Raspberry Pi: Plant Shielding System

by CynChee

This project aims to create a plant shielding system by activating a cover attached to a motor servo when the light exposure exceeds a certain threshold (i.e there is too much sunlight). It will also activate an LED and Buzzer to notify surrounding people about this.

Supplies:

Raspberry Pi

Micro Servo 9g

BH1750 Ambient Light Sensor

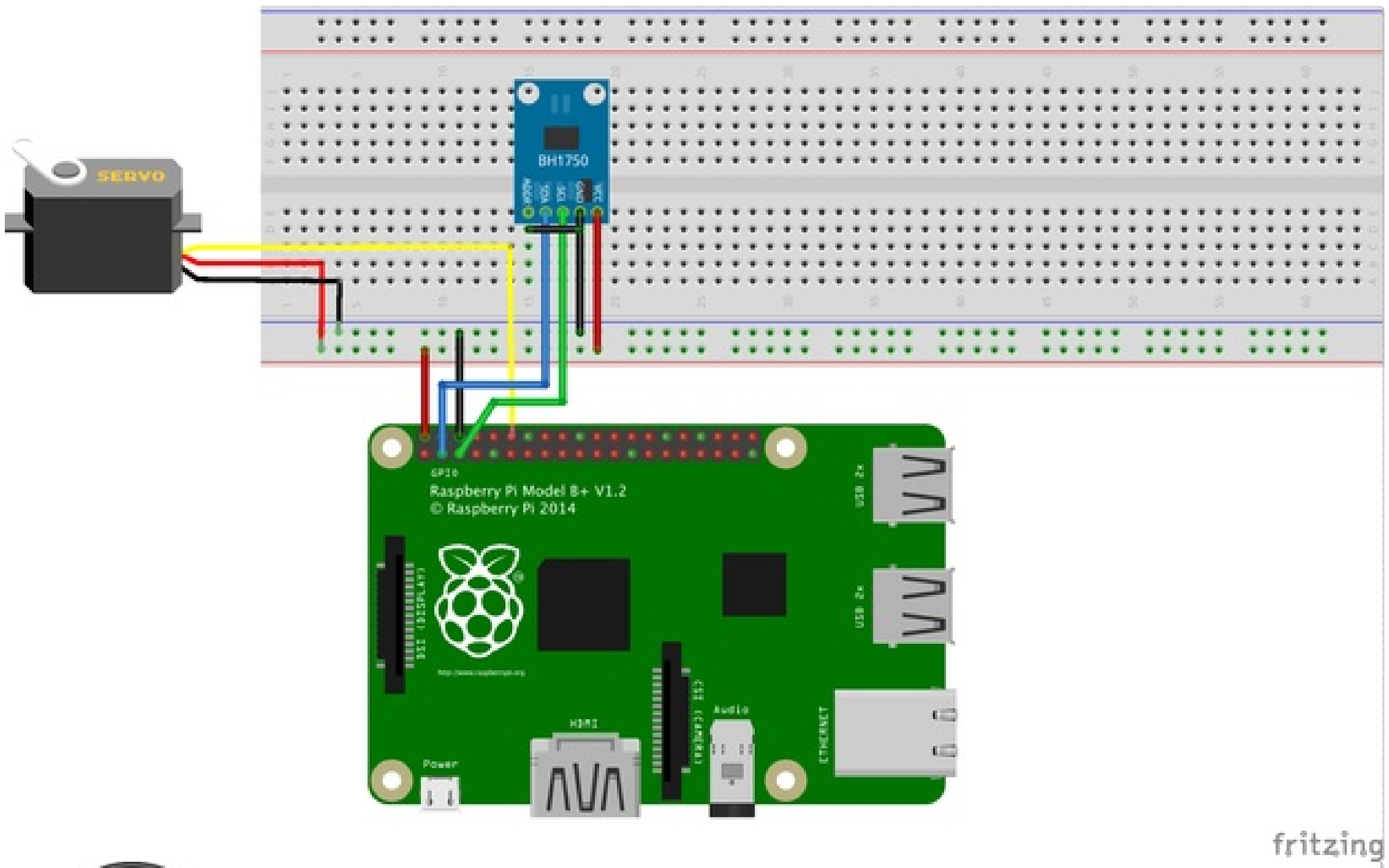
LED

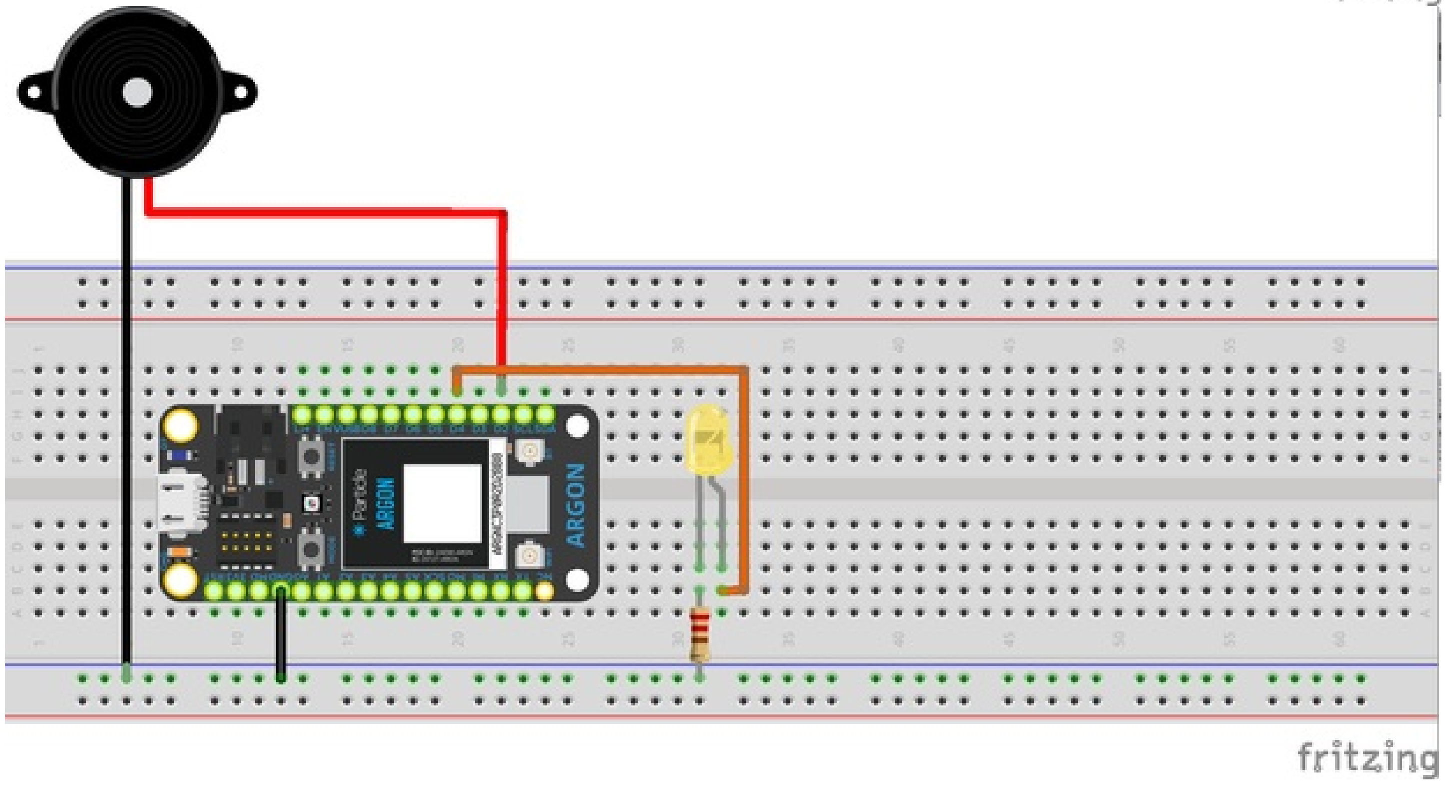
Piezo Buzzer

Resistors 220 Ohm

Particle Argon

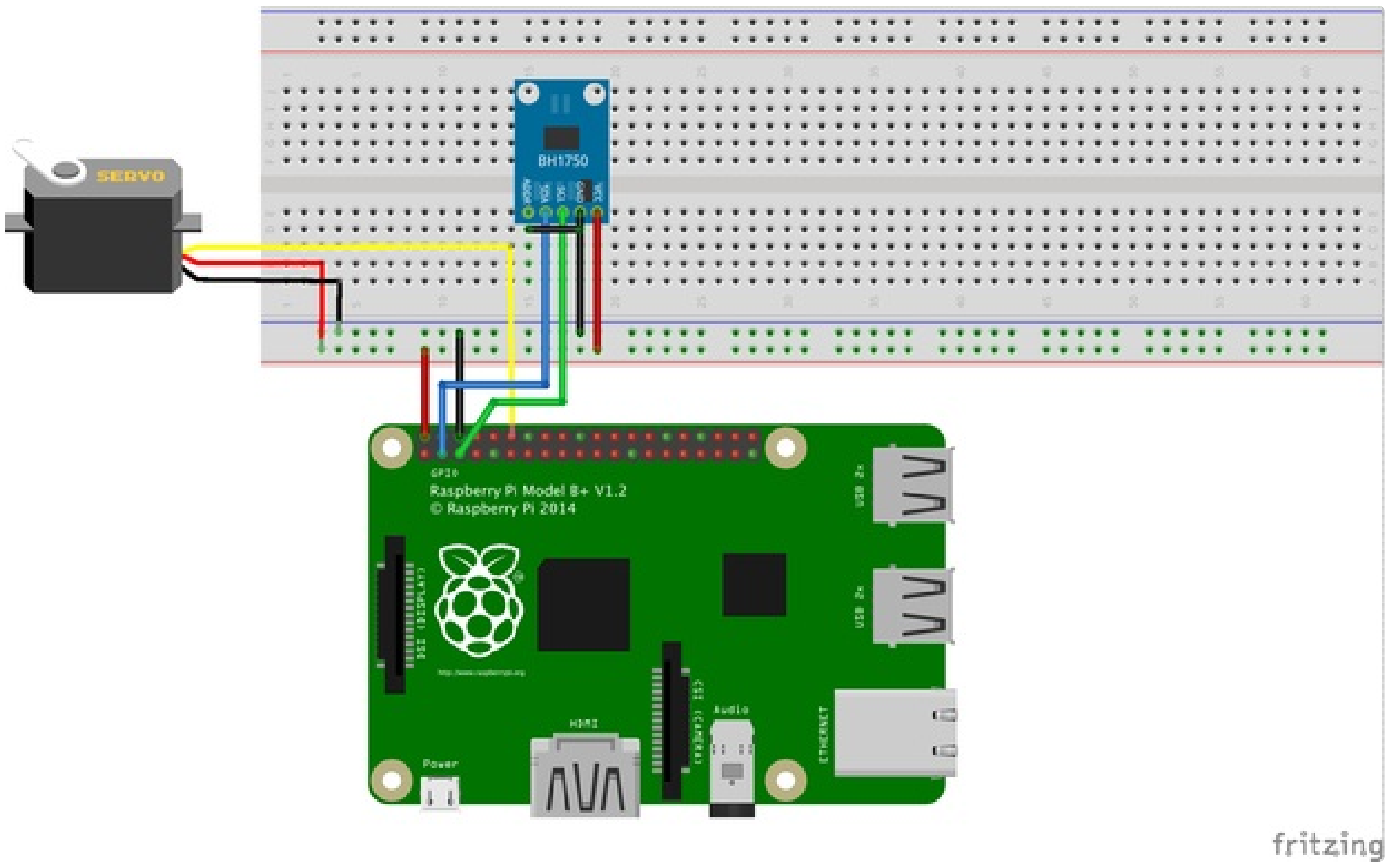
Jumper Wires





Step 1: Setup for Raspberry Pi System

Set up the equipment as provided in the diagram. Connect pin 2 (5V) and pin 6 (GND) of the Raspberry Pi to the breadboard.



Step 2: Raspberry Pi Connections

For the servo, connect the VCC to the power supply, then GND. The signal pin is then connected to pin 12 (GPIO18).

For the BH1750 Ambient Light Sensor, the connections are as follows:

GND to pin 6 (GND)

ADD (connected horizontally to the GND pin of the sensor)

SDA to pin 3 (GPIO2)

SCL to pin 5 (GPIO5)

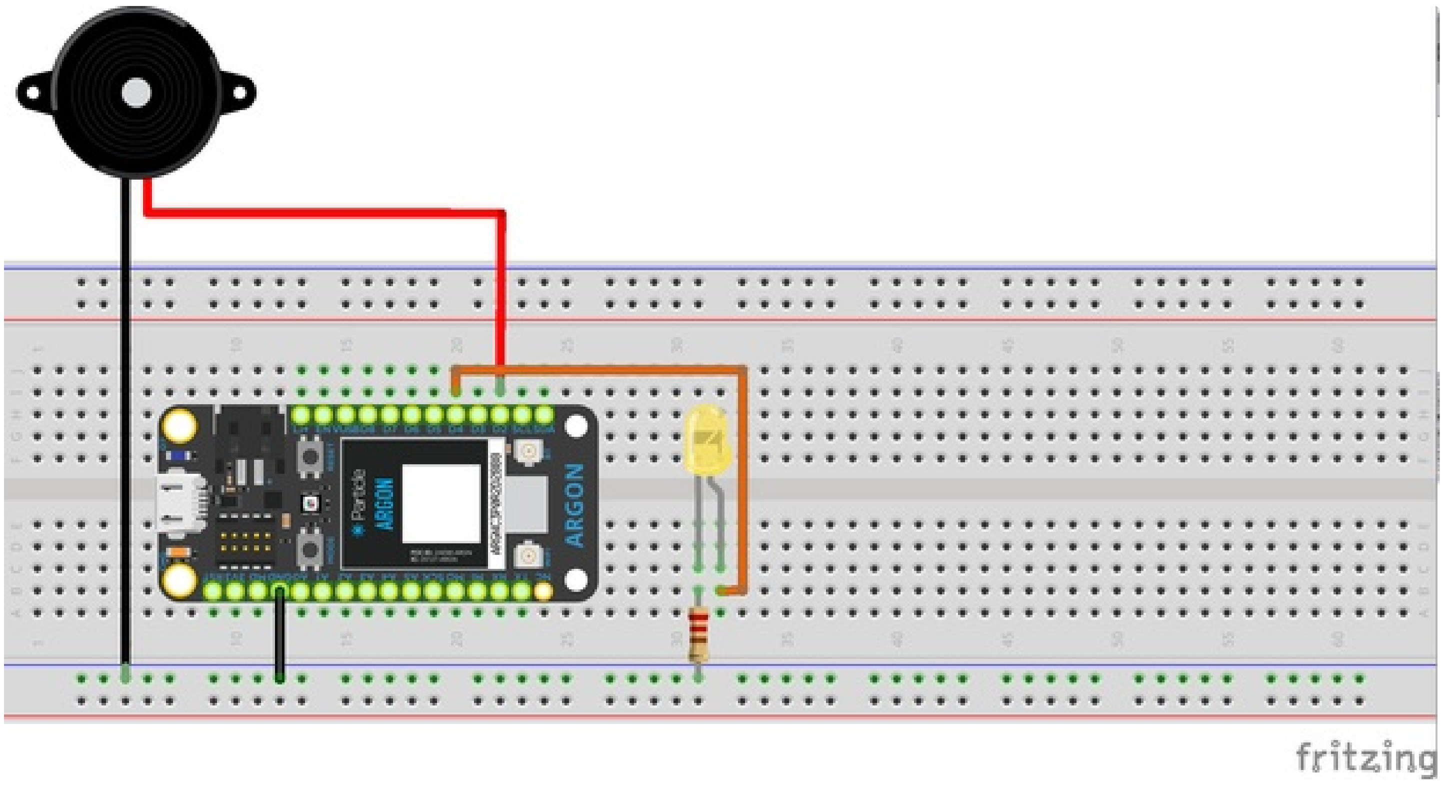
VCC to pin 2 (VCC)

Step 3: Setup and Connections for Particle Argon System

For the Particle Argon, connect the GND pin to the breadboard.

For the LED, the cathode (negative terminal - shorter leg) is connected to a 220 ohm resistor which is then connected to GND. The anode (positive terminal - longer leg) is connected to pin D4.

The buzzer has its negative terminal connected to GND, and the positive terminal connected to pin D2.



Step 4: Setup Webhooks

In this project, we want the Raspberry Pi to communicate with the Particle Argon since these two systems will be placed separately. In order to do this, we want the RPi to send HTTP posts requests via Webhooks service, which then publishes an event via IFTTT.

The code syntax and how to find our Webhooks key can be found in the Webhooks service FAQ here.

Step 5: Setup IFTTT

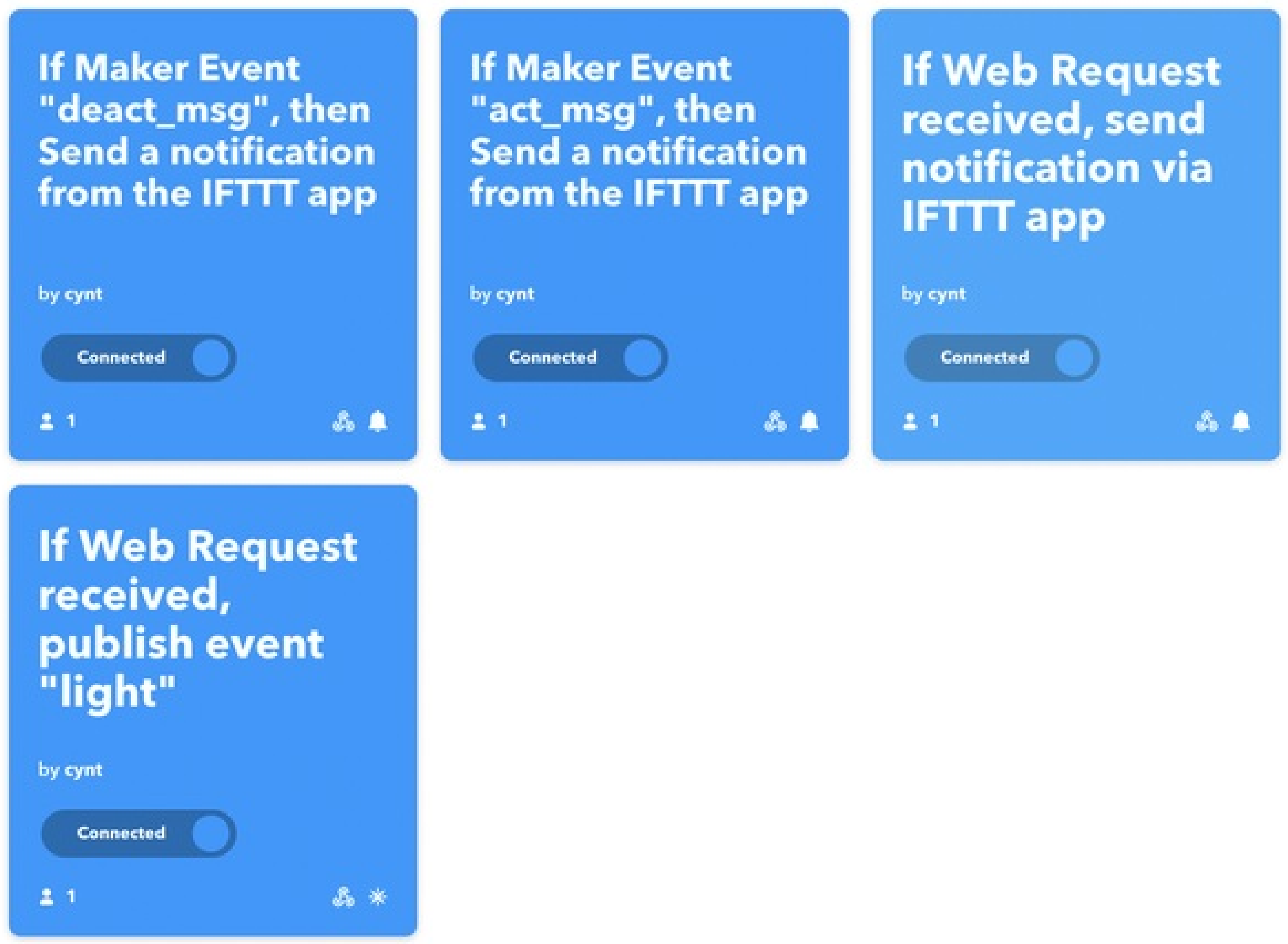
Firstly, in order to use IFTTT you need to create an IFTTT account. Once you have done that, you can add new applets. In this project, the applets created are as follows:

IF (Webhooks -> Receive a web request) THEN (Particle -> Publish an event)

IF (Webhooks -> Receive a web request) THEN (Notifications -> Send a notification from the IFTTT app)

The first applet enables the RPi to send web requests via Webhook and the second applet enables the Particle Argon to receive the request, then publish an event which it detects, triggering the LED and Buzzer.

When a web request is received, it also automatically sends a notification via the IFTTT app. To use this functionality, ensure the IFTTT app is installed on your mobile phone and notifications are enabled.



If Maker Event "deact_msg", then Send a notification from the IFTTT app

by cynt

Connected

1

If Maker Event "act_msg", then Send a notification from the IFTTT app

by cynt

Connected

1

If Web Request received, send notification via IFTTT app

by cynt

Connected

1

If Web Request received, publish event "light"

by cynt

Connected

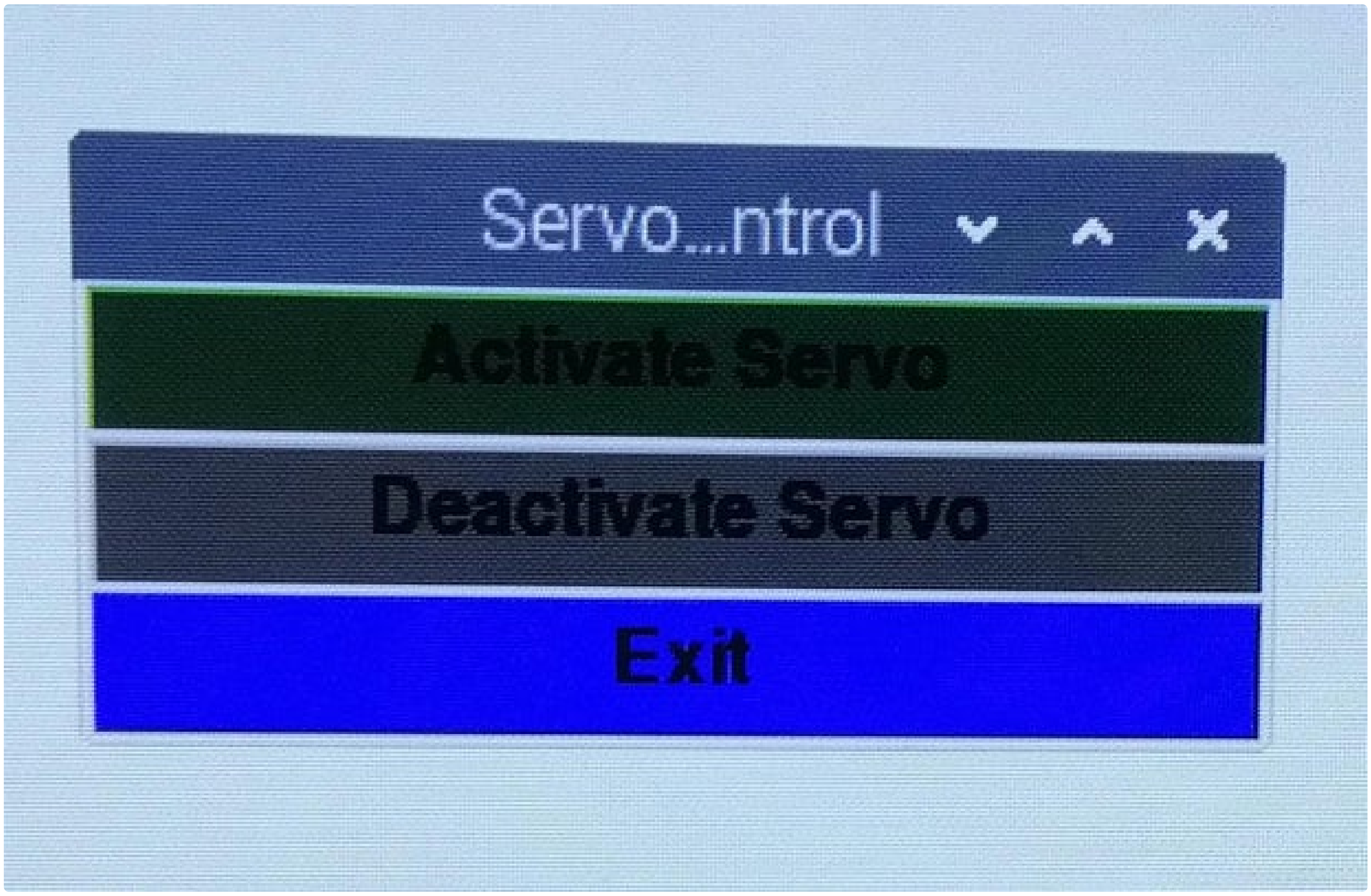
1

Step 6: Coding for Raspberry Pi

```
#this code is for detecting light intensity and activating the servo is it exceeds the threshold
#it also sends web request when this happens

#importing libraries
import RPi.GPIO as GPIO
from gpiozero import Servo
from time import sleep
import smbus
import requests
DEVICE = 0x23
RES = 0x20
bus = smbus.SMBus(1)

#posts an HTTP request to Webhook
def light():
    requests.post('https://maker.ifttt.com/trigger/light/with/key/b2KC93lYIFpBaCZ0hOnhQz', params={"value1":"none","value2":"none","value3":"none"})

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

##pin setup
GPIO.setup(18, GPIO.OUT)
servo = GPIO.PWM(18,500)
servo.start(0)

try:
    #stores previous light reading
    prevdata = 0
    while True:
        data = bus.read_i2c_block_data(DEVICE, RES)
        data = int((data[0] << 8) + data[1])
        print(data)
        #if current light level > threshold and previous < threshold
        if data > 500 and prevdata < 500:
            level = 'Too bright. Activating Servo'
            light()
            for dc in range(50,101,5):
                servo.ChangeDutyCycle(dc)
                sleep(0.5)

        #if light is now dark below threshold
        if data < 500 and prevdata > 500:
            print('Deactivating servo')
            for dc in range(100,49,-5):
                servo.ChangeDutyCycle(dc)
                sleep(0.5)

        elif data < 500:
            level = 'Too dark'

        print(level)
        prevdata = data #update light reading
        sleep(5)

except KeyboardInterrupt:
    print("Program stopped")
```

Step 7: Creating a GUI in Raspberry Pi

We also want to implement a Graphical User Interface in Raspberry Pi which allows us to manually activate and deactivate the servo. It also sends notifications via IFTTT the same way it does when the servo is activated due to light readings. An example of how the GUI would look is shown in the picture (excuse the quality).



Step 8: Code for GUI in Raspberry Pi

```
#import libraries
from tkinter import *
import tkinter.font
import RPi.GPIO as GPIO
from time import sleep
import requests

#setup board and pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
servo = GPIO.PWM(18,500)
servo.start(0)

#create window
win = Tk()
win.title("Servo Control")
myFont = tkinter.font.Font(family = "Helvetica", size = 12, weight = "bold")

#posts HTTP requests to Webhook
def act_msg():
    requests.post('https://maker.ifttt.com/trigger/act_msg/with/key/b2KC93lYIFpBaCZ0hOnhQz', params={"value1":"none","value2":"none","value3":"none"})

def deact_msg():
    requests.post('https://maker.ifttt.com/trigger/deact_msg/with/key/b2KC93lYIFpBaCZ0hOnhQz', params={"value1":"none","value2":"none","value3":"none"})

#activates servo manually
def activate():
    act_msg()
    for dc in range(50,101,5):
        servo.ChangeDutyCycle(dc)
        sleep(0.5)

#deactivates servo manually
def deactivate():
    deact_msg()
    for dc in range(100,49,-5):
        servo.ChangeDutyCycle(dc)
        sleep(0.5)

#closes window
def close():
    win.destroy()

#Buttons
activateButton = Button(win, text = "Activate Servo", font = myFont, command = activate, bg = 'green', height = 1, width = 24)
deactivateButton = Button(win, text = "Deactivate Servo", font = myFont, command = deactivate, bg = 'grey', height = 1, width = 24)
exitButton = Button(win, text = "Exit", font = myFont, command = close, bg = 'blue', height = 1, width = 24)

activateButton.grid(row = 0,column = 1)
deactivateButton.grid(row = 1, column = 1)
exitButton.grid(row = 2, column = 1)
```

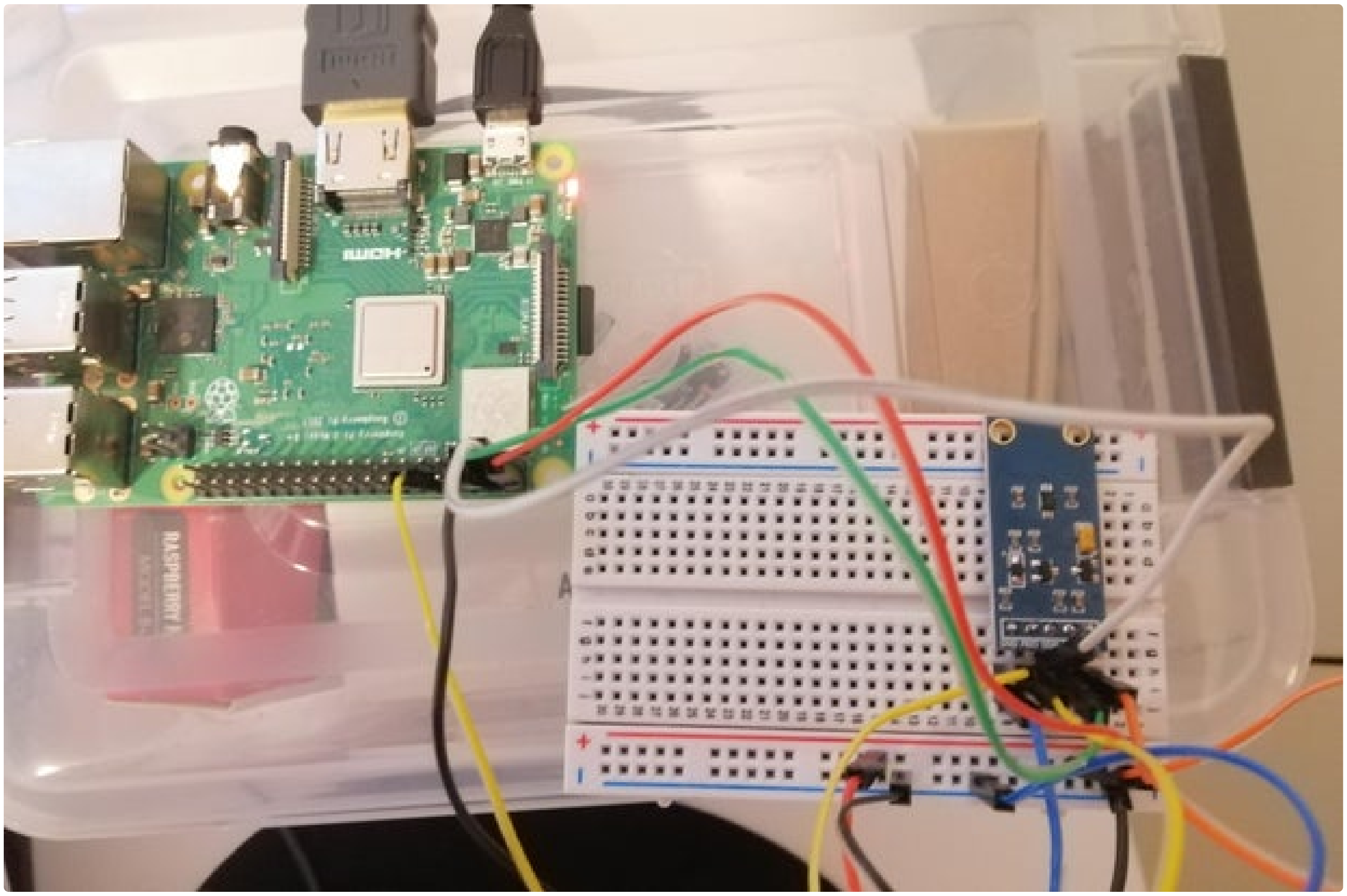
Step 9: Code for Particle Argon

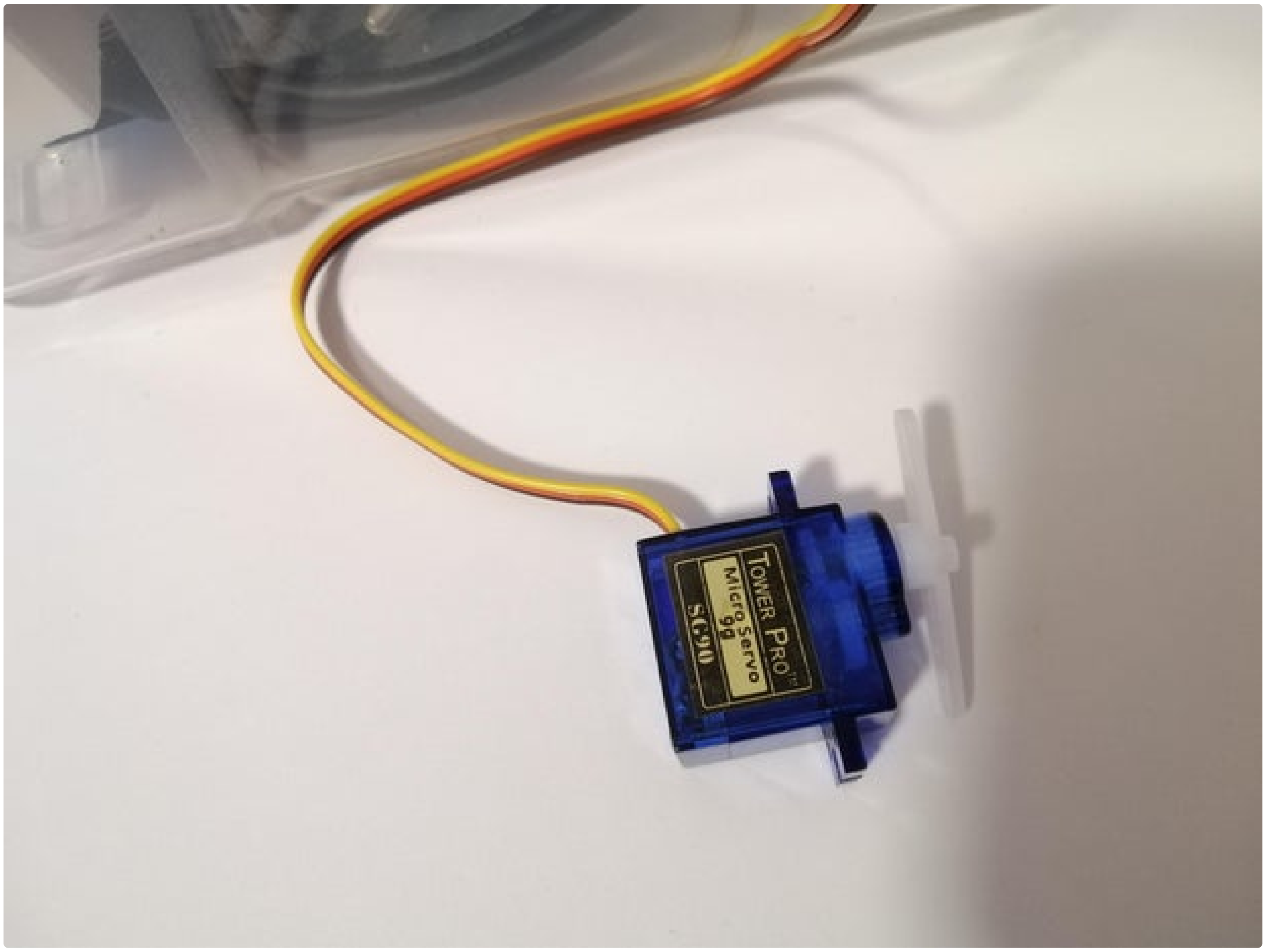
```
# activates the LED and Buzzer whenever the event "light" is published, which is done when it receives
# web requests from the Raspberry Pi

int led = D4;
int buzzer = D2;

// The following line is optional, but recommended in most firmware. It
// allows your code to run before the cloud is connected. In this case,
// it will begin blinking almost immediately instead of waiting until
// breathing cyan,
SYSTEM_THREAD(ENABLED);

// The setup() method is called once when the device boots.
void setup()
{
 pinMode(led, OUTPUT);
 pinMode(buzzer, OUTPUT);
 Particle.subscribe("light", alert, ALL_DEVICES); // subscribes to the event "light" to alert when it is published
}

// blinks the LED
void blink_led()
{
    digitalWrite(led, HIGH);
 delay(500);
 digitalWrite(led, LOW);
 delay(500);
}

// activates the buzzer
void buzzer_sound()
{
    tone(buzzer, 2000, 500);
    delay(1000);
}

// alerts the user by blinking the LED and playing the buzzer
void alert(const char *event, const char *data)
{
    blink_led();
    blink_led();
    blink_led();
    buzzer_sound();
    buzzer_sound();
    buzzer_sound();
}

// The loop() method is called frequently.
void loop()
{
}
```

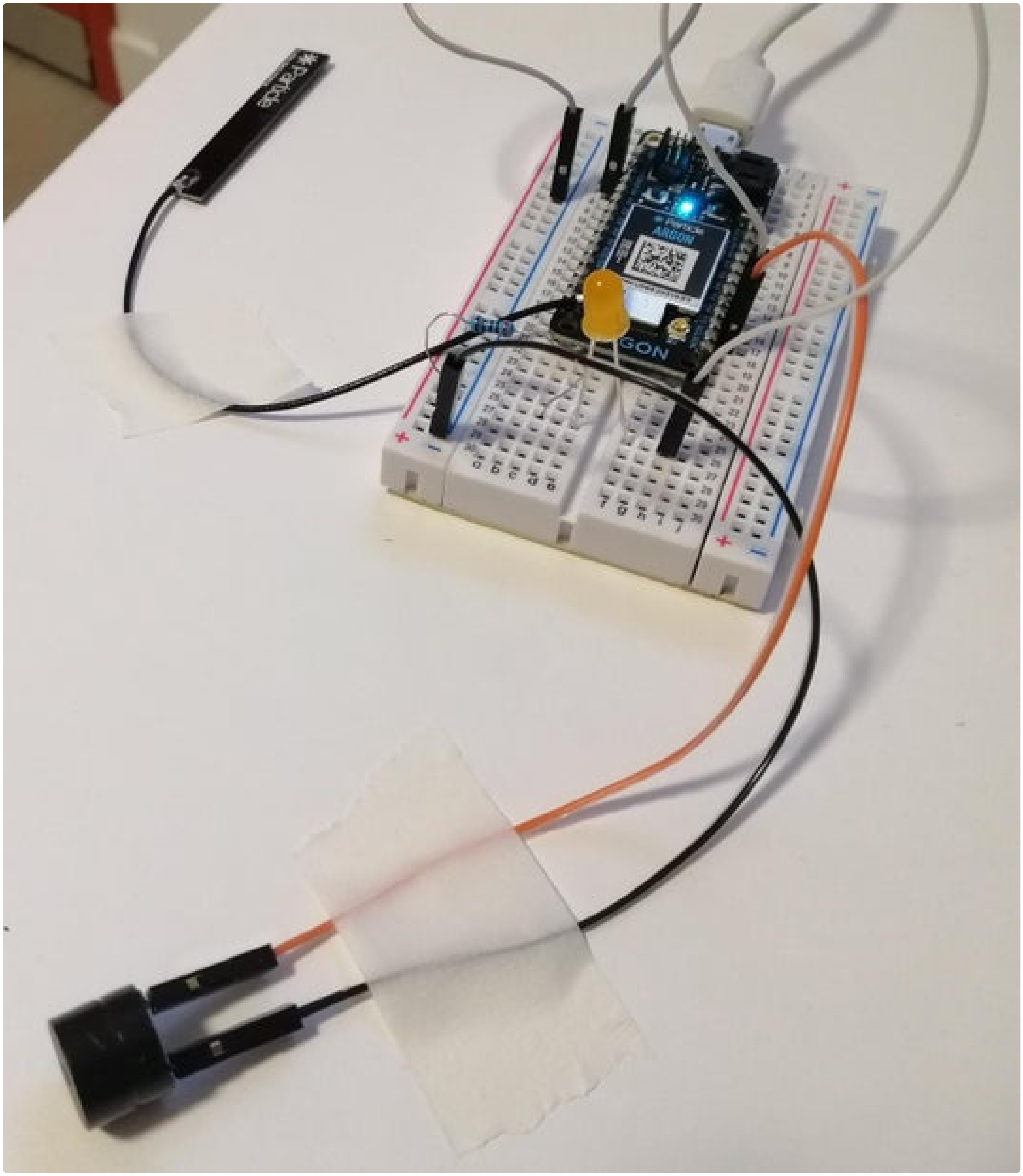
Step 10: Test the System

You can now test the system by running the code, then exposing the light sensor to bright light. The servo motor should rotate and the LED along with the Buzzer should also activate. Once the light source is removed, the motor should rotate back to its original position. You should also receive a notification via the IFTTT app on your mobile phone.

This also works when you manually activate and deactivate the motor using the GUI. If you have the relevant applets set up, you should also receive app notifications when manually triggering the motor.




TOWER PRO™
Micro Servo
9g
SG90



Step 11: ...And You're Done!

Congratulations, you have successfully integrated the two systems to create a Plant Shielding System!

Step 12: Extra Notes

This tutorial is created as part of an assignment for Deakin University, School of IT, Unit SIT210 - Embedded Systems Development. This is an on the initial incomplete version published on 19/05/2022. This is the finalized version of the project (04/06/2022).

thanks for sharing.