to your local wifi network and periodically queries a remote weather service for forecast data. It then analyzes the response for a specific weather

event (e.g. “Rain”) and if it detects that event, will briefly turn on a fan that is connected to the arduino.

Before you upload the code to your board, you’ll want to do a few things in the following sections:.

- **ArduinoJson**

```
#include <ArduinoJson.h>
```

    - Make sure you have installed this library via the IDE’s *Tools -> Manage Libraries*

- **Wifi101**

```
#include <WiFi101.h>
```

    - Make sure you have installed this library via the IDE’s *Tools -> Manage Libraries*
    - Make sure your firmware is updated to the latest version by following the instructions here.

- **NETWORK_SSID** and **NETWORK_PASSWORD**

```
char NETWORK_SSID[] = "YOUR_LOCAL_WIFI_NETWORK_SSID";
char NETWORK_PASSWORD[] = "YOUR_LOCAL_WIFI_NETWORK_PASSWORD”;
```

    - Set these to your wifi network’s ssid and password, respectively.

- **API_KEY**

```
String API_KEY = "YOUR_API_KEY";
```

    - Set this to your **openweathermap.org** api key. You can procure one here.

- **LOCATION**

```
String LOCATION = "Mountain View, CA";
```

- Set this to the city that you are interested in monitoring weather.

- **ALERT_TRIGGER**

```
String ALERT_TRIGGER = "Rain";
```

  - Set this to one of the possible choices, namely *"Thunderstorm, Drizzle, Rain, Snow, Mist, Smoke, Haze, Dust, Fog, Sand, Ash, Squall, Tornado, Clear, or Clouds"* .
  - This is the weather event that when detected in the forecast will turn on the connected fan for a brief moment.

- **DELAY_BETWEEN_WEATHER_QUERIES**

```
int DELAY_BETWEEN_WEATHER_QUERIES = 5000;
```

  - Set this to how long you want to wait between queries to the weather service (in milliseconds).
  - Note that if a weather trigger event is detected, then this delay will start after the fan is turned on and then off.

- **FAN_ON_DURATION**

```
int FAN_ON_DURATION = 5000;
```

  - Set this to the duration you want the fan to run for if a weather trigger event is detected in the forecast (in milliseconds).

Now upload the sketch to your arduino and enjoy your little signals experiment!

The Code: Explained

If you're curious about how the code works, we dive into each section below.

- **setup**

```
void setup() {
 initSerial();
```

```
 initWifi();
 initFan();
}
```

- As with all arduino sketches, when it first runs, this method is called.
- We initialize serial communication so that we can print debug statements, initialize the wifi connection, and get the fan ready to be turned on and off.
- Let's go through each method in turn.

- **initSerial**

```
void initSerial() {
 Serial.begin(9600);
 while (!Serial) {
   yield();
 }
 Serial.println("Serial is READY!");
}
```

  - We specify the baud rate we want to communicate over and then wait for it to be ready before continuing.

- **initWifi**

```
void initWifi() {
 Serial.println("Initializing Wifi...");

 // check for the presence of the shield:
 if (WiFi.status() == WL_NO_SHIELD) {
   Serial.println("WiFi shield not present, cannot initialize Wifi");
   return;
 }

 while ( status != WL_CONNECTED) {
   Serial.println("Trying to connect to SSID " + String(NETWORK_SSID) +
"...");
   status = WiFi.begin(NETWORK_SSID, NETWORK_PASSWORD);
 }
 Serial.println("Wifi connected!");
 IPAddress ipAddress = WiFi.localIP();
```

```
 Serial.println("Arduino IP address is ");
 Serial.println(ipAddress);
}
```

- First we check if the arduino board has wifi capability and if not, we exit. If it does, we try to connect to the given wifi network and wait until it completes before continuing.
- Once it connects we also print out the IP address of the connected board.

- **initFan**

```
void initFan() {
 pinMode(FAN_PIN, OUTPUT);
}
```

- We set the pin mode of the fan to be *OUTPUT* so we can turn it on and off easily.

- **loop**

```
void loop() {
 String response = queryWeather();
 if (response != "") {
   reactToWeatherResponse(response);
 }

 delay(DELAY_BETWEEN_WEATHER_QUERIES);
}
```

- Now that we've completed the setup method, the board will call this method repeatedly. Let's go through that logic now.

- **queryWeather**

```
String queryWeather() {

 String response = "";
```

```
 if (wifiClient.connected() == 0) {
   Serial.println("Connecting to weather server...");
   if (wifiClient.connect(SERVER_ADDRESS, SERVER_PORT)) {
     Serial.println("connected to weather server!");
   } else {
     Serial.println("Couldn't connect to weather server");
   }
 }

 Serial.println("Querying weather...");

 wifiClient.print("GET /data/2.5/forecast?");
 wifiClient.print("q=" + LOCATION);
 wifiClient.print("&cnt=1");
 //wifiClient.print("&units=metric");
 wifiClient.print("&units=imperial");
 wifiClient.println("&appid=" + API_KEY);
 wifiClient.println("Host: api.openweathermap.org");
 wifiClient.println("Connection: close");
 wifiClient.println();


 String line = "";
 while (wifiClient.connected()) {
   line = wifiClient.readStringUntil('\n');
   response = response + line;
 }
 Serial.println(response);

 return response;
}
```

- Here, we query the weather service for a forecast.
- We first check if the wifi client is connected to the weather server, and if it is not, we initiate the connection.
- Then we begin the *HTTP GET* request in earnest. In this request, we specify the URL path and the query string where we specify the location, units, api key and host. And then we harvest the response as it comes in from the server.

- **reactToWeatherResponse**

```
void reactToWeatherResponse(String response) {
 int responseArrayLength = response.length() + 1;
 char responseArray[responseArrayLength];
 response.toCharArray(responseArray, responseArrayLength);
 DynamicJsonDocument dynamicJsonDocument(1024);
 deserializeJson(dynamicJsonDocument, responseArray);
 const char* soonestWeatherPeriodForecast =
dynamicJsonDocument["list"][0]["weather"][0]["main"];

 if (String(soonestWeatherPeriodForecast) == ALERT_TRIGGER) {
   Serial.println("ALERT TRIGGER DETECTED!!");
   respondToAlertTrigger();
 }
}
```

    - Once we have the response, we call this method. Here we convert the response to a *JSON* object and extract the most recent forecast. If this forecast matches our weather trigger, then we call *respondToAlertTrigger*.

- **respondToAlertTrigger**

```
void respondToAlertTrigger() {
 Serial.println("Responding to alert trigger...");
 brieflyTurnOnFan();
}
```

    - Here, we call the method to briefly run the fan.

- **brieflyTurnOnFan**

```
void brieflyTurnOnFan() {
 Serial.println("Briefly turning on fan...");
 digitalWrite(FAN_PIN, HIGH);
 delay(FAN_ON_DURATION);
 digitalWrite(FAN_PIN, LOW);
}
```

    - We set the fan pin to *HIGH* to turn on the fan
    - We wait via delay for the time specified
    - We set the fan pin to *LOW* to turn off the fan

- **delay**
    - We instruct the loop to wait for a given duration before it repeats the cycle again

Further experimentation

To build the original Air object, you can download the 3D file for this Little Signal and try to integrate your own electronics. The 3D files for the other five objects - Button, Movement, Rhythm, Shadow, and Tap - are also available if you want to play around with them all.

Here are some ideas for those other objects:

Tap
Try replacing the actuation of a fan with that of a servo motor. You can do so by replacing the brieflyTurnOnFan function called out in the respondToAlertTrigger function in the code. Instead of triggering the fan to briefly turn on, you can create a new function that triggers a servo motor to move a certain number of degrees. You can mount a small teaspoon to your motor to create the tapping arm of the object.

Shadow
Play with Shadow by utilizing the same servo and modified code from the Tap example. Simply find something round to mount slightly off-center to the top of your servo motor. In effect, this is a cam that will translate to non uniform circular motion that can create subtle movement of shadow cast from the object.

Movement
Swap in a tiny 6V linear actuator to replace the fan. With the 6V actuator you can control its position using just the arduino board via the 5V pin, ground pin, and a PWM pin (no 12V power supply needed). Treat the actuator as a servo where bigger and smaller PWM values manipulate the extension of the actuator. When the weather event is triggered, use the PWM pin to extend the actuator. When no weather event is triggered, use the PWM pin to retract the actuator. This mechanic simulates the peg movement seen in the movement object.

Button

You can repurpose the linear actuator from the Movement example for this one. Try swapping the trigger event from a weather event/threshold to an accumulative metric, such as precipitation. You can use that data for your region and map its value to the extension of the linear actuator so you can see relatively how much rain has fallen.

Rhythm

Connect a piezo buzzer to the ground pin and a digital pin on the arduino (note: you may need a resistor for this circuit depending on your buzzer). When the weather event is detected, call a function that will successively call the built-in tone function. In this function, you can adjust frequency and the delay between calls to tone, to modulate the rhythm of the audio produced. More urgent or extreme weather events might call for more rapid rhythms while calmer weather events might call for slower rhythms, or none at all.

References

1. https://nerd-corner.com/arduino-fan-controller/
2. https://fritzing.org/