I. Arduino UNO Code

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
  #include <avr/power.h>
#endif

//NeoPixel Stuff
#define LED_PIN 5
#define NUM_PIXELS 8
#define LIM 0.5
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_PIXELS, LED_PIN);
int currentData;
uint32_t colorValue;

#define SOUND_CTRL_PIN A0
#define SOUND_PIN A1
#define WEB_CTRL_PIN1 A2
#define WEB_CTRL_PIN2 A3
#define CURRENT_PIN1 A4
#define CURRENT_PIN2 A5
#define CH1_ENABLE 12
#define CH2_ENABLE 13

int x = 0;
int y = 0;

float current_1 = 0;
float current_2 = 0;

int raw_current_1;
int raw_current_2;
float LEDCurrentData;
int sound_en;

int rgb_current_1;
int rgb_current_2;

int sound = 0;
bool sound_control = false;
bool on_from_sound = true;

void setup() {
  pinMode(SOUND_CTRL_PIN, INPUT);
  pinMode(SOUND_PIN, INPUT);
  pinMode(WEB_CTRL_PIN1, INPUT); //Current sensor data
  pinMode(WEB_CTRL_PIN2, INPUT); //Current sensor data
  pinMode(CURRENT_PIN1, INPUT);
  pinMode(CURRENT_PIN2, INPUT);
  pinMode(CH1_ENABLE, OUTPUT); //Website control
  pinMode(CH2_ENABLE, OUTPUT); //Website control
  Serial.begin(9600);

  strip.begin();    //Initialize neopixel LED strip
  strip.setBrightness(50);
  strip.show(); //Initialize all pixels to off
}

void loop() {

  //Sound Functionality
  sound = analogRead(SOUND_PIN);  //Get raw sound data from mic
  //Check if website enabled control
  sound_en = analogRead(SOUND_CTRL_PIN);
  if (sound_en > 614) {
    sound_control = true;
  }
  else {
    sound_control = false;
  }

  /*******************************************
   * Either sound control or website control
   * can be activated, this is decided based
   * on the sound control button on the web
   * server. Relays are then enabled/disabled
   * based on the current mode
   *******************************************/
  if (sound_control == true) {
    //Check if sound is detected
    if (sound < 150 || sound > 350) {
      on_from_sound = !on_from_sound;   //Change light status
    }

  //Write to relays based on current sound status
    if (on_from_sound) {
      digitalWrite(CH1_ENABLE, HIGH);
    digitalWrite(CH2_ENABLE, HIGH);
   }
   else {
    digitalWrite(CH1_ENABLE, LOW);
    digitalWrite(CH2_ENABLE, LOW);
   }
  }
  else {
   x = analogRead(WEB_CTRL_PIN1);  //Read ctrl signal from RPi
   if (x > 614) {
    digitalWrite(CH1_ENABLE, HIGH);
   }
   else {
    digitalWrite(CH1_ENABLE, LOW);
   }

   y = analogRead(WEB_CTRL_PIN2);   //Read ctrl signal from Rpi
   if (y > 614) {
   digitalWrite(CH2_ENABLE, HIGH);
   }
   else {
    digitalWrite(CH2_ENABLE, LOW);
   }
  }

  //Current Data and LED Control
  //185mV is the output indicating there is 1A of current going though the current sensor
  //The analogRead function can read up to 5V with a maximum resolution of 1024 bits,
so 0-1023 are possible values
  raw_current_1 = analogRead(CURRENT_PIN1); //Read in data from current sensor
  raw_current_2 = analogRead(CURRENT_PIN2); //Read in data from current sensor
  //Voltage output for current sensors is Vcc/2 + 185mV/A
  current_1 = ((raw_current_1 - 511) * 0.0049)/.185;
  current_2 = ((raw_current_2 - 511) * 0.0049)/.185;
  String sentence_1 = "Current through channel 1: ";
  String sentence_2 = "Current through channel 2: ";
  String amps = "A";
  String comma = ", ";
  String channel_data = sentence_1 + current_1 + amps + comma + sentence_2 +
current_2 + amps;
  Serial.println(channel_data);//Put in current data that is being sent as string

  LEDCurrentData = current_1 + current_2;
  LEDCurrentData = (LEDCurrentData / LIM) * 255;   //Scale to strip.Color input value
range
  colorValue = strip.Color(LEDCurrentData, 255-LEDCurrentData,0);
  //Flash all LEDs with designated color
  for(int i=0; i<NUM_PIXELS; i++)
  {
    strip.setPixelColor(i, colorValue);
    strip.show();
  }
  delay(250);
}
```

II. Raspberry Pi 0 Code

A. Python

```
'''

STRUCTURE OF CODE SOURCED FROM
https://randomnerdtutorials.com/raspberry-pi-web-server-using-flask-to-control-gpios/

'''

import RPi.GPIO as GPIO
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)

import serial
import time

ser=serial.Serial("/dev/ttyUSB0",9600)
ser.baudrate = 9600

GPIO.setmode(GPIO.BCM)

# Create a dictionary called pins to store the pin number, name, and pin state:
pins = {
   23 : {'name' : 'Channel 1', 'state' : GPIO.LOW},
   24 : {'name' : 'Channel 2', 'state' : GPIO.LOW},
   25 : {'name' : 'Sound Control', 'state': GPIO.LOW}
   }

# Set each pin as an output and make it low:
for pin in pins:
   GPIO.setup(pin, GPIO.OUT)
```

```
    GPIO.output(pin, GPIO.LOW)

@app.route("/")
def main():
    # For each pin, read the pin state and store it in the pins dictionary:
    for pin in pins:
        pins[pin]['state'] = GPIO.input(pin)

    read_ser = ser.readline()
    # Put the pin dictionary into the template data dictionary:
    templateData = {
        'pins' : pins,
        'read_ser' : read_ser
        }
    # Pass the template data into the template main.html and return it to the user
    return render_template('main.html', **templateData)

# The function below is executed when someone requests a URL with the pin number and action in it:
@app.route("/<changePin>/<action>")
def action(changePin, action):
    global start_time
    # Convert the pin from the URL into an integer:
    changePin = int(changePin)
    # Get the device name for the pin being changed:
    deviceName = pins[changePin]['name']
    # If the action part of the URL is "on," execute the code indented below:
    if action == "on":
        #if system was previously off, start timer
        if (GPIO.input(changePin) == GPIO.LOW):
            start_time = time.time();
        # Set the pin high:
        GPIO.output(changePin, GPIO.HIGH)
        # Save the status message to be passed into the template:
        message = "Turned " + deviceName + " on."
        # Check elapsed time
        if (time.time() - start_time > 3600):
            GPIO.output(23, GPIO.LOW)
            GPIO.output(24, GPIO.LOW)
            return redirect(url_for('action', changePin = changePin, action = "off"))
        #Start/reset timer
    if action == "off":
        GPIO.output(changePin, GPIO.LOW)
        message = "Turned " + deviceName + " off."
```

```
# For each pin, read the pin state and store it in the pins dictionary:
   for pin in pins:
      pins[pin]['state'] = GPIO.input(pin)

   # Along with the pin dictionary, put the message into the template data dictionary:
   read_ser = ser.readline()
   templateData = {
      'pins' : pins,
      'read_ser' : read_ser
   }


   return render_template('main.html', **templateData)


if __name__ == "__main__":
   app.run(host='0.0.0.0', port=80, debug=True)
```

B. HTML

```
<!DOCTYPE html>
<meta http-equiv="refresh" content="2">
<head>
   <title>Remote AC Switch</title>
   <!-- Latest compiled and minified CSS -->
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7
" crossorigin="anonymous">
   <!-- Optional theme -->
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css"
integrity="sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r"
crossorigin="anonymous">
   <!-- Latest compiled and minified JavaScript -->
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"
crossorigin="anonymous"></script>
</head>

<body>
   <h1>Remote AC Switch Web Server</h1>
```

```
  {% for pin in pins %}
  <h2>{{ pins[pin].name }}
  {% if pins[pin].state == true %}
     is currently <strong>on</strong></h2><div class="row"><div class="col-md-2">
     <a href="/{{pin}}/off" class="btn btn-block btn-lg btn-default" role="button">Turn
off</a></div></div>
  {% else %}
     is currently <strong>off</strong></h2><div class="row"><div class="col-md-2">
     <a href="/{{pin}}/on" class="btn btn-block btn-lg btn-primary" role="button">Turn
on</a></div></div>
  {% endif %}
  {% endfor %}
  <h3>Current Reading</h3>
  <h3>{{ read_ser }}</h3>

</body>
</html>
```